AN ABSTRACT OF THE THESIS OF

Guoning Chen for the degree of Doctor of Philosophy in Computer Science presented on June 24, 2009.

Title:

Topological Analysis, Visualization,and Design of Vector Fields on Surfaces

Abstract approved: _____

Eugene Zhang

Analysis, visualization, and design of vector fields on surfaces have a wide variety of major applications in both scientific visualization and computer graphics. On the one hand, analysis and visualization of vector fields provide critical insights to the flow data produced from simulation or experiments of various engineering processes. On the other hand, many graphics applications require vector fields as input to drive certain graphical processes. This thesis addresses vector field analysis and design for both visualization and graphics applications.

Topological analysis of vector fields provides the qualitative (or structural) information of the underlying dynamics of the given vector data, which helps the domain experts identify the critical features and behaviors efficiently. In this thesis, I introduce a more complete vector field topology called *Entity Connection*

Graph (ECG) by including *periodic orbits*, an essential component in vector field topology. An efficient technique for periodic orbit extraction is introduced and incorporated into the algorithm for ECG construction. The analysis results are visualized using the improved evenly-spaced streamline placement with all separation features being highlighted. This is the first time that periodic orbits have been extracted from surface flows. Through applications to engine simulation datasets, I demonstrate how the extracted topology helps engineers interpret the flow data that contains certain desirable behaviors which indicate the ideal engineering process.

Accuracy is typically of paramount importance for visualization and analysis tasks. However, the trajectory-based vector field topology approaches are sensitive to small perturbations such as error and noise which are contained in the given data and introduced during data acquisition and processing. This makes rigorous interpretation of vector field topology and flow dynamics difficult. To overcome that, I advocate the use of *Morse decomposition* to define a more reliable vector field topology called *Morse Connection Graph (MCG)*. In particular, I present the pipeline of Morse decomposition of an input vector field. A technique based on the idea of $\tau-map$ is introduced to produce desirably fine Morse decompositions of vector fields. To address the issue of slow performance of the global $\tau-$map framework, I describe a hierarchical MCG refinement framework. It enables the $\tau-$map approach to be conducted within a Morse set of interest which greatly reduces the computation cost and leads to faster analysis. It is my hope that the work on Morse decomposition will invoke the

investigation of other similar data analysis problems such as scalar field and tensor field analysis.

The techniques of time-independent vector field design have been well-studied. However, there is little attention on the systematic design of time-varying vector fields on surfaces. This dissertation addresses this by developing a design system that allows the creation and modification of time-varying vector fields on surfaces. More specifically, I present a number of novel techniques to enable efficient design over important characteristics in the vector field such as *singularity paths*, *pathlines*, and *bifurcations*. These vector field features are used to generate a vector field by either blending basis vector fields or performing a constrained optimization process. Unwanted singularities and bifurcations can lead to visual artifacts, and I address them through singularity and bifurcation editing. I demonstrate the capabilities of the design system by applying it to the design of two types of vector fields: the orientation field and the advection field for the application of texture synthesis and animation.

Topological Analysis, Visualization, and Design of Vector Fields on Surfaces

by

Guoning Chen

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 24, 2009
Commencement June 2010

Doctor of Philosophy thesis of Guoning Chen presented on June 24, 2009.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Guoning Chen, Author

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Professor Eugene Zhang, whose wisdom, encouragement, and support have helped me through my Ph.D. study at Oregon State University. He has been a role model for me because of not only his rigorous attitude in research but also kindness toward people. I cherish what I have learned from him. It has been my honor to work with him.

I would like to thank Professor Konstantin Mischaikow whose mathematical expertise has set the solid backbone of the presented work. I appreciate every discussion with him, which has helped me grow as a serious researcher.

I wish to thank Professor Robert S. Laramee who provides me not only the datasets used in this thesis, but also invaluable suggestions and help in every aspect of my research and academic career. It has always been my pleasure to collaborate with him.

I would also like to thank Professor Peter Wonka who brought me the exciting project of street modeling. Our collaboration has led to a wide range of possibilities of future research. I highly appreciate his help in the development of my career.

I thank Dr. Vivek Kwatra for sharing his texture synthesis and animation tool to enable the project of time-dependent vector field design. I also thank

Professor Andrzej Szymczak for providing the efficient computation algorithm of Conley index. I am also grateful for the valuable discussion with Professor Gerik Scheuermann, Professor Harry Yeh, and Dr. Paweł Pilarczyk. I wish to thank Dr. Pascal Mueller for creating the beautiful images for the street modeling project.

My thanks goes to Professor Mike Bailey and Professor Ronald A. Metoyer. From them I have learned not only cutting-edge graphics knowledge and techniques but also how to be a good teacher.

I would like to thank all my colleagues, Jonathan Palacios, William Brendal, Nadia Payet, Madhu Srinivasan, Zhongzang Lin, Qingqing Deng, Gregory Esch, Patrick Neil, Randy Rauwendaal, Mizuki Kagaya, and all other members of the graphics group at Oregon State University. Working with them has brought me knowledge and happiness. I thank all of my friends for always being around me and providing me the priceless support.

Finally, I would express my deepest gratitude to my parents for their love and unconditional support through my whole life. Most of all, I am utterly grateful to my wife, Rui Lu, for her love, support, and always being by my side. I would also thank my parents-in-law for their support during the early life time of our lovely son, Kevin. I love you all.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1 – Introduction

Vector fields are of particular interest in many scientific and engineering processes including Computational Fluid Dynamics (CFD), aerodynamics, weather study, automotive and aircraft design, and tsunami and hurricane modeling. Studying the vector fields derived from these applications provides insights to the behaviors of the underlying *dynamical systems*, which enables the evaluation and control of these processes. Vector fields also have a wide range of applications in computer graphics. For instance, in the applications of surface parameterization, remeshing, non-photorealistic rending, texture synthesis, and computer animation, vector fields are typically required as the initial input. To that end, extracting the essential characteristics from the given vector fields and developing effective techniques for the design of various vector fields are the two themes of the presented work. In this thesis, I will restrict the discussion of analysis to the context of time-independent vector fields. Similarly, since the problems on time-independent vector field design have been well-studied [8, 21, 71, 101], this work will focus on a harder problem, i.e, time-varying vector field design on surfaces.

1.1 Topological Analysis of Vector Field

Vector field visualization and analysis has been successfully applied to help interpret the dynamical systems of a wide variety of applications. Among all analysis techniques, vector field topology conveys the qualitative (i.e. structural) information of the underlying dynamics of the given vector fields. Vector field topology consists of a number of *recurrent* flow features and their connectivity information. It is typically expressed as a *topological graph* with these features of interest as nodes and their connectivity relations as edges. The corresponding embedded graph partitions the flow domain into sub-regions with equivalent qualitative characteristics. Therefore, this graph allows an efficient interpretation of the essential dynamics of the flow.

There are two requirements for computing vector field topology: accuracy and efficiency. Accuracy requires a high fidelity of the recovered information in the composed images with respect to the given data. It is of paramount importance for visualization and analysis tasks. Efficiency provides the evaluation of the computation performance, which determines the practicality of an analysis technique. Both accuracy and efficiency are addressed in the topological analysis techniques introduced in this dissertation. First, in order to provide the engineers a more complete topology of their vector data, I extend the topological graph of vector fields by including an essential component, *periodic orbits*, that was not incorporated before, and define *Entity Connection Graph (ECG)*. Second, I examine the instability of trajectory-based topology representations of vector fields, and pro-

pose *Morse Connection Graph (MCG)* as a more reliable representation of vector field topology.

1.1.1 Entity Connection Graph (ECG)

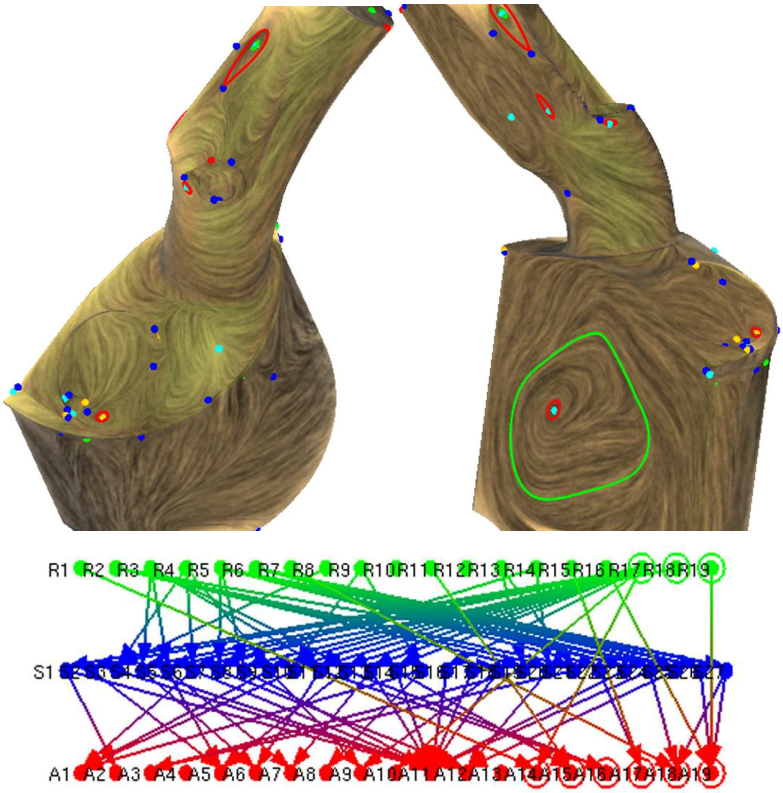


Figure 1.1: The visualization of CFD data simulating in-cylinder flow through a gas engine from two viewpoints (top), and the corresponding ECG (bottom). Through the application of the periodic orbit extraction algorithm we can observe a closed streamline about a central axis corresponding to the ideal pattern of tumble motion in the gas engine simulation results. This is precisely the type of re-circulation that the engineers strive to realize when designing the intake ports of a gas engine cylinder.

Helman and Hesselink introduced the notion of flow topology to the visualization community in [27, 28]. Since then, much research has been carried out on topological analysis of vector fields in the past two decades (Chapter 3). Most of these techniques focus on the analysis of local features such as *fixed points* which represent local recurrent behavior or stagnant points. Little work addresses the detection of large-scale recurrent features such as periodic orbits. *Periodic orbits* are essential structures of *non-gradient* vector fields, such as those in electromagnetism, chemical reactions, fluid dynamics, locomotion control, population modeling, and economics. There is a fundamental need to be able to incorporate them into the subject of vector field visualization. For example, in the application of automobile engine design and combustion simulation, the existence and locations of the periodic orbits provide clues to the swirl motion inside the chamber of the engine. Efficient periodic orbit detection and visualization can help design engineers better understand how the shape of the chamber and the initial speed of the fluid through the intake ports impact engine efficiency [8].

To include the detected periodic orbits into the topological graph, I extend the traditional vector field topological skeleton and introduce the entity connection graph (ECG) (see Figure 1.1). Efficient algorithm of computing an ECG of a given vector field is introduced as well as a novel and practical algorithm for periodic orbit extraction. The ECG computation significantly reduces the computation time even for large datasets constituting hundreds of thousands of sample points. The results are visualized using an enhanced streamline-based method in which periodic orbits and separatrices are highlighted. This is particularly desirable for

vector fields on surfaces since only portions of a periodic orbit may be visible for any given viewpoint (see Figure 1.2). In addition, I introduce a technique that allows the user to create periodic orbits on surfaces. Finally, I provide a general framework and efficient algorithms that allow topological simplification on arbitrary vector fields defined on surfaces.

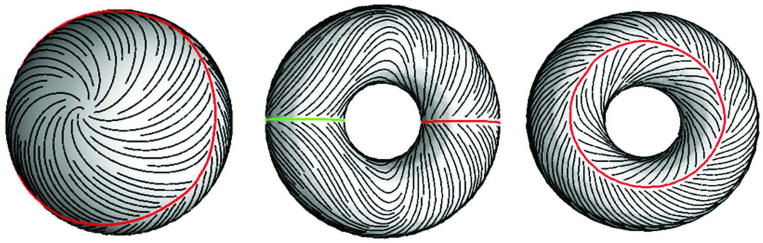


Figure 1.2: Example vector fields created using our design system. The visualization makes use of the enhanced streamline-based method proposed in this work. The red cycles represent the attracting periodic orbits and the green ones represent the repelling periodic orbits.

1.1.2 Morse Decompositions of Vector Fields

Reliable analysis of vector fields is crucial for the rigorous interpretation of the flow data stemming from engineering applications. However, any numerical or experimental method is subject to errors and thus one must be concerned with whether these errors are significant enough to produce misleading information. In the domain of numerical analysis the existence of spurious solutions would be an example of such misleading information. The reason is because the vector field topology based on individual trajectories is sensitive to errors and noise.

In order to address this challenge I present a rather different approach for the representation, extraction and visualization of flow topology. The representation of the global dynamics is realized in terms of an acyclic directed graph called the *Morse connection graph* (MCG). The nodes in this graph, which I refer to as *Morse sets*, correspond to polygonal regions in the phase space, which I define to be *Morse neighborhoods*. All the recurrent dynamics is contained in the Morse neighborhoods. The edges in an MCG indicate how the flow moves from one Morse neighborhood 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 [35].

My contributions in this work lie in the following. First, I present a theoretically sound framework based on Morse decompositions from which more rigorous statements can be made with respect to the extraction of flow topology than the individual trajectory-based analysis. Second, two approaches, the *geometry-based* method and the *τ-map based* approach, are then proposed to realize the process called *flow combinatorialization* which is considered a key step in the presented Morse decomposition pipeline. The result of this process is a directed graph $\mathcal{F}$ whose nodes are the underlying polygonal primitives (e.g. triangles) and edges indicate the flow dynamics (e.g. the advection of particles from one triangle to another). The geometry-based approach constructs $\mathcal{F}$ by simply examining the flow behaviors across mesh edges, which typically leads to coarser MCG than desired (see Figure 1.3, (a)). On the other hand, τ-map based method resolves this

issue by constructing a more accurate $\mathcal{F}$ using the idea of *outer approximation* computation. It keeps track of the image of each polygonal primitive advected by the flow over time τ. This improvement results in finer MCGs than the ones using a geometry-based method (see Figure 1.3, (e)). However, the τ-map approach requires a large amount of tracing operations. Furthermore, because an ideal τ value is typically not known for a given flow, the user must carry out multiple computations with different τ values before a satisfying result is achieved. This leads to a slow analysis process which can be prohibitive for large datasets. To approach that, I propose an efficient Morse decomposition framework based on a hierarchical refinement process. This framework determines a Morse set of a coarse MCG to refine based on a number of criteria including the Conley index of this Morse set. The refinement is then conducted locally inside the Morse neighborhood of this Morse set. It is worth noting that an efficient algorithm to compute the upper bound of the Conley index of a Morse set based on flow combinatorialization graph is introduced to assist the automatic refinement process.

1.2 Time-Varying Vector Field Design on Surfaces

In this section, I turn to a different but related topic: vector field design. A wide variety of computer graphics applications require vector fields as the input, such as texture synthesis [21, 38, 43, 44, 49, 84, 91, 101], non-photorealistic rendering [29, 30, 101], fluid simulation [65], hair modeling [23], crowd animation [11], and shape deformation [89]. The design and control of steady (time-independent) vector fields

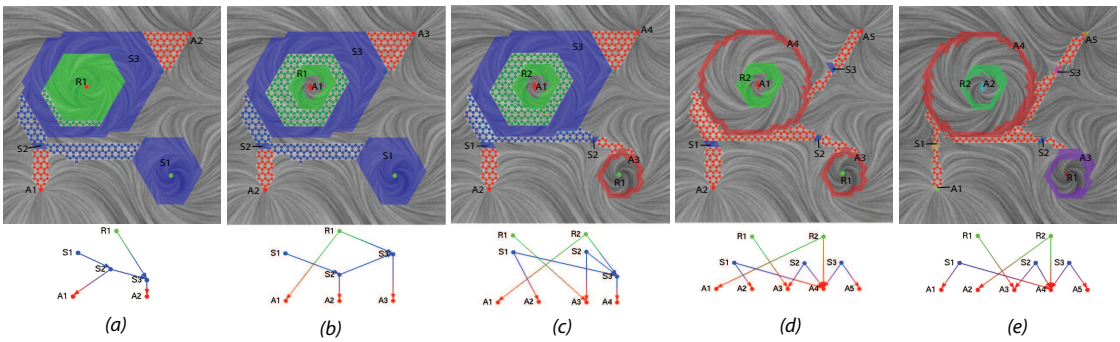


Figure 1.3: This figure provides an example of locally refining a Morse decomposition of an analytic flow data over different Morse sets (b): R1, (c): S1, and (d): S3 with $\tau = 7.8$, 10, 10, respectively. (a) provides the Morse decomposition using a geometry-based method. In addition, the MCG generated using a global τ-map idea with $\tau = 12$ is also shown in (e) for comparison. Note how our finest MCG is comparable to the one using global Morse decomposition. Different colors indicate different Morse sets. The color-dotted regions indicate the connection between Morse neighborhoods. Note that the connection regions are also refined during the process. In the MCGs, green dots stand for the source Morse sets, red dots for the sink Morse sets, and blue dots for the saddle Morse sets.

on two dimensional manifolds has been well explored in the recent years [8,21,101]. In contrast, there has been relatively little work in designing time-varying vector fields despite the potential benefits in applications such as controlled fluid and crowd simulation, shape deformation design, hair animation, artistic rendering of videos, and texture synthesis and animation. I address the problem of the design of time-varying vector fields on surfaces in this work.

More specifically, I have identified time-varying vector field design as an important problem in computer graphics. I present an approach to this problem by designing parameterized vector fields to approximate time-varying vector field design. This enables the techniques for steady vector field design to be extended to

time-varying vector field design. I describe the disparate usage of various vector field characteristics in diverse computer graphics applications. This distinction shows the need of design of different types of time-varying vector fields to achieve various effects. Particularly, I show the design of orientation field and advection field for the orientation and movement of the texture structures in texture synthesis and animation. I present a number of techniques to enable different vector field features such as singularity paths, pathlines, and bifurcations to be input as user specifications. The extended radial basis field approach and an extended constrained optimization technique are introduced to produce a parameterized vector field with coherent transition. I provide operations to support topological control in a parameterized vector field, including bifurcation removal and movement, and singularity movement. To my best knowledge, this is the first time a time-varying vector field design tool is developed for the general purpose of computer graphics applications.

1.3 Thesis Structure

The structure of the thesis is as follows.

Vector Field Analysis (ECG): The mathematical foundations of time-independent vector fields on surfaces are introduced in Chapter 2. An important concept, *Conley index* that is used to classify flow features in general is introduced as well as its computation algorithm. The entity connection graph and an efficient algorithm for its construction are introduced in Chapter 3. A novel and efficient approach

for periodic orbit extraction is also described in this chapter. Based on the ECG, I investigate a number of topological simplification scenarios in Chapter 4 and provide a uniform framework to accomplish these simplifications. Chapter 5 describes efficient techniques to create periodic orbits in surface vector fields which has been applied to create the synthetic vector fields used in this dissertation. An enhanced streamline-based technique is presented for time-independent vector field visualization which highlights the embedded ECG structure of the given vector field during the visualization.

Vector Field Analysis (MCG): I review the basic concepts of Morse decompositions and describe a general pipeline for the efficient computation of Morse decompositions of vector fields in Chapter 6. The Morse decompositions based on the idea of τ-map approach is detailed in Chapter 7 to compute finer MCGs. A hierarchical framework for the fast computation is proposed in Chapter 8. In addition, the computation of the upper bound of a Conley index is first introduced to the visualization community.

Time-Varying Vector Field Design on Surfaces: Chapter 10 reviews the time-varying vector field background, especially an important concept known as *bifurcation*. The concept of *parameterized vector field* is introduced. In Chapter 11, I present a novel design system for the creation and control of time-varying vector fields on surfaces. A number of novel interfaces for the creation of a time-varying vector field is introduced. The design vector fields have been applied to guide the texture synthesis and animation to convey various effects on surfaces such as fluid animation, caustic reflection, lava, and weather animation.

Chapter 2 – Topological Analysis of Time-Independent Vector Field

Vector field topology depicts the qualitative structure of a given flow. This structural information typically provides critical insights to the underlying dynamics of the given data. For instance, in the study of combustion process of an engine using CFD techniques, the locations where fuels enter (i.e. intakes) and leave (i.e. outlets) typically correspond to *sources* and *sinks* of the flow in the cross sections perpendicular to the flow direction at intakes or outlets, respectively. Sources and sinks are examples of topological features called *fixed points*. The encounter of fuel and oxygen induces stretching behaviors which may exhibit as *saddles* in the flow patterns (see the front of the gas engine in Figure 1.1, upper-left). If the engine is well designed, the two different materials are then gradually mixed and their motion starts forming circular patterns, i.e. *periodic orbits* (see the back of the gas engine in Figure 1.1, upper-right). Therefore, through detecting these topological features, the engineer is able to evaluate the designed engine. In this chapter, I briefly review a number of important topological concepts that will be applied in the following chapters where detailed computation algorithms are discussed.

2.1 Previous Work

Conventional vector field topology, such as *topological skeleton of vector fields* was introduced by Helman and Hesselink [27, 28] to the visualization community. It consists of fixed points as nodes and certain special integral curves known as *separatrices* connecting these fixed points as edges. This embedded graph-like configuration partitions the flow domain into different regions. Inside each region, the flow dynamics possesses similar qualitative nature [78]. This allows the domain experts to abstract the structure of the flow dynamics and extracts the essential information of the flow dynamics efficiently.

Following Helman and Hesselink, much research has been conducted to address the computation of vector field topology in 2D vector fields. Tricoche et al. [80] and Polthier and Preuß [53] present efficient algorithms to locate fixed points in a vector field. Scheuermann et al. extend the work on first-order fixed points to the analysis of higher-order fixed points using *Clifford algebra* and present solutions to the visualization of higher-order fixed points [62]. The approaches of visualizing non-linear topology of a given vector field are presented in their work [61, 62]. In addition to fixed points, periodic orbits are also essential structures of *non-gradient* vector fields. There is a fundamental need to be able to incorporate them into the subject of vector field visualization. To approach that, Wischgoll and Scheuermann [97] 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. This technique has also been extended to

3D vector fields and time-dependent flows [98]. Theisel et al. [74] propose a mesh-independent technique for the detection of periodic orbits in planar flow. Interested readers can find a complete survey of these aforementioned work in [39, 40, 54, 94]. However, the conventional vector field topology does not incorporate periodic orbits into the topological graph. In this work, I develop a system for vector field visualization and analysis that extracts and visualizes boundary flow topology. It provides the user with a variety of capabilities in that fixed points, periodic orbits, and separatrices can be identified (see Figure 2.1). A comprehensive vector field topology, called *entity connection graph (ECG)* is defined to include periodic orbits. Furthermore, an efficient periodic orbit extraction algorithm is proposed which permits the efficient extraction of these recurrent features and can be applied to surface flow. This technique has been applied to the analysis of the engine simulation data and airfoil simulation and experiment data. It is the first time periodic orbit extraction and visualization has found utility in a real application.

2.2 Time-Independent Vector Fields

The control of vector fields on surfaces is realized using concepts from the topological theory of dynamical systems. Consider a manifold M and a subset $X \subset M$. The boundary of X is denoted by ∂X and closure by $\mathrm{cl}(X)$. A vector field and the corresponding topological concepts can be defined as follows.

Definition 2.2.1 *A time-independent* vector field *can be expressed in terms of a differential equation* $\dot{x} = V(x)$ *(where* $\dot{x} = \frac{dx}{dt}$*). The set of solutions to it gives*

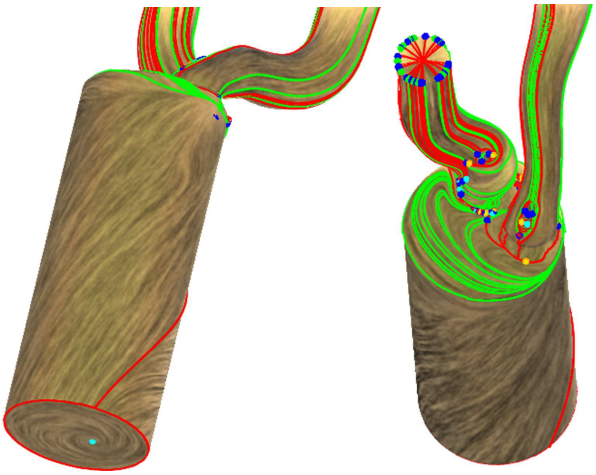


Figure 2.1: The visualization of CFD data simulating in-cylinder flow through a diesel engine from two viewpoints. Compare them to the idealized flow shown in Figure 3.6 (left). Figure 3.5 provides complementary visualization of the flow inside the diesel engine. Both the texture and the topology-based visualizations indicate a nice pattern of swirl motion at the boundary of the combustion chamber while the regions near the intake ports reveal deviation from the ideal.

rise to a flow *on* M*; that is a continuous function* $\varphi : \mathbf{R} \times M \to M$ *satisfying* $\varphi(0, x) = x$*, for all* $x \in M$*, and*

$$\varphi(t, \varphi(s, x)) = \varphi(t + s, x) \tag{2.1}$$

for all $x \in M$ *and* $t, s \in \mathbf{R}$*.*

Definition 2.2.2 *Given* $x \in M$*, its* trajectory *is*

$$\varphi(\mathbf{R}, x) := \cup_{t \in \mathbf{R}} \varphi(t, x). \tag{2.2}$$

where

$$\varphi(t, x) = x + \int_0^t V(\varphi(s, x))ds$$

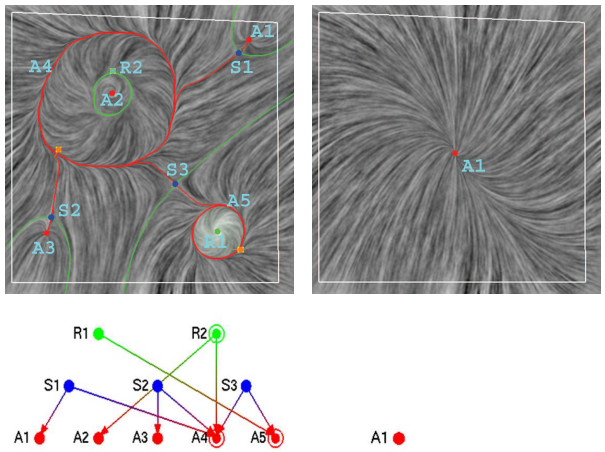


Figure 2.2: An example vector field (upper left) and its ECG (lower left). The vector field contains a source (green), three sinks (red), three saddles (blue), a repelling periodic orbit (green), and two attracting periodic orbits (red). Separatrices that connect a saddle to a repeller (a source or a periodic orbit) are colored in green, and to an attractor (a sink or a periodic orbit) are colored in red. The fixed points and periodic orbits are the nodes in the ECG (lower left) and separatrices are the edges. In addition, a periodic orbit can be connected directly to a source, sink, or another periodic orbit. Such connections are also depicted as edges in the ECG. The simplified field of (upper left) is shown in (upper right) and its corresponding ECG is (lower right). Notice the Conley index for both vector fields inside the white loop are the same, which allows the vector field in the left to be simplified into the one shown in the right.

Definition 2.2.3 *$S \subset M$ is an* invariant set *if $\varphi(t, S) = S$ for all $t \in \mathbf{R}$.*

Definition 2.2.4 *A point $x \in M$ is a* fixed point *if $\varphi(t, x) = x$ for all $t \in \mathbf{R}$. More generally, x is a* periodic point *if there exists $T > 0$ such that $\varphi(T, x) = x$. The trajectory of a periodic point is called a* periodic orbit*.*

Remark: Observe that for every $x \in M$, its trajectory is an invariant set. Other simple examples of invariant sets include fixed points and periodic orbits.

2.2.1 Hyperbolicity and Limit Sets

Consideration of the important qualitative structures associated with vector fields on a surface requires familiarity with *hyperbolic* fixed points, periodic orbits, and separatrices which I define as follows.

Definition 2.2.5 *Let x_0 be a fixed point of a vector field $\dot{x} = V(x)$; that is $V(x_0) = 0$. The linearization of V about x_0, results in a 2×2 matrix $Df(x_0)$ which has two (potentially complex) eigenvalues $\sigma_1 + i\mu_1$ and $\sigma_2 + i\mu_2$. If $\sigma_1 \neq 0 \neq \sigma_2$, then x_0 is called a* hyperbolic fixed point.

Observe that on a surface there are three types of hyperbolic fixed points: *sinks* $\sigma_1, \sigma_2 < 0$, *saddles* $\sigma_1 < 0 < \sigma_2$, and *sources* $0 < \sigma_1, \sigma_2$. Because I consider systems with invariant sets such as periodic orbits, the definition of the limit of a solution with respect to time is non-trivial. The *alpha* and *omega limit sets* of $x \in M$ are

$$\alpha(x) := \cap_{t<0}\mathrm{cl}(\varphi((-\infty, t), x)), \quad \omega(x) := \cap_{t>0}\mathrm{cl}(\varphi((t, \infty), x))$$

respectively. A periodic orbit Γ is *attracting* if there exists $\epsilon > 0$ such that for every x which lies within a distance ϵ of Γ, $\omega(x) = \Gamma$. A *repelling* periodic orbit can be similarly defined ($\alpha(x) = \Gamma$)). Finally, given a point $x_0 \in M$, its trajectory is a *separatrix* if the pair of limit sets $(\alpha(x), \omega(x))$ consist of a saddle fixed point and another object that can be a source, a sink, or a periodic orbit. Figure 2.2 provides an example vector field (upper-left). Fixed points are highlighted by colored dots

(sources: green; sinks: red; saddles: blue). Periodic orbits are colored in green if repelling and in red if attracting. Separatrices that terminate in a source or a repelling periodic orbit are shown in green and those terminate in a sink or an attracting periodic orbit are colored in red. For convenience, I will refer to a source and a sink as a *node* in the remainder of the thesis wherever appropriate.

2.2.2 Entity Connection Graph (ECG)

As indicated before, conventional topological skeleton of vector fields consists of fixed points and their connectivity information. This connectivity typically corresponds to the separatrices that start from saddles and end at other fixed points. I have pointed out that periodic orbits are important recurrent features in the flow (i.e. examples of invariant sets) that are interesting to flow experts. However, periodic orbits are not incorporated in the conventional topological skeleton which makes this representation of vector field topology incomplete. To overcome that, I define *Entity Connection Graph (ECG)* which includes periodic orbits as an essential component of the topological graph.

Definition 2.2.6 *An* Entity Connection Graph*, or* ECG *is a topological graph* $G = (S, E)$ *where* S *represents the extracted fixed points and periodic orbits, and* E *indicate the direct connections of the connected pairs in the flow.*

To better understand the structure of an ECG, consider a vector field V on a surface S that contains at least a fixed point or periodic orbit, i.e., the ECG of V is not empty. V induces a partition of S. Each sub-region in the partition is a *basin*

that can be bounded by fixed points, periodic orbits, and/or separatrices (e.g. the region bounded by $A2$ and $R2$ in Figure 2.2 upperleft). A streamline inside a basin flows from a source object α to a destination object ω. Both α and ω can be a node fixed point (a source or a sink) or a periodic orbit. In addition, for each of the three cases (node-node, node-periodic orbit, and periodic orbit-periodic orbit), the link between α and ω can be either direct, i.e., there is an edge connecting them in the ECG, or indirect, i.e., they are connected to some common saddles through separatrices. Note that a periodic orbit separates nearby flow into two parts. On either side, there can be one or more basins. When there is one basin, the periodic orbit is directly linked to a node or another periodic orbit. In the case of multiple basins, the periodic orbit is linked to other nodes or periodic orbits through saddles.

2.2.3 Poincaré Index

There is a topological descriptor, called *Poincaré index* that has been applied to locate and characterize an isolated fixed point. A fixed point x_0 is called *isolated* if there exists a neighborhood N surrounding x_0 such that x_0 is the unique fixed point in the interior of N. The definition and computation of Poincaré index is based on the rotation of a vector field along a simple closed curve. Mathematically, the Poincaré index of a block N is defined as the *winding number* of the *Gauss map* along ∂N, the boundary curve of N [102]. In other words, the index of a simple closed curve Γ in a plane relative to a continuous vector field $V = (V_X, V_Y)^T$ is the

number of the positive field rotations while traveling along Γ in positive direction (i.e. counter-clockwise), which is denoted as [78].

$$I(\Gamma, V) = \frac{1}{2\pi} \oint_{\Gamma^+} d\theta$$

where θ is the angle between V and $x-$axis satisfying

$$\cos\theta = \frac{V_X}{\sqrt{V_X^2 + V_Y^2}}, \quad \sin\theta = \frac{V_Y}{\sqrt{V_X^2 + V_Y^2}}$$

Note that $\theta \in [0, 2\pi)$ (i.e. modulo 2π). According to this definition and computation, the Poincaré index of a fixed point free region is 0, and the isolated fixed points can be characterized as follows:

if $I(\partial N, V) = 1$, x_0 is either a source or a sink;

if $I(\partial N, V) = -1$, x_0 is a saddle.

From these results, we observe that Poincaré index is not able to distinguish the difference between a source and a sink. Furthermore, the Poincaré index for a periodic orbit is zero, which equals that of an emptyset. Therefore, Poincaré index theory does not provide enough utility to handle periodic orbits, thus limiting its potential uses. In the next, I will introduce a more general topological descriptor that can be used to characterize invariant sets including fixed points and periodic orbits.

2.3 Conley Index

In this section, I introduce a topological invariant, called *Conley index* that can be used to classify the types of different invariant sets . Furthermore, it will be employed in the simplification process (Chapter 4) which provides a topological constraint on the possible simplification or modification of the vector field within the *isolating neighborhood* (or *isolating block*) of cancellation.

Definition 2.3.1 *A compact set $N \subset M$ is an* isolating neighborhood *if for all $x \in \partial N$, $\varphi(\mathbf{R}, x) \not\subset N$.*

That is, the flow enters or leaves N eventually everywhere on ∂N.

An invariant set S is *isolated* if there exists an isolating neighborhood N such that S is the maximal invariant set contained in N. Observe that hyperbolic fixed points and periodic orbits are examples of isolated invariant sets. Isolated invariant sets posses two essential properties. First, there are efficient algorithms for identifying isolating neighborhoods [35]. Second, there exists an index, i.e. Conley index [45], that identifies the types of modifications to the structure of the invariant set that are topologically permissible. For example, the Conley index of the vector field shown in Figure 2.2 (upper-left) inside the white loop is identical to that of a sink. Topological simplification of the complex field inside the region can result in the field shown in the right.

I now turn to the definition of Conley index. The rigorous definition of Conley index requires the introduction of *homology* and *cohomology* [70] which is beyond the scope of this dissertation. Given the 2D spatial discretization (a triangulation),

it is possible to define Conley index of an isolating neighborhood through the homotopy type of the quotient space of this neighborhood. This also leads to an efficient computation algorithm of the Conley index.

2.3.1 Definitions of Conley Index

The Conley index of a set M is easy to define if M is an *isolating block* (i.e. isolating neighborhood), i.e. if every point x on the boundary of M is an exit point or an entry point. An entry point is a point x whose trajectory for sufficiently small negative times is outside M. Similarly, x is an exit point if its trajectory is outside M for all sufficiently small positive times. Let S be the maximal invariant set in the isolating block M. The Conley index of S is the relative homology [33] of the index pair (M, L). Because we are restricting our attention to flows on orientable surfaces, for an isolating block M, its Conley index is the homotopy type of the quotient space M/L where L is the *exit set*: the subset of the boundary of M consisting of all exit points. The quotient space is the pointed topological space obtained from M by identifying all points of L to a single distinguished point.

A few simple examples of isolating blocks with exit sets shown in red can be found in Figure 2.3. Any Morse set computed using the geometry method is an isolating block. However, Morse sets arising from flow combinatorialization are not guaranteed to be isolating blocks. Therefore, they require a different technique for Conley index estimation which will be described in Section 8.3.

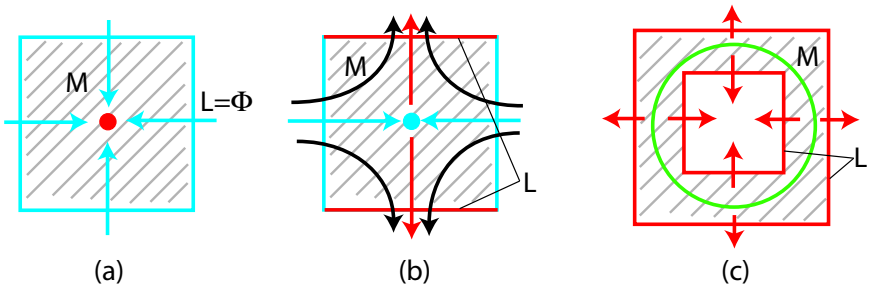


Figure 2.3: A number of simple examples of isolating blocks. (a) shows a region containing a sink $(1, 0, 0)$; (b) is a region with a saddle $(0, 1, 0)$; (c) displays a ring-like region enclosing a repelling periodic orbit $(0, 1, 1)$. In all cases, M is the shadow region. Red lines represent the exit sets.

2.3.2 Computing Betti Numbers of A Quotient Space

Because the Conley index defined as the homotopy type is hard to deal with, I represent it by its Betti numbers $CH_*(M) = (\beta_0, \beta_1, \beta_2)$ that are computed as described below. In what follows, β_k denotes the k-dimensional Betti number of M/L. Assume that M is a subset of a two-dimensional manifold surface, a triangulation of M is available and that L is a union of boundary edges of M.

β_0 simply counts the number of connected components in M/L that do not contain the distinguished point (i.e. disconnected point), i.e. the number of connected components in M that are disjoint with L. Thus, β_0 is easy to compute. In particular, if M is connected, then β_0 is zero if $L \neq \emptyset$ and 1 otherwise. In the left isolating block shown in Figure 2.3, there is one single connected component, and there is no exit set. Therefore, the β_0 for this block is 1. For the other two blocks, there is also only one connected component for each case, but there exit sets are non-empty. Therefore, the β_0's for these two isolating blocks shown are both 0.

β_2 is equal to the number of connected components of M whose entire boundary is contained in L. This is because if there is a boundary edge of M that is not in L, the pair (M, L) can be reduced to a homotopy equivalent pair where both sets are one-dimensional by means of elementary collapses [69]. On the other hand, the formal sum of all triangles in each connected component M_0 of M whose boundary is contained in L defines a generator of the two-dimensional homology group of M/L and these generators span the entire homology group [18]. Based on this criterion, the β_2's for the first two isolating blocks shown in Figure 2.3 are 0 because there are no pure exit boundary. The β_2 of the right block is 1 since both its boundaries belong to the exit set.

To compute β_1, one can view the quotient space M/L as a two-dimensional CW-complex whose two-, one- and zero- dimensional cells correspond to triangles, edges and vertices (respectively) in $M \setminus L$. Additionally, there is one special zero-dimensional cell that corresponds to L. Therefore, $\beta_k = 0$ for $k > 2$. Also, the Euler characteristic of the quotient space, $\chi(M/L)$, is easy to compute: it is equal to $n_2 - n_1 + n_0$, where n_i is the number of its i-dimensional cells of M/L with the distinguished point taken out. On the other hand, $\chi(M/L) = \beta_2 - \beta_1 + \beta_0$ ([18]). Since β_2 and β_0 are already known, this equation uniquely determines β_1. Note that the Euler characteristic of the quotient space can also be computed from the characteristics of M and L: $\chi(M/L) = \chi(M) - \chi(L)$ [18]. In summary, β_1 can be

computed given the information above:

$$\beta_1 = \beta_0 + \beta_2 - (\chi(M) - \chi(L))$$
$$= \beta_0 + \beta_2 - (|V(M)| + |F(M)| - |E(M)| - (|V(L)| - |E(L)|))$$

where $|V(\cdot)|$, $|E(\cdot)|$, and $|F(\cdot)|$ represent the number of vertices, edges, and faces of $(\cdot)$. Note that the Euler characteristic of L is computed as $(|V(L)| - |E(L)|)$ because there is no 2-dimensional components for a one-dimensional curve.

Figure 2.3 provides a number of simple cases of Conley index computation. For instance, consider a region M containing a sink (Figure 2.3, left). Clearly $\chi(M) = 1$ since M is a topological disk and hence $\chi(M/L) = \chi(M) - \chi(L) = 1$. $\beta_0 = 1$ and $\beta_2 = 0$ according to previous analysis. β_1 can be determined to be 0 from the equation $\chi(M/L) = \beta_2 - \beta_1 + \beta_0$. Hence the Conley index for a sink (represented as the vector of Betti numbers $(\beta_0, \beta_1, \beta_2)$) is $(1, 0, 0)$. Similarly, one can compute the Conley index of a saddle (middle), which is $(0, 1, 0)$. In this case, the Euler characteristic of the quotient space is $\chi(M/L) = \chi(M) - \chi(L) = 1 - 2 = -1$. Now, let us examine the case of a repelling periodic orbit (right). In this case, $\beta_2 = 1$ since M is connected and L is equal to its boundary. $\beta_0 = 0$ because the only connected component of M intersects L. Also, $\chi(M) = 0$ and $\chi(L) = 0$ (M is homotopy equivalent to a circle and L - to two circles). Hence $\chi(M/L) = 0$ and $\beta_1 = \beta_2 + \beta_0 - \chi(M/L) = 1$. The index is $(0, 1, 1)$.

In terms of Morse decomposition and MCGs, given the three Betti numbers of the Conley index, a Morse set can be classified as follows. If $\beta_0 = 1$, it is a

sink-like Morse set (colored in red); if $\beta_2 = 1$, it is a source-like Morse set (colored in green); otherwise, it is a saddle-like Morse set (colored in blue).

The most important Conley indices are as follows:

$$\begin{aligned}
x_0 \text{ an attracting fixed point(e.g.sink)} &\Rightarrow CH_*(x_0) = (1,0,0) \\
x_0 \text{ a saddle fixed point} &\Rightarrow CH_*(x_0) = (0,1,0) \\
x_0 \text{ a repelling fixed point(e.g.source)} &\Rightarrow CH_*(x_0) = (0,0,1) \\
\Gamma \text{ an attracting periodic orbit} &\Rightarrow CH_*(\Gamma) = (1,1,0) \\
\Gamma \text{ a repelling periodic orbit} &\Rightarrow CH_*(\Gamma) = (0,1,1) \\
S = \emptyset &\Rightarrow CH_*(S) = (0,0,0)
\end{aligned}$$

It shows that Conley index is a more general topological descriptor than Poincaré index in the characterization of invariant sets.

2.4 Morse Decomposition and Morse Connection Graph

Even for flows restricted to surfaces, invariant sets can be extremely complicated and cannot be assumed to consist of hyperbolic fixed points, periodic orbits and separatrices [51]. Furthermore, even if the recurrent dynamics is restricted to fixed points and periodic orbits, it is impossible to develop an algorithm that will identify all of them. For example, it is easy to generate continuous vector fields that contain infinitely many isolated fixed points and/or periodic orbits. Even for continuous piecewise linear vector fields, it is not clear whether infinitely many

isolated periodic orbits may exist. Further investigation is required to answer this question. Thus it requires a language that allows us to manipulate a broader but useful class of invariant sets.

In this section, I introduce an important process to which I will resort during the topology computation.

2.4.1 Morse Decompositions

Central to our effort is the need for a computationally robust decomposition of invariant sets.

Definition 2.4.1 *A* Morse decomposition*, $\mathcal{M}(S)$, of S consists of a finite collection of isolated (or disjoint compact) invariant subsets of S, called* Morse sets*,*

$$\mathcal{M}(S) := \{M(p) \mid p \in \mathcal{P}\} \tag{2.3}$$

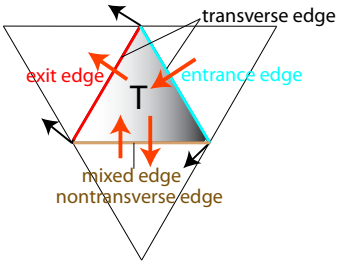
such that if $x \in S$, then there exists $p, q \in \mathcal{P}$ such that $\alpha(x) \subset M(q)$ and $\omega(x) \subset M(p)$.

It has proved that any structures associated with recurrent dynamics of φ_λ, i.e. fixed points, periodic orbits, chaotic dynamics, must lie in the Morse sets [35]. Furthermore, there exists a partial order $>$ on $\mathcal{P}$ satisfying $q > p$ if there exists $x \in S$ such that $\alpha(x) \subset M(q)$ and $\omega(x) \subset M(p)$. Let $C(p, q) := \{x \in M \mid \alpha(x) \subset M(p) \text{ and } \omega(x) \subset M(q)\}$. An efficient means of presenting the partial order on a Morse decomposition is given by the associated *Morse Connection Graph*

(MCG) which is the minimal directed graph whose vertices consist of the Morse sets $\{M(p) \mid p \in \mathcal{P}\}$ and whose directed edges $M(q) \to M(p)$ imply $q > p$. Note that a MCG contains supplementary information with respect to the topological skeleton presented by Helmann and Hesselink [28]. For example, consider the idealized magnetic field over the Earth's surface in which only two fixed points exist and none of the connecting orbits between them is a separatrix. Similarly, a periodic orbit can be connected to a source (Figure 1.2, left) or another periodic orbit (Figure 1.2, middle) without any separatrices in the field. The detail of the computation of MCG and the pipeline of Morse decomposition will be presented in the later chapters.

2.4.2 Computation of Morse Decompositions

Computing a Morse decomposition and its associated MCG can be done as follows. Let $\mathcal{T}$ denote a triangulation of the phase space. An edge in this triangulation is classified as a *transverse edge* if the flow leaves one of incident triangles completely (a one-way road, or *exit/entrance edge*). Otherwise, the edge is *nontransverse* (two-way, or *mixed edge*). See the inlet image for an illustration. Construct equivalence classes on $\mathcal{T}$ using the following relationship and transitivity. Two triangles $T_0, T_1 \in \mathcal{T}$ are equivalent if $T_0 \cap T_1$ consists of a nontransverse edge. Taking the union of all triangles in an equivalence class produces a polygonal region, whose boundary consists



of transverse edges only. This results in an isolating neighborhood by definition. According to the classification of the triangle edges, a directed graph can be constructed. The nodes of this directed graph are the individual triangles of the triangulation. A directed edge is inserted between two neighboring triangles if the flow direction points from one triangle to the other crossing the common edge of these two triangles. For instance, consider an edge e shared by triangles T_0 and T_1. If e is an exit edge with respect to triangle T_0 (i.e. the vector field points from T_0 to T_1), a directed edge pointing from T_0 to T_1 is inserted to the directed graph. Figure 2.4 illustrates such a construction. The orange arrows (right) represent the obtained directed edges. It is proven in [35] that the maximal invariant sets within the strongly connected path components of this directed graph produce a Morse decomposition for the vector field and furthermore, the MCG can be obtained from the tree that results from the collapsing each strongly connected component to a single vertex. Standard algorithms [12] indicate that this procedure can be performed in linear time in the number of vertices and edges in the graph. In particular, the flow experts are interested in strongly connected components with non-trivial Conley index (0,0,0). This technique is sufficient for the computation of a Morse decomposition of a vector field, which I refer to as the *geometry-based method*. It enables the definition and computation of the following vector field topology analysis. A general pipeline of Morse decomposition in practice will be presented in Chapter 6.

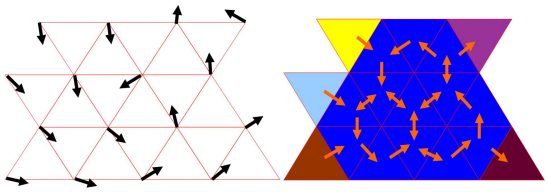


Figure 2.4: An example for building a directed graph based on the input vector field defined on a triangular mesh. In the obtained directed graph, each node refers to a particular triangle, the direction of each directed edge are determined by the type of the edge. Based on the input vector field (left), I build the directed graph and compute the strongly connected components in the graph (right).

2.4.3 MCG vs. ECG

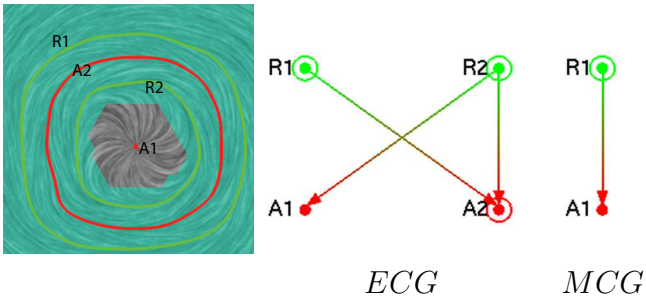


Figure 2.5: This example shows the difference between ECG and MCG for a piecewise linear vector field created using our tool. The vector field shown in the left contains one fixed point and three periodic orbits. Therefore, the ECG consists of four nodes (middle). However, due to the resolution of the underlying mesh, there are only two Morse sets (colored regions) with one containing the fixed point and the other containing the periodic orbits. Consequently, there are two nodes in the MCG (right).

A node in the MCG is an isolated invariant set, which may contain multiple fixed points and periodic orbits (Figure 2.5). For many engineering applications, such as the study of in-cylinder flow, engineers are often more concerned with individual fixed points and periodic orbits. Therefore, there is a need to build a

graph $\mathcal{G}$, whose nodes consist of fixed points and periodic orbits. Similar to an MCG, the edges in $\mathcal{G}$ represent the connectivity information between the nodes according to the vector field. This graph is an Entity Connection Graph (ECG). Based on this discussion, an ECG is a refinement of the MCG of the same vector field. Figure 2.2 (lower-left) shows an ECG of the vector field in the upper-left. Here $\mathcal{P}$ is the set of labels ($R1$ and $R2$, $S1$-$S3$, and $A1$-$A5$), and $M(p)$ is the actual object that p represents, i.e., $M(R1)$ is a source. In fact, an MCG can be obtained from the corresponding ECG by merging nodes that are in the same Morse set (Figure 2.5: $R1$, $R2$, are $A1$ are merged as $R1$ in the MCG). Furthermore, the MCG is equal to the ECG when the vector field has a finite number of fixed points and periodic orbits, all of which have an isolating neighborhood of their own. Figure 2.5 shows the difference between the MCG and ECG of a piecewise linear vector field created with our system.

Given that the ECG is a refinement of the MCG, the reader may wonder why I emphasize the existence of both graphs. There are two reasons. The first is that I make use of information from the MCG to compute the ECG. The second has to do with the validity of the information which will be discussed in detail in Chapter 6. Note that the ECG will not be complete without boundary analysis.

2.5 Vector Field Representation

I now describe the computational model of our system. In this model, the underlying domain is represented by a triangular mesh. Vector values are defined at the

vertices only, and interpolation is used to obtain values on the edges and inside triangles. This applies to vector field editing, simplification, and analysis such as fixed point and periodic orbit extraction.

For the planar case, I use the popular piecewise linear interpolation method [80]. On curved surfaces, I borrow the interpolation scheme of Zhang et al. [101], which guarantees vector field continuity across the vertices and edges of the mesh. These interpolation schemes support efficient flow analysis operations on both planes and surfaces.

Chapter 3 – ECG Computation

In this chapter, I describe the detail of the computation of the ECG of given a vector field. In brief, I first extract the topological features from the flow including fixed points and periodic orbits (Section 3.1). Then, the connectivity between these detected features is computed to construct the ECG (Section 3.2). In addition, the constructed ECG is visualized in the embedded fashion in the flow field to highlight the critical structural information (Section 3.4).

3.1 Periodic Orbit Extraction

Periodic orbits are essential features in a non-linear vector field, we need the ability to detect and locate periodic orbits in a fast and accurate manner. In this section, I present a new algorithm for periodic orbit identification

The proposed periodic orbit detection method is inspired by Wischgoll and Scheuermann [97], in which they locate periodic orbits in a planar vector field by starting streamline tracing from a neighborhood of a fixed point and keeping track of repeated cell cycles. While this method is capable of detecting periodic orbits in many situations, it assumes that any periodic orbit can be approached by a fixed point, which is not always true. One example case is the repelling periodic orbit (the green closed curve) between the two surrounding attracting orbits (the

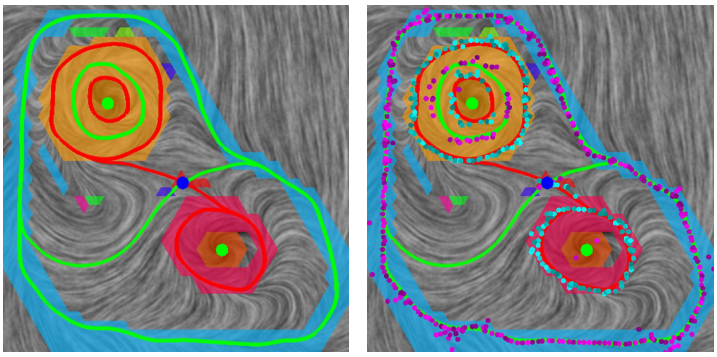


Figure 3.1: An example of the presented periodic orbit detection algorithm. First, I compute strongly connected components and only consider components where periodic orbits may exist (left: colored regions). Next, I extract attachment points (right: cyan) and separation points (right: magenta) on the interior edges in these connected components. By combining the ideas of strongly connected components with the extraction of attachment and separation points, the presented algorithm is fast and efficient in finding periodic orbits.

red closed curves) in Figure 3.1. To detect periodic orbits even when they are not approached by any fixed point, I have developed a new periodic orbit detection method that has drawn ideas from the Morse decomposition [20] and separation and attachment lines [36].

A periodic orbit must situate inside a region of flow recurrence, which corresponds to certain types of strongly connected components in the domain (Section 2.4). Note that a strongly connected component does not contain a periodic orbit if it either consists of a single triangle or is a topological disk that contains no fixed point. Figure 3.1 shows an example of the strongly connected components that may contain periodic orbits (left: colored regions). The actual periodic orbits are highlighted for visualization purpose.

Recall that multiple periodic orbits may exist in an isolated Morse set (a

strongly connected component, Figure 3.1, left). To extract individual periodic orbits in a fast and efficient manner, we need a good geometric indicator as to which strongly connected components might contain periodic orbits. Kenwright presents efficient techniques in extracting open and closed separation and attachment lines [36]. I now apply these ideas to periodic orbit extraction. The algorithm is as follows:

1. *Step 1*: I compute the strongly connected components of a directed graph derived from the flow 2.4.2. In addition, the components that do not contain a periodic orbit are discarded, i.e., if a component S consists of a single triangle or if S is a topological disk that contains no fixed points. Let $\mathcal{S}$ be the set of strongly connected components that may contain a periodic orbit.
2. *Step 2*: I extract the attachment and separation points for every edge in the *interior* of a strongly connected component in $\mathcal{S}$.
3. *Step 3*: For every strongly connected component $S \in \mathcal{S}$, I start streamline tracing for each attachment point in S according to the flow. If the streamline reaches a fixed point or the boundary of S, I stop tracing and discard the attachment point. Otherwise, the streamline will approach an attracting periodic orbit. In case the periodic orbit has been discovered previously, it will be ignored. Otherwise, the periodic orbit is recorded, and a sequence of dense and evenly-spaced points are placed along the orbit. These points allow tracing from subsequent attachment points to quickly determine whether it is approaching an existing or new periodic orbit.

4. *Step 4*: I locate the repelling periodic orbits by repeating step 3 with the following two modifications: tracing will now (1) start from separation points, and (2) be in the backward direction of flow.

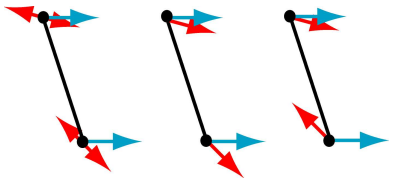


Figure 3.2: An example scenario in which inconsistent tensor assignment can lead to false separation or attachment points. In the left image, given the vector values u at the vertices of an edge (cyan arrows) and the Jacobian tensor (red arrows represent the major eigenvectors e_1), it is clear that there is not any separation point on the edge. However, by converting the tensor field into a vector field (middle and right) and evaluating $e_1 \times u$ can cause false separation point to appear (right).

Kenwright evaluates $e_i \times u$ at the vertices of the edge and use linear interpolation to locate attachment and separation points. This formulation assumes that an eigenvector field can be treated as a vector field. However, as pointed out by Zhang et al. [103], treating an eigenvector field as a vector field will lead to discontinuities in the vector field and cause visual artifacts in tensor field visualization and non-photorealistic rendering. I have observed similar problems during the computation of attachment and separation points. For instance, consider the example shown in Figure 3.2, in which the vector field is constant along an edge e (cyan arrows) and the Jacobian along the edge is nearly constant (major eigenvectors are shown in red bidirectional arrows). When choosing a consistent direction assignment for the eigenvectors at the vertices (middle), I conclude that no separation or

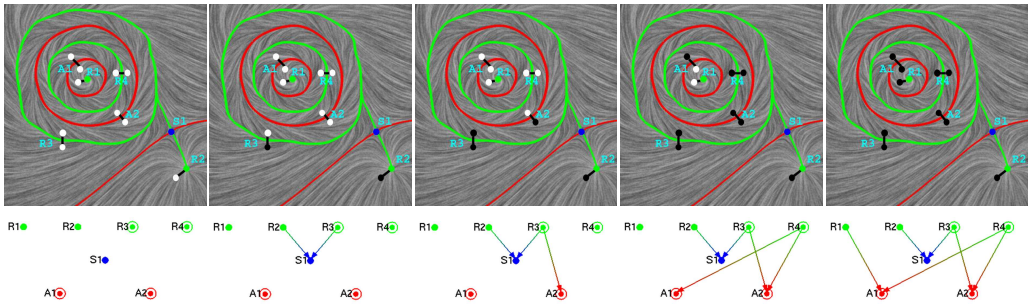


Figure 3.3: This figure illustrates the algorithm for construction of ECG's. First (a), I perform fixed point and periodic orbit extraction. I mark as unvisited (white disks) for every source/sink and for both sides of every periodic orbit. Next (b), I compute all the separatrices and mark as visited (black disks) for $R2$ and the outer side of $R3$ since they are connected to the saddle $S1$ in the ECG. In (c), I start from the inner side of $R3$ and follow the flow forward to find the link to the outer sider of $A2$. An edge is added to the ECG, and both sides in the link are now marked as visited. In (d), I perform similiar operations to the unvisited sides of every repelling orbits (both sides of $R4$) to find all the links to a sink or an attracting orbit. Finally (e), I start from any unvisited side of an attracting periodic orbit and follow the flow in the reverse direction to locate links to unmarked sources.

attachment point exists on e. However, the assignment in the right will lead to a false identification of a separation point. To overcome this problem, I simply assume the Jacobian is constant along an edge and evaluate it at the middle of an edge by performing linear interpolation on the Jacobians at the vertices. This efficiently removes the need to carefully assign directions to eigenvectors at the two vertices of an edge.

To perform tracing on surfaces, I use a Runge-Kutta scheme [7] that has been adapted to surfaces with a piecewise interpolation scheme that guarantees vector field continuity across vertices and edges [101].

It should be pointed out that the presented detection technqiue of periodic orbits detects all orbits given a triangulation of the flow domain and the piecewise linear interpolation scheme. In order to extract all the periodic orbits in the flow, we need the ability of subdividing the mesh or seeking non-linear interpolation scheme which are beyond the scope of this work.

3.2 ECG Construction and Display

I now describe how to construct the ECG for a vector field. Recall that ECG represents the fixed points, periodic orbits, and their connectivity in the given flows. To compute the ECG, I perform a three-stage operation. First, I locate the fixed points and periodic orbits. These are the nodes in the ECG. Next, I compute all the separatrices by tracing from every saddle in its incoming and outgoing directions until the trajectories end in a node or a periodic orbit. Finally, I identify

edges in the ECG that are not separatrices. The methods for fixed point extraction and separatrix computation are according to Helmann and Hesselink [28]. Periodic orbits are identified using the algorithm described in Section 3.1.

I now describe how to compute non-separatrix edges in the ECG. As discussed earlier, this corresponds to an edge in the ECG that does not involves any saddle. There are four cases: (1) a source and a sink (type 1), (2) a source and an attracting periodic orbit (type 2), (3) a sink and a repelling periodic orbit (type 3), and (4) a repelling periodic orbit and an attracting periodic orbit (type 4). Note a node (i.e. a source or a sink) can only be involved in one non-separatrix edge, and so does each side of a periodic orbit. A flag is assigned to every node. The flag is set to 1 if the node is connected to a saddle in the ECG. Otherwise, the flag is set to 0. Similarly, a flag is defined for each side of a periodic orbit to record whether there is at least one separatrix approaching the periodic orbit from that side. To compute non-separatrix edges, edges emanating from repelling orbits are first located. For each repelling periodic orbit γ and each side, if the corresponding flag is 0, I find a nearby point on that side of γ and perform tracing in the direction of the flow until the streamline terminates at a sink or an attracting periodic orbit. In case of a sink, its flag is marked as 1 and insert an edge (type 3) in the ECG. If the streamline ends in an attracting periodic orbit, the flag is marked to be 1 for the side of the attracting orbit from which the streamline approaches. An edge (type 4) is then inserted into the ECG. Notice that at the end of this step, all non-separatrix edges of types 3 and 4 are found. I now perform the same operations to all the extracted attracting periodic orbits whose side or sides are

still marked as 0, except that tracing is now done in the reverse direction of the flow. This allows us to find all type 2 edges. Finally, I go through every source that still has a flag of 0 and trace from a nearby point in the forward direction until it terminates at a sink. This will find all the type 1 edges. It appears that type 1 edges are rather uncommon. In fact, the only instance that I know of is the idealized magnetic field over a sphere, which contains two fixed points and no periodic orbits. Figure 3.3 illustrates this process of ECG construction with an example vector field that contains two sources, one saddle, and four periodic orbits. In (a), all fixed points and periodic orbits are extracted. I also mark as unvisited (white disks) for all the sources and sinks and for both sides of every periodic orbit. Next (b), I compute separatrices and mark as visited (black disks) any node or any side of a periodic orbit that is connected to a saddle. In the next stage, I start from any unvisited side of a repelling periodic orbit and follow the flow forward to locate links to a sink or an attracting orbit. In (c), such an operation found a link between the inner side of $R3$ and the outer side of $A2$, both of which are now marked as visited. Performing this operation on all the extracted repelling periodic orbits leads to (d), in which links such as $R4/A1$ and $R4/A2$ are found. Finally (e), I start from any unvisited side of an attracting periodic orbit and follow the flow in the reverse direction to locate the remaining edges in the ECG.

To display an ECG, I arrange the detected fixed points and periodic orbits in three rows, with sources and repelling periodic orbits in the top row, sinks and attracting periodic orbits in the bottom row, and saddles in the middle row

(Figures 2.2 and 2.5). I also provide the user with the capability to select an object either in the flow display or the graph display, and the presented system will highlight the object in both screens. This allows a user to navigate through a rather complex flow field with relative ease.

3.3 Applications

3.3.1 Application to Analytic Data

For all the fields designed with the presented system, I use the proposed techniques in previous sections to detect periodic orbits and construct ECG's. In addition, I have tested the presented method on other datasets generated from mathematical formulas and from fluid simulation. Figure 3.4 shows a vector field that corresponds to

$$V(x,y) = \begin{pmatrix} y \\ -x + y\cos(x) \end{pmatrix} \tag{3.1}$$

It has been proven that this system has exactly n periodic orbits in the region $\sqrt{x^2+y^2} < (n+1)\pi$ [100]. I sample the vector field at the vertices of a bounded underlying mesh, and employ the piecewise linear interpolation scheme [80] to obtain values inside triangles. The left of this figure shows the periodic orbits extracted using the proposed method, and the right portion displays the corresponding ECG. There are five periodic orbits. Notice the proposed method is able to detect periodic orbits even when there are no saddles in the field.

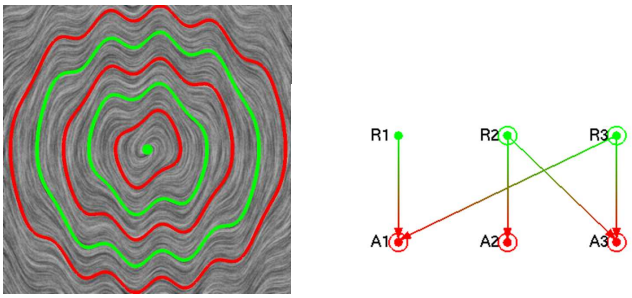


Figure 3.4: The vector field defined in Equation 3.1 over the region $\{(x, y)|max(|x|, |y|) < 11\pi\}$. There is one source in the region enclosed by five periodic orbits. The proposed algorithm was able to capture all of these orbits without requiring the presence of any separatrices.

3.3.2 Application to Engine Simulation Data

The technique has also been applied to two datasets from automotive engine simulation [42], more specifically, the design and optimization of in-cylinder flow. Engineers responsible for the design of, in this case, a diesel engine try to create an ideal pattern of motion, which can be described by a swirling flow around an imaginary axis. Achieving these ideal patterns of flow optimizes the mixture of oxygen and fuel during the ignition phase of the valve cycle. Optimal ignition leads to very desirable consequences associated with the combustion process including: more burnt fuel (less wasted fuel), lower emissions, and more output power. One type of flow, referred to as the *swirl motion*, is shown in Figure 3.6 (right). Such an ideal is often strived for diesel engines.

In Figure 3.5 I visualize the flow and its topology inside the combustion chamber from the diesel engine simulation. I have sliced through the geometry in the same manner that engineers do when analyzing the simulation results. The first slice,

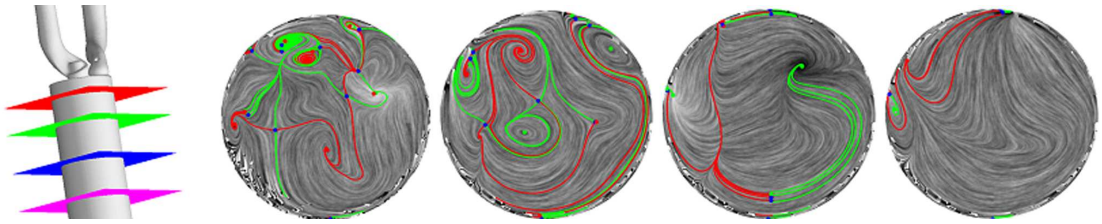


Figure 3.5: Visualizing the simulation of flow in a diesel engine: the combustion chamber (leftmost) and four planar slices of the flow inside the chamber for which the plane normals are along the main axis of the chamber. From left to right are slices cut at 10%, 25%, 50%, and 75% of the length of the cylinder from the top where the intake ports meet the chamber. The vector fields are defined as zeros on the boundary of the geometry (*no-slip condition*). The automatic extraction and visualization of flow topology allows the engineer to gain insight into where the ideal pattern of swirl motion is realized inside the combustion chamber. In fact, the behavior of the flow and its associated topology, including periodic orbits, is much more complicated than the ideal. Figure 2.1 provides complementary visualization of the flow on the boundary of the diesel engine.

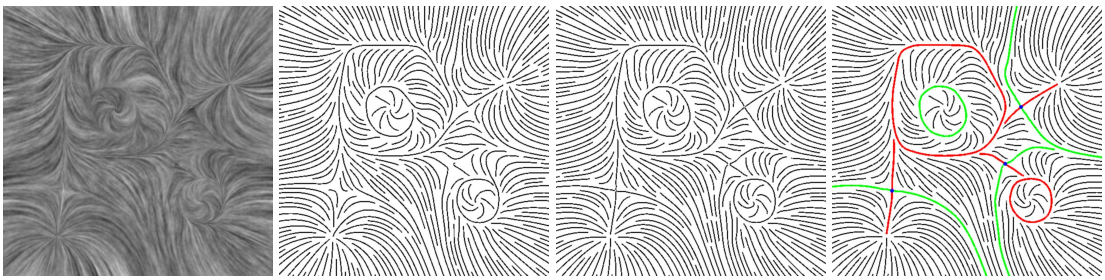
at 10% the length of the volume, indicates a swirl pattern that deviates rather strongly from the ideal -which would result in a simple recirculation orbit around the center. The second slice, at 25% down the chamber geometry we see a periodic orbit very close to the center that starts to approximate the ideal swirl motion. However, other less ideal fixed points are found near the perimeter of the geometry. The method applied here is similar to the moving cutting plane topology approach of Tricoche et al. [79]. It is noted that caution must be used when interpreting these results since the vector field has been projected onto 2D slices. On the other hand, the engineers involved are very familiar with the simulation data and are well aware of its overall characteristics.

Figure 2.1 shows from two viewpoints some simulation results in which undesired fixed points and periodic orbits are present. There are a total of 226 fixed

points and 52 periodic orbits. The total time to construct the ECG for the flow is 29.15 seconds on a 3.6 GHz PC with 3.0 GB RAM. Another type of motion, termed *tumble flow*, is shown in Figure 3.6 (left). The axis of rotation in the tumble case is orthogonal to that of the swirl case. The dataset that is being visualized (Figure 1.1) is also from simulation, and it contains 56 fixed points and 9 periodic orbits. The ECG for this dataset is shown in the bottom row. Through the application of the introduced algorithm for automatic periodic orbit extraction and visualization we can observe a closed streamline about a central axis corresponding to the ideal pattern of tumble motion in the gas engine simulation results. This is precisely the type of re-circulation that the engineers strive to realize when designing the intake ports of a gas engine cylinder. The proposed algorithm enables the CFD engineers to automatically detect and visualize this highly sought-after pattern of flow in a direct manner for the first time (see Figure 1.1). The total time for computing the ECG of this time is 31.58 seconds. The ECG produced from the diesel engine simulation results is of even higher complexity than that of the gas engine. Table 3.1 shows the complexity for both simulation datasets and the timing results in seconds.

This vector field analysis technique is also applied to other experimental and simulation data sets such as the airfoil data [25,67] and a cooling jacket data [59].

Figure 3.6: Idealized in-cylinder flow through a gas engine (left) and a diesel engine (right). Figures 1.1 and 2.1 show the visualization of CFD data simulating such flows.



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

Figure 3.7: An example of the enhanced streamline-based visualization technique on the plane: (a) a texture-based method (IBFV [86]), (b) a streamline-method [31], (c) the enhanced streamline method which uses vector field topology, (d) same image from (c) with periodic orbits and separatrices being highlighted. Notice with the proposed method (c and d), vector field topology is well-maintained by streamlines and they are easily discernable.

Table 3.1: The complexity and timing results for two CFD data simulating in-cylinder flow through a combustion engine (Figures 1.1 and 2.1). An edge in the ECG corresponds to a link between a source and destination object pair, in which both objects can be a fixed point or a periodic orbit. Times (in seconds) are measured on a 3.6 GHz PC with 3GB RAM.

dataset name	# polygons	# fixed points	# periodic orbits	# edges in ECG	time extracting fixed points	time extracting periodic orbits	time computing edges	time total
gas engine	105,192	56	9	97	0.16	22.33	9.09	31.58
diesel engine	886,296	226	52	295	3.16	21.52	4.48	29.15

3.4 Topology-Based Streamline Visualization

Visualization is crucial for the analysis and design of vector fields. Most existing visualization techniques, such as texture- and streamline-based methods, are designed for fixed points. While they perform well for illustrating local patterns such as fixed points, other features (separatrices and periodic orbits) are often not well-preserved. In Figure 3.7, a vector field with three periodic orbits is depicted using IBFV [86] (a), and evenly-placed streamlines [31] (b). Notice that it is difficult to see periodic orbits and separatrices using texture-based methods such as IBFV. Streamline-based methods can better illustrate trajectories. However, most existing methods such as Jobard and Lefer [31] and Verma et al. [88] do not take into account periodic orbits or separatrices in seed placement and streamline termination criteria. This causes visual discontinuity in periodic orbits and missing separatrices.

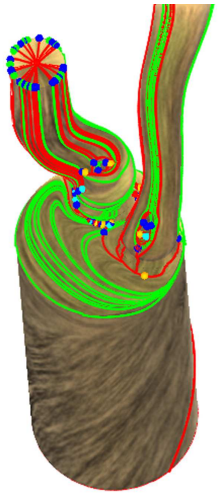
Several researchers have incorporated vector field topology into texture-based methods [93]. Most of the figures in this paper are created in that fashion. On the other hand, streamline-based methods can better illustrate individual streamlines,

which makes it an attractive approach when interactive display is not required. In this section, I describe a method for which vector field topology is used for streamline placement.

I adapt the evenly-placed streamline method of Jobard and Lefer [31] with the following modifications. First, periodic orbits and separatrices are extracted in the vector field and make them the initial streamlines. To avoid visual clusterings near sources, sinks, and periodic orbits, a separatrix is terminated if it is within a distance from the non-saddle end. Next, additional streamlines are added in the same manner as Jobard and Lefer [31]. This modification ensures that vector field topology is maintained in the visualization and no visual discontinuity for periodic orbits (Figure 3.7, c). Finally, vector field topology is highlighted with colors (d) such that attracting periodic orbits and outgoing separatrices from saddles are colored in red while repelling periodic orbits and incoming separatrices are colored in green. To avoid confusions near sources and sinks, the only fixed points included in the visualization are saddles, which are colored in blue. Figure 1.2 shows additional examples. Notice a periodic orbit on a 3D surface (middle-left and middle right) is often partially visible from any given viewpoint. They are difficult to discern without being highlighted.

Chapter 4 – Vector Field Simplification Based on ECG

Overly excessive information may be extracted from the raw flow data which is also subject to noise and error during simulations and analysis. This greatly offsets the advantages of topological analysis of vector fields and makes the interpretation difficult (see the clusters of fixed points at the top of the intake ports of the diesel engine (the inlet image from Figure 2.1, right). An efficient simplification technique is needed to extract the topology graph consisting of features of more importance. Vector field topological simplification allows the user to do so. It refers to the process of reducing the complexity of a vector field. It has many applications, such as flow visualization, texture synthesis, and non-photorealistic rendering. In this chapter, I present a uniform framework for topological simplification based of the extracted ECG of a given vector field. This framework allows the pairwise cancellation of any object pairs that are connected in the ECG.



4.1 Previous Work

In general, there are two classes of simplification techniques:
topology-based (TB), and non-topology-based (NTB) [101]. Existing NTB tech-

niques are usually based on performing Laplacian smoothing on the potential of a vector field inside the specified region. One example of these work is by Tong et al. [76], who decompose a vector field using Hodge-decomposition and then smooth each-component independently before summing them.

TB techniques simplify the topology of a vector field explicitly which reduce vector field complexity with topological guarantee. One of the earliest investigations on the subject of topology simplification in visualization was done by De Leeuw and Van Liere [13]. They make use of the distance to determine the pair of fixed points to be cancelled. In follow-up work, they perform topology simplification based on area metrics [15]. Boundary regions in the local neighborhood of sources and sinks are computed and topology is simplified based on flow regions with small areas. These techniques are applied to two important applications from vector field simulation [14]. Tricoche et al. [83] present a simplification method that also provides a piecewise analytic description for the simplified field. In this way, complementary visualizations such as texture-based methods [39] may be combined with the visualization result. They extend this method to time-dependent, 2D flows [81]. Tricoche et al. [80] also present a topology simplification method very similar to De Leuuw and Van Liere [15] however simplifications are achieved by actually modifying the vectors of the original, underlying data field. Particularly, they perform a sequence of cancelling operations on fixed point pairs that are connected by a separatrix. They refer to this operation as *pair annihilation*. A similar operation, named *pair cancellation*, has been used to remove a wedge and trisector pair in a tensor field [16]. Theisel et al. [73] present an algorithm for compressing

vector fields while preserving their topology. Later, they combine both topological simplification and topology preserving compression techniques [72]. The technique simplifies the topology of the underlying vector field based on assigning an importance to each critical point and separatrix. Features with less importance (below a certain weight threshold) are simplified. Compression is applied to the vector field with the simplified topology. Edelsbrunner et al. [19] perform pair cancellation on scalar fields defined on surfaces by changing the values of the scalar function near the fixed point pair. This is equivalent to simplifying the gradient vector field of the scalar function. I will follow this convention and refer to such an operation as fixed point pair cancellation. Zhang et al. [101] provide a fixed point pair cancellation method based on Conley theory (Chapter 2). They also extend this operation to surfaces and to fixed point pairs that are *not* connected by a separatrix, such as a center and saddle pair. For an overview of related work on vector field topology, see Laramee et al. [40]. In this chapter, I describe a more general framework for cancelling object pairs such as fixed points and periodic orbits based on the obtained ECG (Chapter 3).

4.2 Constrained Optimization

One of the essential operations in the presented system is constrained optimization, which refers to solving a vector-valued discrete Laplacian equation over a region N in the domain (a triangular mesh) where the vector values at the boundary vertices of N are the constraints. This operation is used to create periodic orbits (Chap-

ter 5) and to perform topological simplification. The equation has the following form:

$$\overline{V}(v_i) = \sum_{j \in J} \omega_{ij} \overline{V}(v_j) \tag{4.1}$$

where v_i is an interior vertex, v_j's are the adjacent vertices that are either in the interior or on the boundary of N, and V represents the vector field. The weights ω_{ij}'s are determined using Floater's mean-value coordinates [22]. Equation 4.1 is a sparse linear system, which I solve by using a bi-conjugate gradient method [55]. For convenience, a vertex v is referred to be *fixed* if the vector value at v is part of the constraints. Otherwise, v is *free*. Note that a similar formulation has been used to reduce the complexity of vector fields [101], tensor fields [2] and N-RoSy fields [50].

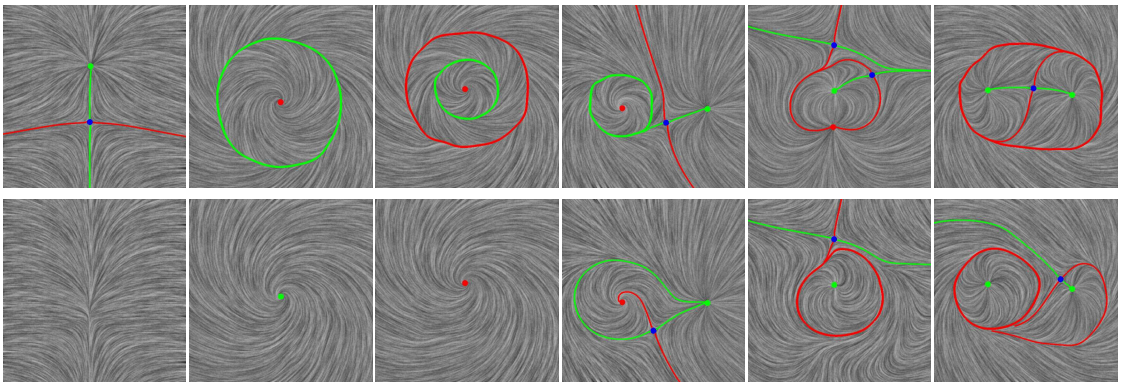
4.3 Vector Field Simplification Using Pairwise Cancellation

A well-known topological simplification operation is *pair cancellation* on a pair of fixed points with opposite Poincaré indices and a unique orbit connecting them. This operation has also been referred to as pair annihilation [80]. After cancellation, both fixed points disappear. Tricoche et al. [80] perform this operation in planar domains based on Poincaré index theory, which does not apply to periodic orbits. Zhang et al. [101] provide an efficient implementation of the pair cancel-

lation operation based on Conley index theory. They also extend fixed point pair cancellation to surfaces and for pairs that are not connected by a separatrix, such as a center and saddle pair. However, neither technique deals with periodic orbits, which limits their potential applications in visualization and graphics. Ir address this by providing a general framework that allows cancellations of a repeller and attractor pair in which either object or both can be a periodic orbit. Similar to Zhang [101], the proposed framework is based on Conley index theory. Before providing the details on the general framework, I first comment on what I mean by pair cancellation.

Pair cancellation P involves a repeller R and an attractor A. P is *direct* if there is at least one edge between R and A in the ECG, and P is *indirect* if R and A are linked through either one or two saddles. When a node or a periodic orbit is linked to a saddle through one connecting separatrix, the pair are *singly connected*. Otherwise, they are *doubly connected*. I have identified six direct cancellation scenarios (Figure 4.1) and seven indirect ones (Figure 4.2) on the plane. The presented system can handle all of these cases. To my best knowledge, previous pair cancellation methods are only available to handle case (1) in Figure 4.1.

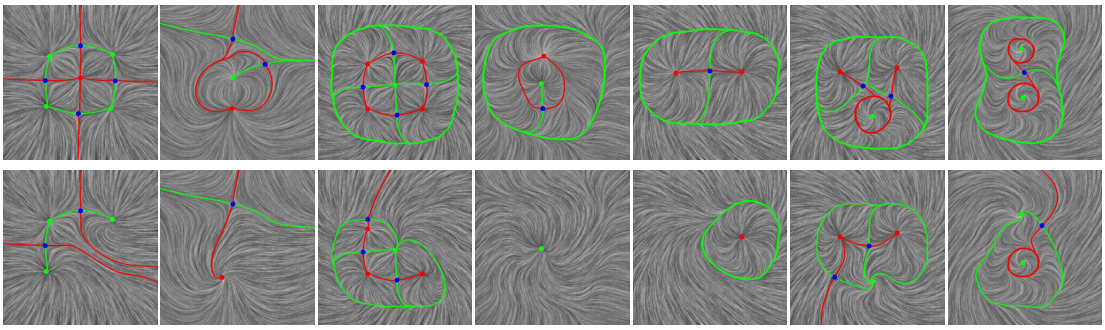
When performing pair cancellation, we expect the complexity of the flow to be reduced near the object pair, such as the case in Figure 4.1 (1). However, the reduction in the complexity does not mean the resulting flow will always be free of fixed points and periodic orbits. For instance, a sink and periodic orbit cancellation results a source as shown in Figure 4.1 (2). In fact, the characteristic of the resulting flow is constrained topologically by the Conley index of the isolating



(1) (2) (3) (4) (5) (6)

Figure 4.1: The six direct cancelation scenarios: (1) a source and saddle with a unique connecting separatrix, (2) a sink and a periodic orbit, (3) an attracting periodic orbit and a repelling one, (4) a periodic orbit and a saddle with a unique connecting separatrix, (5) a sink and a saddle with two connecting separatrices, and (6) a periodic orbit and a saddle with two connecting separatrices. The top row shows the original vector fields, while the bottom row displays the vector field after cancellation. Notice that the presented cancellation operations are only applied to the intended objects.

block over which the flow is modified. When cancelling a node and saddle pair, the Conley index of such a block is $(0, 0, 0)$, which is the same as a fixed point-free vector field. For a sink and periodic orbit pair, the Conley index is $(0, 0, 1)$ which is that of a source. To that end, the Conley index of the isolated block predicts the minimal complexity of the flow we can possibly achieve after cancellation. Furthermore, pair cancellation does not always lead to simpler behaviors, such as Figure 4.1 (5). Cancelling a doubly-connected node-saddle pair leads to a periodic orbit. In fact, the only other case in which the flow is not simplified through pair cancellation is shown in Figure 4.1 (6), where a doubly-connected periodic orbit and saddle pair is replaced by another such pair. Both cases are direct cancellations



(1) (2) (3) (4) (5) (6) (7)

Figure 4.2: The seven indirect cancellation scenarios: (1) a source and a sink pair with two saddles between them, (2) a source and a sink with one saddle between them, (3) a sink and a periodic orbit with two saddles between them, (4) a sink and periodic orbit with one saddle between them and two orbits between the saddle and the sink, (5) a sink and periodic orbit with one saddle between them and two orbits between the saddle and the periodic orbit, (6) two periodic orbits with two saddles between them, and (7) two periodic orbits with a saddle between them. The top row shows the original vector fields, while the bottom row displays the vector field after cancellation. Notice that presented cancellation operations are only applied to the intended objects.

of doubly-connect object pair. In all other cases, pair cancellation leads to simpler but not necessarily trivial flow.

I now describe the framework for a single pair cancellation that can now handle (1) periodic orbits, (2) doubly connections, and (3) indirect cancellation. Given a repeller R and an attractor A, the algorithm first searches the ECG to find the smallest interval that contains both R and A. This is achieved by finding all the nodes in the ECG that can both reach A and be reached from R. There are three possibilities: (1) R and A are directly related, (2) R and A are indirectly linked through a set of saddles S_i's, and (3) there are no paths from R to A in the ECG. Case (3) will be ignored. Note that the first stage is conducted purely on the

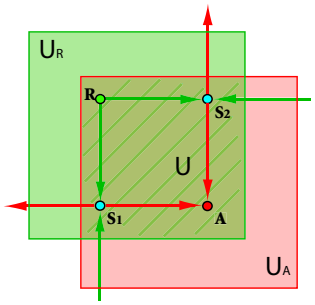


Figure 4.3: An example shows the regions obtained for cancelling a repeller R and an attractor A pair. The region growing first follows the flow forward from R and the interval S_i respectively to get U_R, then follows the flow backward from A and the interval S_i respectively to get U_A. Thus, $U = U_R \bigcap U_A$ is the region (the shadow region) in which I will perform smoothing.

graph level. Let $\mathcal{R} = \{R\} \bigcup \{S_i\}$ and $\mathcal{A} = \{A\} \bigcup \{S_i\}$. Note when R and A are directly connected, the set of $\{S_i\}$ is empty. It should also be noted that Kalies and Ban [35] provide a dimension independent algorithm for determining intervals in a Morse decomposition.

In the second stage, I consider the minimal set of triangles in the domain that contain $\mathcal{R}$. I then grow from these triangles by adding one triangle at a time across mixed or exit edges. We now have a region U_R that contains all the triangles reachable from any object in $\mathcal{R}$. Then, I perform region growing [101] from the minimal set of triangles that contain $\mathcal{A}$ by adding triangles across mixed or entrance edges. This results in a region U_A that consists of triangles that can reach any object in $\mathcal{A}$. $U = U_R \bigcap U_A$ is an isolating block that is necessary to perform pair cancellation.

In the last step, I replace the flow inside U by performing constrained optimization (Section 4.2). While this method does not guarantee that the flow will

Algorithm 1: A general framework for pair cancellation

Input: A vector field V, its ECG, a repeller R and an attractor A
Output: The vector field $\overline{V}$ after cancellation
triangles $original\_T$ and $neighbor\_T$
Begin
  Search ECG for any intermediate nodes S_i between R and A.
  T_+ = set of triangles containing either R or S_i for some i.
  Perform region growing from T_+ according to V by adding triangles across exit or mixed edges.
Let U_R be the resulting set of triangles.
  T_- = the set of triangles containing either A or S_i for some i.
  Perform region growing from T_- according to $-V$ by adding triangles across entrance or mixed edges.
Let U_A be the resulting set of triangles.
  $U = U_R \quad U_A$(See Figure 4.3)
  Perform vector field smoothing on the interior vertices of U accroding to Equation(4.1).
The resulting vector field is $\overline{V}$.
  Return $\overline{V}$.
End

be simpler, in practice I have observed that it performs well. Note that other methods can also be used to modify the flow.

This framework is illustrated as the following algorithm.

For any pair cancellation operation relying on the ECG, it is possible that region growing from the repellers and attractors can "walk" over fixed points, periodic orbits, and separatrices that are not intended for cancellation. Including these triangles in the constrained optimization may cause unwanted topological modifications. To address this issue, I tag all the triangles in the mesh that contain either a fixed point, or part of a periodic orbit or separatrix. During the construction of isolating blocks, I do not allow triangles to be added if they are tagged and contain features not intended for cancellation.

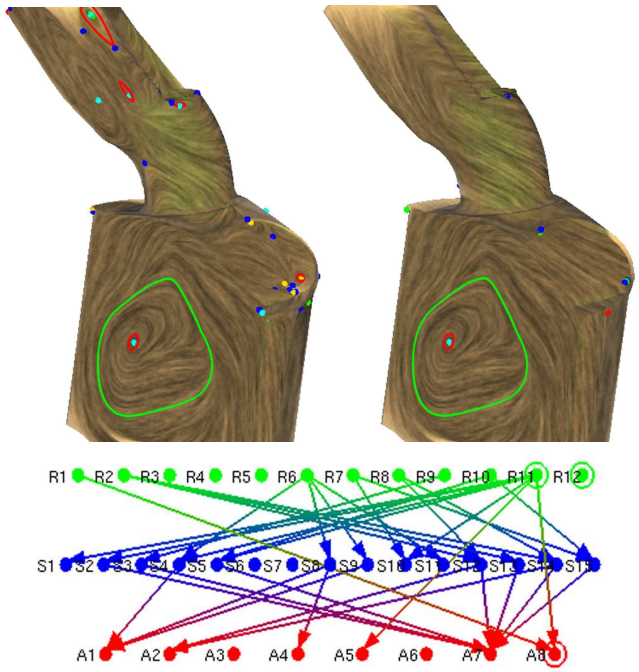


Figure 4.4: User-guided flow smoothing on CFD data simulating in-cylinder flow through a gas engine: before (upper-left) and after (upper-right). Compare the ECG after smoothing (lower) with before smoothing (Figure 1.1, lower).

4.4 User Guided Flow Smoothing

In the proceeding section, I have described techniques that automatically determine a region where the flow needs to be modified. Sometimes it is desirable to provide a user with control over the location and shape of the region. Zhang et al. [101] describe such an operation for graphics applications such as non-photorealistic rendering and texture synthesis. I apply their algorithm to large scale CFD simulation datasets. In addition, unlike Zhang et al. who only accept a topological disk, I now allow a region to have any number of boundaries. Figure 4.4 shows the results of user-guided flow smoothing on CFD simulation data of in-cylinder flow

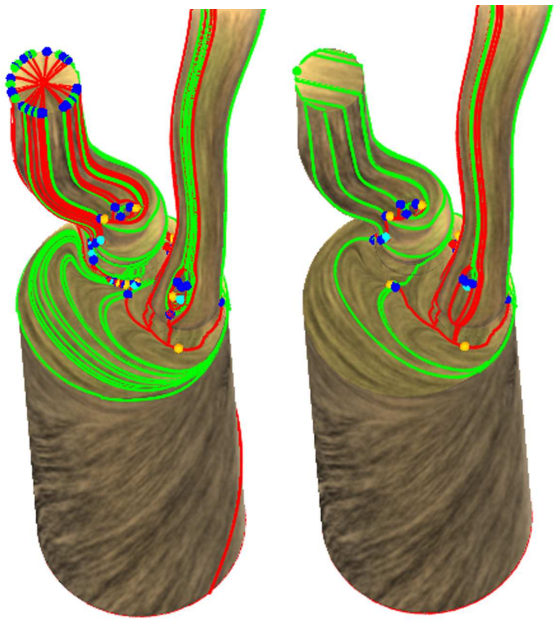


Figure 4.5: User-guided flow smoothing on CFD data simulating in-cylinder flow through a diesel engine: before (left) and after (right).

in a gas engine. The field on the upper-right was obtained by a sequence of five user-guided smoothing operations (the actual region boundaries are not shown). Notice the field is considerably simpler than the original field (upper-left). The simplified vector field retains the important larger scale tumble motion characteristics while smoothing non-ideal behavior. Also compare the ECG of the smoothed field (Figure 4.4, lower) with that of the field before smoothing (Figure 1.1, lower). Figure 4.5 compares the diesel engine dataset (left) with the one obtained from a series of six user-guided simplification operations (right). Flow smoothing is an efficient method of reducing the complexity of a vector field.

Chapter 5 – Periodic Orbit Design in Time-Independent Vector Fields

There are a wide variety of computer graphics applications require vector fields as input. For instances, vector fields are used to orient the texture patches in texture synthesis [21, 38, 43, 44, 49, 84, 91, 101], place brush strokes in non-photorealistic rendering [29, 30, 101], exert external forces in fluid simulation [65, 66], and aerodynamic animation [95] and for hair modeling [23], steer crowds [11], control texture transfer during surface deformation [17], guide surface parameterization and remeshing [57], and conduct shape deformation [89, 90]. Design and control of the vector fields with desired behaviors provides these applications an intuitive way to obtain the controllable effects with sufficient mathematical guarantee. In this chapter, I present techniques that allow the user to create periodic orbits in surface vector fields, which is a complement to the existing design features. I have applied these techniques to create synthetic vector fields that are used to experiment the proposed analysis techniques.

5.1 Previous Work

There has been some work in creating vector fields on the plane and surfaces, most of which is for graphics applications [49, 65, 84, 91]. These methods do not

address vector field topology, such as fixed points. There are a few vector field design systems that make use of topological information. For instance, Rockwood and Bunderwala [60] use ideas from geometric algebra to create vector fields with desired fixed points. Van Wijk [86] develops a vector field design system to demonstrate his image-based flow visualization technique (IBFV). The basic idea of this system is the use of basis vector fields that correspond to various types of fixed points. This system is later extended to surfaces [41, 87]. None of these methods provide explicit control over the number and location of fixed points since unspecified fixed points may appear. Theisel [71] proposes a planar vector field design system in which the user has complete control over fixed points and separatrices. However, this requires the user to provide the complete topological skeleton of the vector field, which can be labor-intensive. There has also been recent work by Weinkauf et al. [92] on the design of 3D vector fields. Recently, Zhang et al. [101] develop a design system for both planar domains and surfaces. This system provides explicit control over the number and location of fixed points through fixed point pair cancellation and *movement* operations. The presented vector field design system is inspired by their system. However, it enables the creation of periodic orbits in surface fields. To my best knowledge, this is the first time periodic orbits can be created and modified (see Chapter 4) in surface flow.

5.2 Periodic Orbit Creation

In this section, I describe novel algorithms for creating periodic orbits in the plane and on surfaces. The input to the design algorithms consists of the desired type of the orbit (attracting or repelling) and a prescribed path, which is an oriented loop. Figure 5.1 shows an example path (left: blue loop). I then generate a sequence of evenly-spaced sample points on the loop (middle: green dots) and treat the tangent vectors at these points as constraints (middle: magenta arrows). Finally, I produce a vector field with a periodic orbit that closely matches the user input (right: red dashed lines). I use the dashed lines to represent the continuous periodic orbit so that it can be visually compared with the user-specified path. In vector field design, there are two approaches to initialize a vector field from the user specified constraints: basis vector fields [86, 101] and constrained optimization (Chapter 4 Section 4.2).

5.2.1 Attracting and Repelling Basis Vector Fields

An intuitive way to build a vector field that satisfies the constraints is to use basis vector fields [49] [86]. In this approach, every user-specified constraint is used to create a basis vector field defined in the plane. A vector field is then constructed as a weighted sum of these basis vector fields [86, 101]:

$$V(P) = \sum_i \omega_i(P) V_i(P) \tag{5.1}$$

where P is any position in the vector field, $V_i(P)$ is the i^{th} basis vector field, that refers to either a singular or a regular design element [101], and $\omega_i(P)$ is the weight for the i^{th} basis vector field. In the implementation, $\omega_i(P) = e^{-\|P-P_i\|^2}$ is used for the i^{th} basis vector field, where P_i is the center position of the i^{th} basis vector field.

This idea has been applied to creating wind forces to guide computer animation [95], to testing a vector field visualization technique [86], and to generating vector fields for non-photorealistic rendering and texture synthesis [101].

In theory, any vector field can be created by using regular elements. In practice, however, it often requires an excessive number of regular elements to generate certain vector field features. For example, at least three regular elements are needed to specify a source or a center. To produce a periodic orbit, regular elements must be specified not only along the prescribed path, but also near the orbit in order to enforce the type of the orbit (attracting or repelling). Given that the cost of summing basis vector fields is proportional to the number of design elements, we wish to reduce the number of basis vector fields while maintaining efficient control. This is achieved with the introduction of two new types of design elements: *attachment elements* and *separation elements*.

Before describing these elements, let us briefly review the concepts of attachment and separation points from Kenwright [36]. Recall that given a vector field V and a point $\mathbf{p}_0$ in the plane, I consider the following two values: $e_1 \times u$ and $e_2 \times u$, where u is the vector value at $\mathbf{p}_0$ and e_1 and e_2 are the major and minor eigenvectors of the Jacobian. $\mathbf{p_0}$ is an *attachment point* if $e_1 \times u = 0$, and a *separation point* if $e_2 \times u = 0$. An attachment line consists of attachment points.

Geometrically, such a line attract nearby flow. A separation line can be defined in a similar fashion except that nearby flow is repelled from the curve. Ideally, an attachment element will result in a basis vector field that has an attachment line as illustrated in Figure 5.2 (middle). The following formula describes an attachment element that has a desired vector value of $(1, 0)$ at (x_0, y_0).

$$V(x, y) = B(x, y) \begin{pmatrix} 1 \\ c(y - y_0) \end{pmatrix} \tag{5.2}$$

where $B(x, y) = e^{-((x-x_0)^2+(y-y_0)^2)}$ is the blending function for the element and $c < 0$ is a parameter that describes the speed at which the flow leaves the line $y = y_0$. The larger $|c|$ is, the more quickly the vectors near the attachment line point towards it. Notice the basis field contains an attachment line at $y = y_0$. Formula 5.2 can also be used to specify a separation element ($c > 0$) and a regular element ($c = 0$). When the vector value is $(\cos\theta_0, \sin\theta_0)$ for some constant θ_0 (θ_0 denotes the angle between the vector and $x-$axis), the formula has the following form:

$$V(x, y) = B(x, y) \left(\begin{pmatrix} \cos\theta_0 \\ \sin\theta_0 \end{pmatrix} + cP(x, y) \begin{pmatrix} -\sin\theta_0 \\ \cos\theta_0 \end{pmatrix} \right) \tag{5.3}$$

where $P(x, y) = -\sin\theta_0(x - x_0) + \cos\theta_0(y - y_0)$ is the signed distance of a point (x, y) to the line that is specified by the location and direction of the design element. Figure 5.2 compares two basis vector fields generated from a regular element (left) and an attachment element (middle). The right image shows an attracting periodic orbit created from four attachment elements. The ideas of attachment

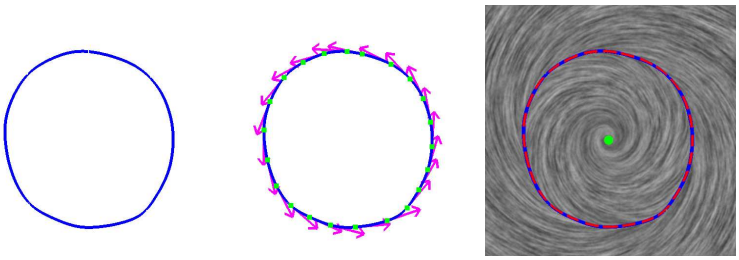


Figure 5.1: Given an oriented loop (left), the design system produces a sequence of sample points (middle: dots) and evaluates tangent vectors at those locations (middle: arrows). I then compute a vector field that contains a periodic orbit (right: red dashed lines) by generating constraints based on these vector values. Notice that the periodic orbit matches closely the user-specified loop.

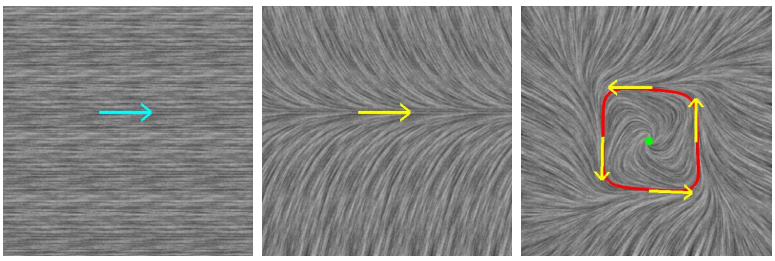


Figure 5.2: This figure compares the basis vector field corresponding to a regular element (left) and an attachment element (middle). The periodic orbit in the right was created by using four attachment elements.

and separation has been used again in the periodic orbit extraction algorithm (Section 3.1).

Vector field design using basis vector fields is intuitive and generates smooth results. However, the cost associated with this approach is proportional to the number of basis vector fields. To specify a relative large periodic orbit with high curvature often requires hundreds of attachment or separation elements, which makes interactive design a difficult task. The problem is magnified on surfaces on 3D as every basis vector field requires a global surface parameterization that is

specific to the underlying design element [101]. Constructing hundreds of surface parameterizations makes it impractical to create a periodic orbit interactively. Next, I describe a different strategy that is based on constrained optimization.

5.2.2 Constrained Optimization for Periodic Orbit Creation

Given a user-specified oriented loop γ and the desired type of the periodic orbit, the design system performs the following operations to create a periodic orbit closely matching the input.

First, I identify a region R_γ, which is a set of triangles that enclose γ. Next, I assign vector values to the vertices of R_γ according to the desired type, path, and orientation of the periodic orbit. Finally, the design system performs a constrained optimization to compute vector values for vertices outside R_γ, i.e., the free vertices in the domain. The quality of the resulting periodic orbit depends on the choice of R_γ and the vector assignment on the boundary of R_γ.

The attachment and separation elements can be used to obtain vector values on R_γ. Basically, each line segment on the loop γ is used to infer a design element. I then compute vector values at the vertices of R_γ using the basis vector fields corresponding to these elements. Note when R_γ is chosen to be the whole domain, this technique becomes the basis vector field method mentioned earlier, which is computationally expensive. In practice, I choose R_γ be the smallest triangle strip containing γ. This greatly reduces the amount of computation that is associated with basis vector fields. In addition, it seems to produce reasonable results both on

the plane and surfaces. I further speed up the process by only evaluating a basis field at the three vertices of the triangle that contains the corresponding element. When a vertex is shared by more than one triangle in R_γ, I simply take the average of the vector values computed from each incident triangle. Fig 5.1 shows that this method tends to produce a periodic orbit (right: dashed red loop) that matches the user-specified loop (right: blue loop). To obtain smoother results, a larger R_γ can be constructed.

I have also extended a similar framework to create fixed points on surfaces. Every fixed point results in three constraints on the vertices that contains the desired fixed point. Vector values elsewhere in the mesh are obtained through constrained optimization. This framework avoids the need to construct a surface parameterization for each basis [101] and makes it possible to interactively create periodic orbits on surfaces in 3D. Figure 1.2 shows a number of vector fields that were created using the design system. Although it works very well in practice, it should be pointed out that the proposed approach does not guarantee to create a periodic orbit according to user input.

Chapter 6 – Morse Decompositions of Vector Fields

In previous chapters I have defined the topology of two-dimensional vector fields as fixed points and periodic orbits as well as the separatrices that connect them [8,27]. This leads to a graph representation of the vector field which is referred to as Entity Connection Graph, or ECG. The features and connectivity information in ECG as well as conventional topological skeleton of vector fields are defined according to trajectory (Section 2.2). I refer to this vector field topology as *trajectory-based* topology. However, analysis and visualization of vector field topology based on individual trajectories can raise questions with respect to interpretation as the discrete nature of fluid flow data poses several challenges. First, data samples are only given at discrete locations, such as cell vertices or cell centers. Interpolation schemes are then used to reconstruct the vector field between the given samples. Second, the given data samples themselves are numerical approximations, e.g., approximate solutions to a set of partial differential equations. Third, the given flow data are often only a linear approximation of the underlying dynamics. Finally, the visualization algorithms themselves, e.g., streamline integrators, have a certain amount of inherent error associated with them. In short, how can we be sure that what we see is authentic when extracting and visualizing the topological skeleton of the flow field? Could the error inherent to multiple numerical approximations produce misleading information? Figure 6.1 provides examples in which

proper interpretation can be difficult when performing analysis based on individual trajectories.

Figure 6.1(a) shows an analytical vector field which contains pitchfork bifurcation [26]. The results shown in the two columns of (a) are obtained by computing sample vector values using two different meshes: (left) a regular triangulated mesh with 6144 triangles, and (right) a triangulated mesh with 1000 triangles. Notice that using different meshes leads to different ECGs (third row of Figure 6.1(a)). Figure 6.1(b) demonstrates a saddle-saddle connection bifurcation [26]. The images to the left of Figure 6.1(b) show the original flow, while the images to the right show the flow that was obtained from the original one after introducing a small amount of perturbation (I have randomly perturbed the vector direction at each vertex by an angle between 0° and 1°.) Notice that ECGs (third row of Figure 6.1(b)) are sensitive to noise. Figure 6.1(c) provides a case of Hopf-bifurcation [40]. The image to the left of Figure 6.1(c) (second row) shows the resulting topology using an adaptive fourth-order Runge-Kutta integration, while the image to the right illustrates the topology of the same vector field using a second-order Runge-Kutta integration [3] [55]. This clearly demonstrates that the ECGs rely on the employed numerical scheme. (The ECGs in all the example flows are computed using the algorithms proposed in Chapter 3.) These observations motivate the study of a more reliable way of defining and extracting vector field topology than the existing techniques. It should be noted that addressing such uncertainty in visualization was identified as one of the most important future challenges by Johnson [32].

In order to address this important challenge I present a rather different ap-

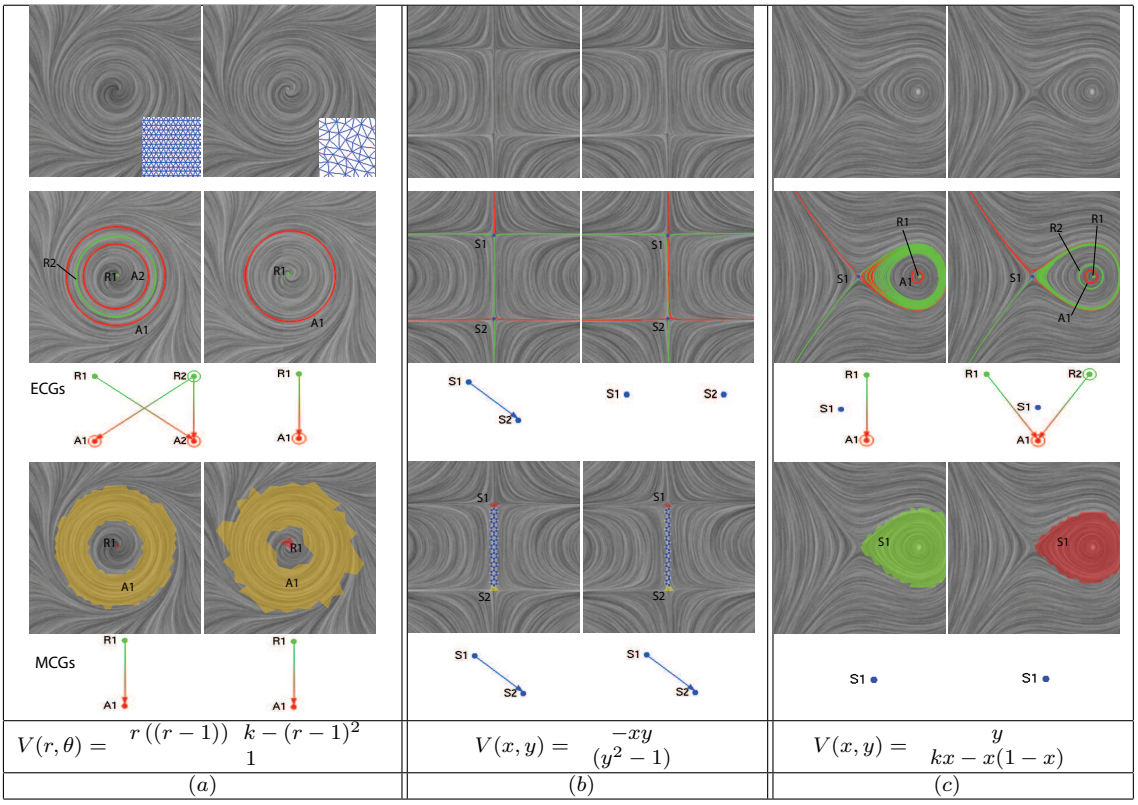


Figure 6.1: Examples of the instability of individual trajectory-based topological analysis of vector field (i.e. ECGs) due to the choice of discretization scheme (a), noise (b), and the error from numerical integration scheme (c). (a) shows a vector field containing pitchfork bifurcation ($k = 0.05$). It illustrates the deviated ECGs obtained under two different discretization schemes. The vector field shown in (b) is an example flow having saddle-saddle connection. The two ECGs are computed based on the original result flow and its perturbation version. It illustrates the possible influence of the unexpected noise in the data. (c) uses a vector field with Hopf-bifurcation ($k = 0.0025$) to illustrate that ECGs can be different using different numerical schemes. The image to the left shows the resulting topology using the adaptive fourth-order Runge-Kutta integration, while the image to the right shows the topology of the same flow using a second-order Runge-Kutta integration. The two bottom rows provide the results of Morse decompositions and the associated MCGs of these fields using the idea of τ-maps proposed in this paper. The τ's for these fields are 40, 20 and 80, respectively. Note that for all the examples shown here, the MCGs are stable.

proach 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) which has been introduced in Chapter 2. The nodes in this graph are Morse sets, whose corresponding polygonal regions in the phase space are Morse neighborhoods. All the recurrent dynamics is contained in the Morse neighborhoods. The edges in an MCG indicate how the flow moves from one Morse neighborhood 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 [35]. In other words, a well defined error, $\epsilon > 0$, can be bounded and included into the map of the flow domain. I demonstrate the stability of MCGs in Figure 6.1 (the last two rows). In previous discussion, MCG is used as the pre-step for ECG computation. From this chapter, I will emphasize the MCG as a topology representation of particular interest.

In this chapter, I start with the review of Morse decomposition. Different from the discussion of the same subject in Section 2.4, I will particularly focus on its stability and computation rather than its relation to ECG. Recall that to perform Morse decomposition, i.e., compute MCGs, I first construct another directed graph by considering the behaviors of the vector field along edges of the triangles, which is referred to as the geometry-based method (Section 2.4.2). I refer to the process of encoding the flow dynamics into a directed graph as *flow combinatorialization*. Because the triangulation is not adapted to the vector field, this can result in

coarse Morse sets (Figure 6.2(b)). In this work I exploit a temporal discretization, which I refer to as a τ-*map*, 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 (Figure 6.2(c) (d)) than the geometry-based method. The key challenges with the τ-map guided approach are choosing an appropriate temporal discretization of the flow and constructing a high-quality flow combinatorialization, which is the discrete outer approximation of a τ-map. In the implementations, I will compute it as a directed graph, denoted by $\mathcal{F}_\tau$ under a time τ. From it I extract the Morse sets.

6.1 Revisiting Morse Decomposition and Morse Connection Graphs (MCGs)

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

$$\{f_d(v_i) \mid v_i \text{ a vertex of } X\} \tag{6.1}$$

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,

$$||f(v_i) - f_d(v_i)|| \leq \epsilon. \tag{6.2}$$

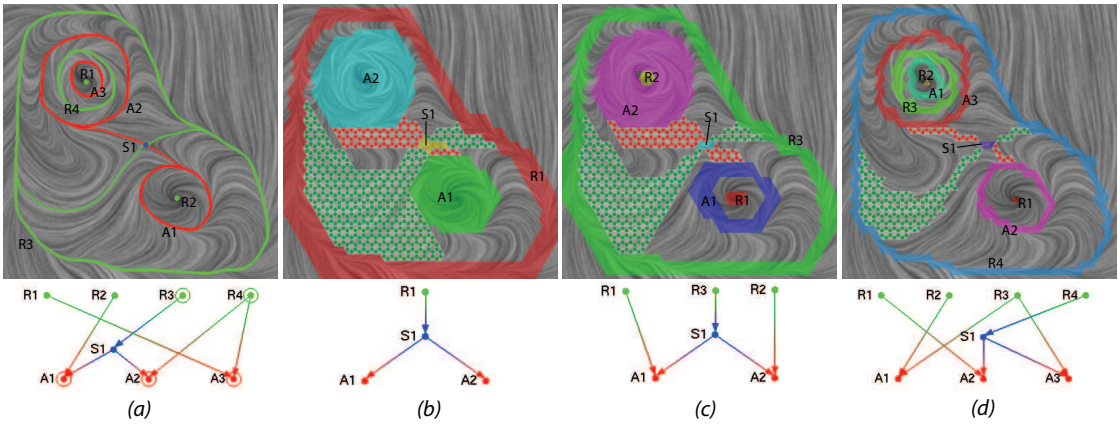


Figure 6.2: This figure shows the various analysis results of an experimental field using ECG and MCGs, respectively: (a) ECG, (b) MCG (geometry-based method), (c) MCG ($\tau = 6$), (d) MCG ($\tau = 24$). The computation time for (b-d) is 0.14s, 1.78s, and 4.31s, respectively. Observe that the larger the τ used, the better (closer to optimal) the Morse decompositions, but the time for computing the Morse decomposition increases accordingly. The coloring scheme of the MCG is described in Figure 6.1. Notice that the graphs shown in (a) and (d) are essentially the same although they are labeled differently. The execution time was measured on a 3.0 GHz PC with 1.0 GB RAM. The color-dotted regions indicate the connections between a saddle Morse set to another Morse set: source (green), sink (red), and saddle (blue).

In addition, since we are only given the data (Eq. 6.1) I 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. 6.2) are global, that is $||f(x) - f_d(x)|| \leq \epsilon$ 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 parameterized by some abstract parameter space Λ. I assume that for each $\lambda \in \Lambda$, the vector field $\dot{x} = F(x, \lambda)$ gives rise to a flow $\varphi_\lambda : \mathbb{R} \times X \to X$.

In this setting I assume that there exist parameter values $\lambda_0, \lambda_1 \in \Lambda$ such that $f(x) = F(x, \lambda_0)$ and $f_d(x) = F(x, \lambda_1)$. Bifurcation theory tells us that even if $\lambda_0 \approx \lambda_1$, the orbits, i.e. fixed points, periodic orbits, separatrices, of φ_{λ_0} and φ_{λ_1} need not agree [26]. 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.

As defined before, a *Morse decomposition* of X for a flow φ_λ is a finite collection of disjoint compact invariant sets, called *Morse sets* [35]

$$\mathbf{M}(X, \varphi) := \{M_\lambda(p) \mid p \in (\mathcal{P}_\lambda, \succ_\lambda)\},$$

where $\succ_\lambda$ is a strict partial order on the indexing set $\mathcal{P}_\lambda$, such that for every $x \in X \setminus \cup_{p \in \mathcal{P}_\lambda} M_\lambda(p)$ there exist indices $p \succ_\lambda q$ such that

$$\omega(x) \subset M_\lambda(q) \quad \text{and} \quad \alpha(x) \subset M_\lambda(p).$$

It has proven that the Morse sets enclose recurrent dynamics (e.g. fixed points, periodic orbits, chaotic dynamics) [35]. 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 X.

Observe that since $\mathcal{P}_\lambda$ is a strictly partially ordered set a Morse decomposition can be represented as an acyclic directed graph. The nodes of the graph correspond to the Morse sets and the edges of the graph are the minimal order relations which through transitivity generate $\succ_\lambda$. This graph is called the Morse connection graph and denoted by MCG_λ [8] (Figures 6.1, 6.2 bottom rows). Moreover, without worrying about the potential noise and numerical errors, an ECG indicates the finest MCG when the vector field has a finite number of fixed points and periodic orbits, all of which have an isolating neighborhood of their own [8]. Though, there may not exist a finest Morse decomposition. Consider the flow generated by the differential equation $x' = x^2 \sin(1/x)$. It has an infinite number of isolated fixed points and hence there is no finest Morse decomposition (remember that there can only be a finite number of Morse sets). Any Morse decomposition of it can be refined further.

6.2 MCG Construction

I now summarize the pipeline of constructing an MCG given the vector field V defined on a triangulated surface X.

First, I perform flow combinatorialization. That is, I encode the flow dynamics

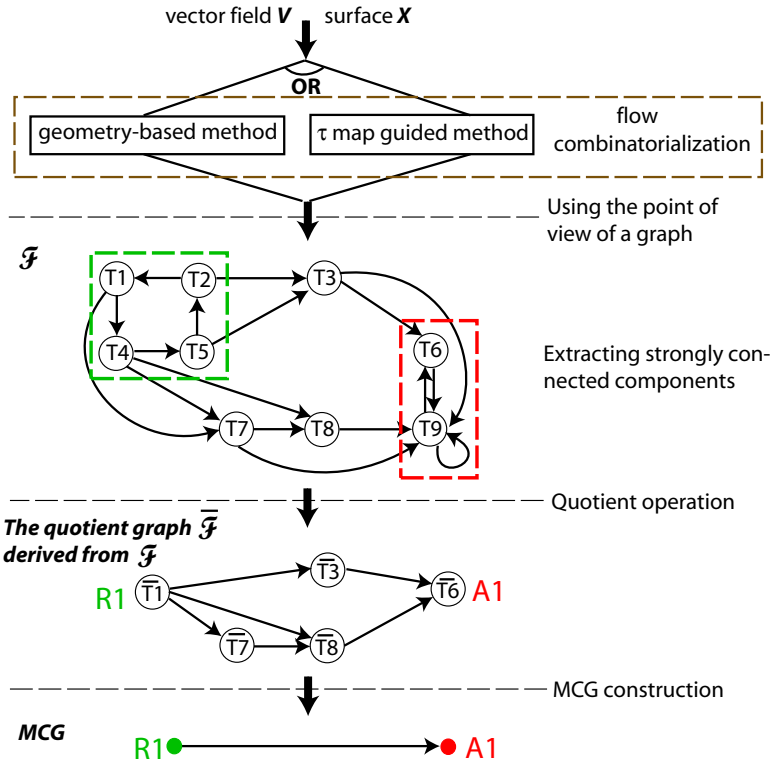


Figure 6.3: This figure illustrates the pipeline of MCG construction. I first compute $\mathcal{F}$ (top) based on the underlying flow. The edges in the multi-valued map demonstrate the mapping relations of the polygons. Based on the $\mathcal{F}$, I extract the strongly connected components, which represent either the Morse sets (middle-top, inside colored boxes) or intermediate nodes that describe gradient-like behaviors (middle-top, T3, T7, T8). I then collapse each strongly connected component of the $\mathcal{F}$ into a single node to obtain a quotient graph $\overline{\mathcal{F}}$. Note that the nodes in this graph correspond to either Morse sets or the polygonal regions of gradient-like flow behaviors (i.e. trivial Conley index). Finally, the MCG (the bottom graph) is obtained by collapsing nodes with trivial Conley index and removing redundant edges.

into a directed graph, denoted by $\mathcal{F}$, whose nodes represent the elements (e.g. triangles) of the underlying mesh and edges indicate the flow dynamics, i.e., an edge from triangle T_i to triangle T_j indicates that $\varphi(T_i) \bigcap T_j \neq \emptyset$ (Figure 6.4, left). The details of this will be described in Section 6.3.

Second, I find the strongly connected components of the directed graph $\mathcal{F}$, which gives rise to the Morse neighborhoods 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 [35] (Figure 6.3, middle-top).

Third, I compute a quotient graph $\overline{\mathcal{F}}$ from $\mathcal{F}$ by treating each strongly connected component of $\mathcal{F}$ as a node (Figure 6.3, middle-bottom). The nodes in this quotient graph $\overline{\mathcal{F}}$ include Morse sets (non-trivial Conley index) and the intermediate nodes corresponding to the polygonal regions with gradient-like flow behaviors (i.e. trivial Conley index). An edge $\overrightarrow{\overline{m}\overline{n}}$ in $\overline{\mathcal{F}}$ indicates that there is at least one edge $\overrightarrow{\overline{k}\overline{l}}$ in $\mathcal{F}$ such as $\overline{k} = \overline{m}$ and $\overline{l} = \overline{n}$.

Finally, I extract the MCG from $\overline{\mathcal{F}}$ by removing intermediate nodes from $\overline{\mathcal{F}}$ as illustrated in Figure 6.3 (the bottom graph). The algorithm for MCG construction can be found in [34].

To visualize the MCG, I classify the nodes of the MCG into three types (Section 2.3): *Source Morse sets*, R_i, are nodes whose Conley indices have non-zero second Betti number (i.e. $\beta_2 = 1$); *Sink Morse sets*, A_i, are nodes whose Conley indices have non-zero zeroth Betti number (i.e. $\beta_0 = 1$); *Saddle Morse sets*, S_i, are neither source Morse sets nor sink Morse sets. The R_i's are colored green,

the A_i's are colored red and the S_i's are colored blue. According to the partial order determined by the edges in the MCG, I 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. Figures 6.1, 6.2 and 7.5 display the MCGs of a number of analytical vector fields. Compared to the three-layer structure of the ECG, an 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 (Figure 6.1(b), Figure 7.5(b)). Note that finer classification of Morse sets, e.g., Saddle Morse sets, can be realized based on Conley index theory [45].

I wish to emphasize that some graphics applications may pursue the individual trajectory-based vector field topology without being concerned with the fact that the obtained ECGs may not be topologically rigorous, such as, the applications in texture synthesis [84,91] and fluid simulation [65]. For such applications, an ECG can still be extracted from Morse decomposition as an additional step [8].

6.3 Flow Combinatorialization Based on τ-maps

I now turn to the issue of flow combinatorialization, i.e., the process of generating the graph $\mathcal{F}$ based on a vector field V defined on a triangulated mesh X. Recall that the geometry-based approach is as follows: The vertices of the directed graph $\mathcal{F}$ correspond to the triangles of the mesh. The edges of $\mathcal{F}$ are obtained by considering the flow behavior across each edge of each triangle. An edge $T_i \rightarrow T_j$

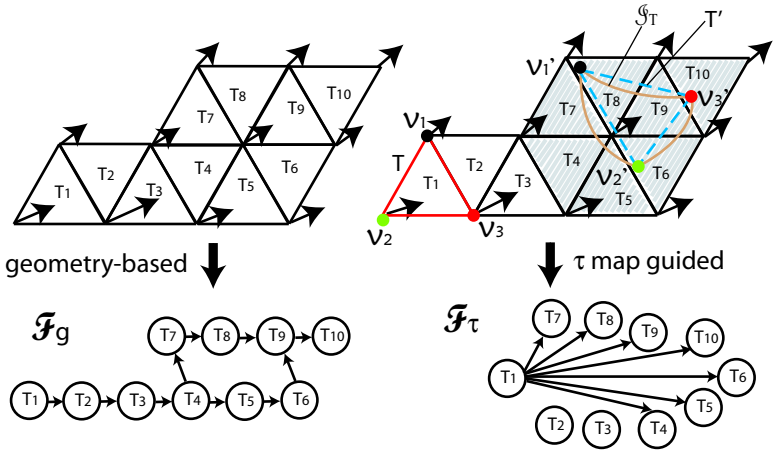


Figure 6.4: This figure compares two ways of performing flow combinatorialization: (left) geometry-based method, and (right) τ-maps. In the directed graphs, each node corresponds to a triangle of the mesh. The red triangle $T = T_1$ is the starting triangle, the light brown curved closure is the real image of T, the blue dashed triangle is the approximation of the real image.

in $\mathcal{F}$ indicates the flow can enter from T_i to T_j, where T_i and T_j are neighboring triangles (Figure 6.4, left). The resulting directed graph is denoted as $\mathcal{F}_g$. An MCG can then be obtained from $\mathcal{F}_g$ using the pipeline described in Section 6.2.

Since the mesh is not fitted to the flow, this approach is not guaranteed to obtain the correct dynamics of the flow. In the experiments, I have found that it often results in a rather coarse outer approximation of the underlying dynamics, i.e., Morse sets that contain multiple fixed points and periodic orbits (Figure 6.2(b)) or no structures at all (Figure 7.7 left column). This makes subsequent analysis and physical interpretation less effective. To obtain the Morse decomposition that are closer to optimal, I introduce the concept of τ-maps, which allows us to move from the continuous time of a flow to discrete time of a map. This leads to the following definition.

Definition 6.3.1 Let $\tau : X \to (0, \infty)$ be a continuous map. A *τ-time discretization* of the flow φ is a map $f_\tau : X \to X$ defined by

$$f_\tau(x) := \varphi(\tau(x), x).$$

I refer to this map as a τ-map. Thus, finding Morse decompositions for the flow φ is equivalent to finding Morse decompositions for f_τ.

The fact that X is a triangulated surface provides us with an appropriate discretization in space. Let $\mathcal{X}$ be the triangulation of X (i.e., a set of triangles). I will approximate f_τ using a *combinatorial multi-valued map* $\mathcal{F} : \mathcal{X} \rightrightarrows \mathcal{X}$, that is a map such that for each triangle $T \in \mathcal{X}$, its image is a set of triangles, i.e.

$\mathcal{F}(T) \subset \mathcal{X}$.

The correct notion of approximation is given by the following definition. Consider $f_\tau : X \to X$. The combinatorial multi-valued map $\mathcal{F} : \mathcal{X} \rightrightarrows \mathcal{X}$ is an *outer approximation* of f_τ if

$$f_\tau(T) \subset \text{int}\left(|\mathcal{F}(T)|\right)$$

for every $T \in \mathcal{X}$ where $|\mathcal{F}(T)| := \cup_{R \in \mathcal{F}(T)} R$, *int* denotes the interior. As an example, I refer readers to Figure 6.4 (right). In this example, I assume that the true image of the triangle $T = T_1$ is $\mathcal{I}_T$. It is obtained by advecting T according to the underlying flow over a time τ. According to the definition, the outer approximation of $\mathcal{I}_T$ is the set of triangles $T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$. Mathematically, we say that T has been mapped to *multiple* triangles of the same mesh by a function (or a map) f_τ that is determined by the underlying flow under a certain time τ.

From the point of view of computation it is useful to view $\mathcal{F}$ as a directed graph, which I denote as $\mathcal{F}_\tau$. (Figure 6.4, right). Similar to $\mathcal{F}_g$, the vertices, T_i, of an $\mathcal{F}_\tau$ are the triangles of the underlying mesh and the edges indicate the outer approximation of the images of the triangles over time τ. For instance, an edge $T_i \to T_j$ indicates that the image of the triangle T_i over time τ will intersect with the triangle T_j (Figure 6.4, right).

Observe that the definition of an outer approximation requires a lower bound on the set of triangles in $\mathcal{F}(T)$, but not an upper bound. In general larger images of $\mathcal{F}$ are easier to compute. For example, one can obtain an outer approximation, by declaring $\mathcal{F}(T) = \mathcal{X}$ for all $T \in \mathcal{X}$. However, the larger the image the poorer the

approximation of the dynamical system of interest, f_τ. I discuss how to compute an $\mathcal{F}_\tau$ in Chapter 7.

6.4 The Stability of MCGs

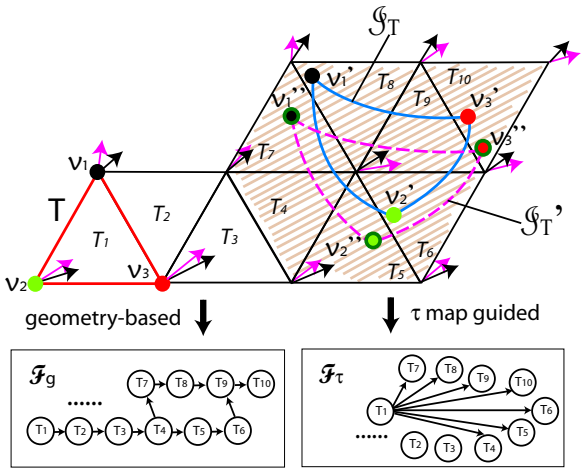


Figure 6.5: This figure illustrates that using outer approximation, Morse decomposition is stable under certain error bound ϵ. The original image of triangle T is $\mathcal{I}_T$ (region inside the blue curve), and the outer approximation of that is the set of shaded triangles $T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$. After a random perturbation (shown by magenta arrows) of the original field (shown by black arrows), I recalculate the image $\mathcal{I}'_T$ of triangle T, which is shown as the magenta curved closure. Although it is different from $\mathcal{I}_T$, the outer approximation consists of the same set of the triangles. Therefore, the corresponding portion of the direction graph $\mathcal{F}_\tau$ remains the same. Hence, we say that Morse decomposition is stable under an error bound ϵ, which here is the maximal allowed perturbation that will not change the outer approximation of the image of each triangle.

The definition of an outer approximation and the fact that the triangles in the strongly connected components of $\mathcal{F}$ form isolating neighborhoods (Chapter 2) for

the Morse sets demonstrate why the MCG remains constant under small perturbations of the vector field (Figure 6.1(b)). 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 $|\mathcal{F}(T)|$. Thus, this property will also hold for any sufficiently small perturbation of f_τ, which means that given a multi-valued map for f_τ we have the same $\mathcal{F}_\tau$ for any sufficiently small perturbation of f_τ. Figure 6.5 provides an illustrative example to explain 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 $\mathcal{I}_T$ 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 forms the outer approximation of $\mathcal{I}_T$. When I artificially introduce a random perturbation for each vector value (shown as magenta arrows) and advect the triangle T under the new flow, I obtain a new image $\mathcal{I}'_T$ of it (shown as magenta dashed curved closure). If we bound the perturbation of each vector to guarantee that the new obtained image $\mathcal{I}'_T$ will intersect the same set of the triangles as the $\mathcal{I}_T$ obtained under the original vector field, we will obtain the same outer approximation of the image of T. Hence, the corresponding portion of the directed graph $\mathcal{F}_\tau$ will not change. The MCG is consequently stable. In other words, the outer approximation provides more tolerance for error in the given data. I also point out that the MCGs obtained using the geometry-based method are also stable. Consider the example shown in Figure 6.5. Note that the flow behavior across each edge of the mesh does not change after a smaller perturbation, neither does the

corresponding portion of $\mathcal{F}_g$. Therefore, the MCG remains the same. On the other hand, in this setting this need not be the case for any particular trajectory such as a periodic orbit or even a fixed point. That is, a particular trajectory may be changed after any perturbation. Of course, we can go one step further and insist that an ϵ-neighborhood of $f_\tau(T)$ be contained in $|\mathcal{F}(T)|$. We will in general get a coarser $\mathcal{F}_\tau$, but the resulting Morse decompositions will be valid for any vector field whose τ-map lies within ϵ of f_τ.

After applying the idea of τ-map based Morse decomposition to the analytical field shown in Figure 6.2, we obtain a finer Morse decomposition (Figure 6.2(d)). The colored regions there indicate the isolating neighborhoods of the Morse sets. Different color regions indicate different Morse sets. The flow-like texture regions without color indicate the regions of gradient-like flow (Section 6.1). The color-dotted regions indicate the connections between a saddle Morse set to another Morse set: source (green), sink (red), and saddle (blue).

In the following chapter, I will present efficient algorithms to compute the flow combinatorialization based on the idea of $\tau-$map.

Chapter 7 – Flow Combinatorialization Using the Idea of τ-map

As we have seen in Chapter 6, when employing the idea of τ-maps, computing the correct flow combinatorialization $\mathcal{F}_\tau$ is the most crucial step in the pipeline of the Morse decomposition. To obtain an accurate $\mathcal{F}_\tau$, it is essential to compute the accurate (sufficient) outer approximation of the image of each triangle of the given mesh over τ time and obtain the directed edges of $\mathcal{F}_\tau$ accordingly.

To approach that, a rigorous method is applicable to any τ-time discretization and produces a rigorous outer approximation assuming that a bound ϵ on the errors in the underlying vector field is known. More specifically, given a triangle T one covers it with squares of size ϵ. For each square S define $\tau_*(S) = \min\{\tau(x) \mid x \in S\}$ and $\tau^*(S) = \max\{\tau(x) \mid x \in S\}$. Using rigorous enclosure techniques [1, 46] one obtains an outer enclosure $\mathcal{I}_S$ of the true image of the square S integrated forward for all times $\tau_*(S) \leq t \leq \tau^*(S)$. Then $\mathcal{I}_T = \cup\mathcal{I}_S$, where the union is taken over all squares S, is an outer approximation for $f_\tau(T)$.

This method is computationally costly. First, the number of squares needed to cover the triangle T is of order ϵ^{-2}, which for small ϵ is large. Second, due to the Gronwall inequality [26] the size of the image of $\mathcal{I}_S$ grows exponentially as a function of the integration time. Thus, to get tight outer approximations one must choose small ϵ. On the other hand, variants of this method have been used to obtain rigorous computer assisted proofs in dynamics [1].

In this chapter, I describe a more practical algorithm for performing flow combinatorialization based on τ-maps. This method can obtain enough information of the image of each triangle through the tracing of vertices and the heuristically chosen samples on the edges of the given mesh without having to compute the outer approximation explicitly. While this method is not rigorous in theory, it works for all the applications I have applied it to in practice. The presented method is based on the following observation: the image of a connected object under a continuous map is still connected. More specifically, the image of a triangle under a τ-map which is a continuous map is either a connected region, a simple curve or a point. And, the image of a line segment (e.g. an edge of the mesh) is a simple curve or a point. I now discuss this method in detail as follows [9].

7.1 Efficient Outer Approximation Computation

Let us start with the study of some possible scenarios of the outer approximation of the image of a triangle T. Assume that T consists of three vertices v_1, v_2, v_3 and three edges e_1, e_2, e_3, where $e_1 = (v_2, v_3)$, $e_2 = (v_3, v_1)$ and $e_3 = (v_1, v_2)$. Considering the definition of an outer approximation in Section 6.3, I let $I(T)$ represent the outer approximation of T obtained using certain numerical integration (such as, Runge-Kutta method). Similarly, let $I(v_i)$ represent the outer approximations of the images of the three vertices and $I(e_i)$ represent the outer approximations of the images of the three edges of T, respectively. Typically, $I(v_i)$ is a single triangle that contains the image of v_i if the integration error is smaller than the diameter of

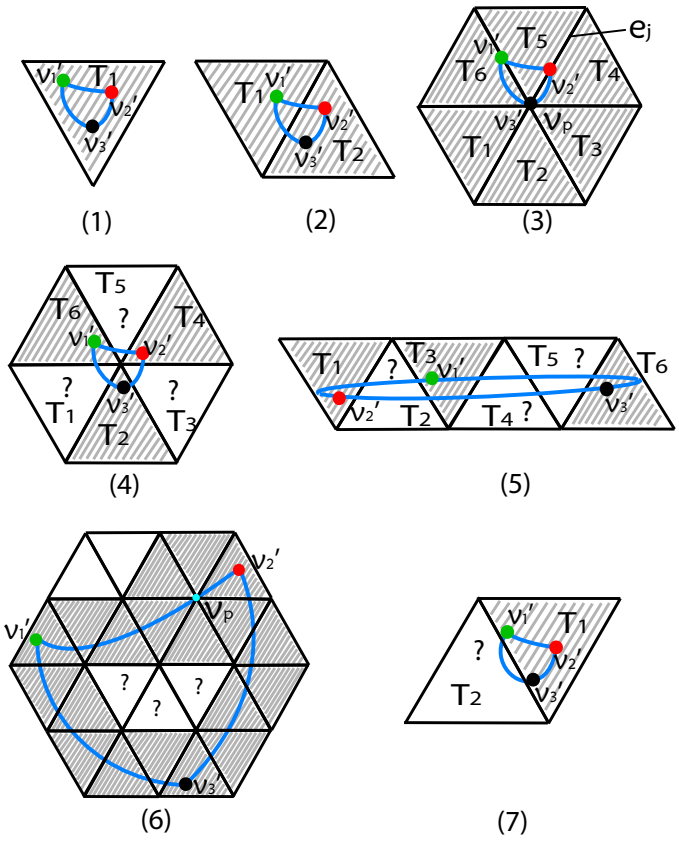


Figure 7.1: Some possible cases of the image of a triangle under a flow.

the triangle. To guarantee obtaining a sufficient outer approximation, if the image of v_i is located at a vertex v_p, I set $I(v_i)$ to be the one-ring neighborhood of v_p (Figure 7.1, cases (3) and (6)). If the image of v_i is located on an edge e_j, I set $I(v_i)$ to be the two triangles that have e_j as one common edge (Figure 7.1, case (3)).

Cases (1), (2) and (3) of Figure 7.1 show the first scenario. In this scenario, $I(T) \subseteq \cup_{i=1}^{3} I(v_i)$. That is, we only need to trace from the three vertices of T, the union of the outer approximations of them will give rise to the outer approximation of T.

Cases (4) and (5) of Figure 7.1 provide examples of Scenario 2. In this scenario, $I(T) \subseteq \cup_{i=1}^{3} I(e_i)$. Therefore, the union of $I(v_i)$ will not provide us a sufficient outer approximation (for instance, triangles T_1, T_3, T_5 of case (4) in Figure 7.1 will be missing), but the union of $I(e_i)$ will. This requires us to keep track of the image of an edge.

7.1.1 Adaptive Edge Sampling

Since we are interested in the outer approximation of an edge instead of the exact image of it, the connected triangle strip that contains the image of the edge is sufficient. The connected triangle strip I refer to here is a triangle strip in which a pair of neighboring triangles share a common edge due to the aforementioned observation of the image of an edge under a continous map ($\tau-$ map here)(for example, Figure 7.2). I introduce the idea of adaptive edge sampling (Algorithm

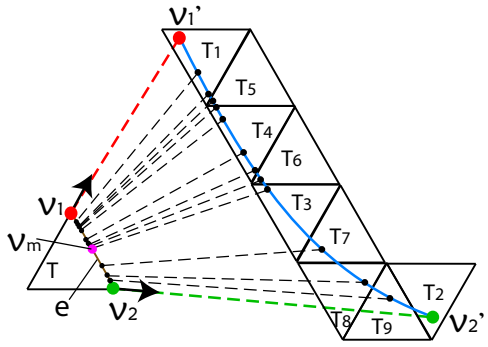


Figure 7.2: This figure provides the notion of adaptive sampling on an edge $e(v_1v_2)$ (right). T is the original triangle. The image of edge v_1v_2 is $v_1'v_2'$. The dash lines show the mapping of the samples to the points on the image. The indexing of the triangles (right) indicate the order of computation.

Algorithm 1: Adaptive sampling on an edge

Routine: adaptive_edge_sampling(v_1, v_2, T_1, *original_T*, *neighbor_T*, V, X, τ, L)
Input: v_1,v_2: two vertices;
T_1, *original_T*, *neighbor_T*: triangles
V: vector field; X: surface; L: recursion level;
τ: user specified integral time
Output: the edges in the graph $\mathcal{F}_\tau$ related to the two triangles *original_T* and *neighbor_T*
Global variables: $\mathcal{F}_\tau$: the directed graph
Local variables: T_2: a triangle; s: a vertex
Begin
$L \leftarrow L + 1$;
if ($L >$ maximum recursion level || $||v_1 - v_2||_2 <$ minimum distance)
$v_1 \leftarrow v_2$; $T_1 \leftarrow T_2$;
new_edge(*original_T*, T_1, $\mathcal{F}_\tau$);
new_edge(*neighbor_T*, T_1, $\mathcal{F}_\tau$);
return;
$T_2 \leftarrow$trace(v_2, τ);
if($T_1 == T_2$ || share_common_edge(T_1, T_2))
$v_1 \leftarrow v_2$; $T_1 \leftarrow T_2$;
new_edge(*original_T*, T_1, $\mathcal{F}_\tau$);
new_edge(*neighbor_T*, T_1, $\mathcal{F}_\tau$);
return;
else
$v_1 \leftarrow v_2$; $s \leftarrow v_2$;
$v_2 \leftarrow (v_1 + v_2)/2$;
call adaptive_edge_sampling(v_1, v_2, T_1, *original_T*, *neighbor_T*, V, X, τ, L);
$v_2 \leftarrow s$;
call adaptive_edge_sampling(v_1, v_2, T_1, *original_T*, *neighbor_T*, V, X, τ, L);
return;
End

1). The basic idea is that I first trace from the two vertices of an edge $e(v_1, v_2)$ (Figure 7.2, right). If the two triangles T_1 and T_2 containing the two advected vertices are the same triangle or they share a common edge, then we do not process e further. Otherwise, more samples are then used until we obtain a connected triangle strip containing the image of e. To compute new samples, I make use of a binary search along the edge e . In detail, if the two triangles containing the images of the two vertices v_1, v_2 are neither the same nor neighbors, I then trace from the middle point v_m of the line segment (v_1, v_2) and determine whether the triangle T_3 that contains the image of v_m and the two triangles T_1 and T_2 form a connected triangle strip or not. If they are not, assume that among them T_1 and T_3 are not neighbors, It means that we need more samples on the line segment (v_1, v_m) to obtain the connected triangle strip between T_1 and T_3. Therefore, I compute the middle point of the line segment (v_1, v_m) and trace from it to obtain the triangle containing the image of it. Repeat this process until a connected triangle strip is found. Figure 7.2 demonstrates the idea of this algorithm. The indexing of the triangles indicate the order of computation. I wish to point out that due to a discrete representation, there is no guarantee of finding a continuous map under a highly divergent flow with a large τ, even though we sample densely along the edges. However, I have not experienced this problem in practice.

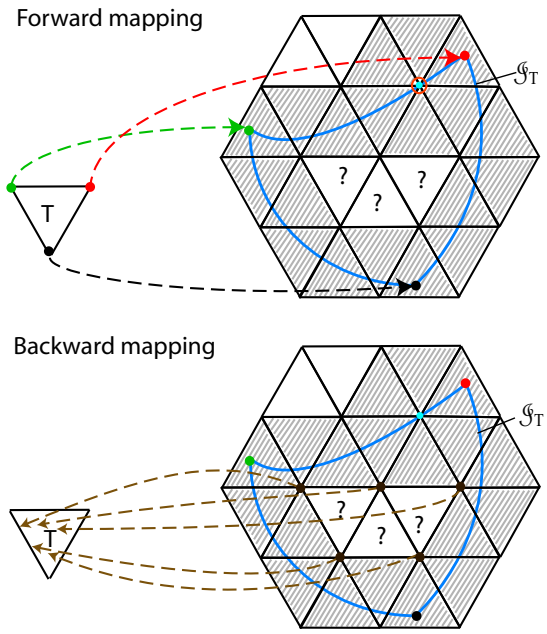


Figure 7.3: A general example of the image of a triangle under a flow showing the scenario of case (6) in Figure 7.1. Through this example, I introduce the idea of backward mapping. The image in the top row illustrates the forward mapping. The red, green and black dashed curves indicate the forward mapping. Using adaptive edge sampling, we can find the connected triangle strip (the shaded region) that contains the image of the boundary of the triangle T. The bottom row image illustrates the idea of backward mapping. The interior vertices have been traced over the same time τ based on the inversed flow. The images of them will fall in the triangle T. The brown dashed curves indicate the backward mapping. Thus, we can obtain the remaining edges in the directed graph. Note that the boundary of the forward image $\mathcal{I}_T$ of T intersects with one vertex (highlighted by an orange circle). To obtain a sufficient outer approximation, I add the one-ring neighborhood of the vertex to the outer approximation.

7.1.2 Backward Mapping

Using the adaptive edge sampling scheme, I successfully compute the outer approximation of scenario 2. But the method fails in case (6) of Figure 7.1, which is an example of scenario 3. In this scenario, $I(T) \supset \cup_{i=1}^{3} I(e_i)$. Therefore, keeping track of the images of the three edges is not sufficient. More specificially, consider the image $\mathcal{I}_T$ of a triangle T under a flow V over time τ (Figure 7.3, top). In this case, we can find all the triangles that contain the images of the three edges of T using the adaptive edge sampling algorithm. But we are not able to find the interior triangles intersecting with $\mathcal{I}_T$. I observe that any sample inside T will be mapped to the image $\mathcal{I}_T$, and any sample inside $\mathcal{I}_T$ should be able to be mapped back to the interior of T as well (Figure 7.3, bottom row). That is, if we sample any point inside each inner triangle, and trace the sample point with respect to the inverse flow $-V$ over the same time τ, the image of it should fall in T. These observations motivate me to introduce the backward mapping as the complement of forward mapping when computing the outer approximation of the image of a triangle. Figure 7.3 (bottom row) illustrates the idea of the backward mapping. For the updating of the graph $\mathcal{F}$, if we trace backward from any sample of a triangle T_i over time τ, and its image falls in triangle T_j, we add an edge $T_j \rightarrow T_i$ to $\mathcal{F}$.

With the assist of backward tracing combined with the adaptive samping scheme, we now can compute a sufficient outer approximation for case (6) in Figure 7.1. Furthermore, more difficult case could be handled as well. Consider case

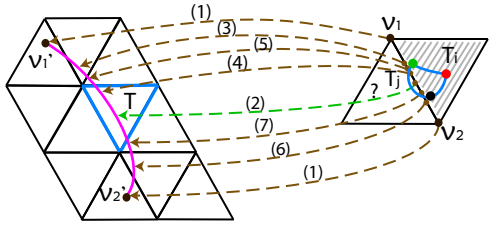


Figure 7.4: This figure describes how the backward mapping and the adaptive edge sampling help to find the complete edges of the directed graph under a highly stretched flow. The edge (v_1v_2) has been sampled to obtain the continuous triangle strip that contains the image of it using Algorithm 1. The brown dashed curves illustrate the backward tracing along $-V$. (1)–(7) indicate the sampling and tracing order. Note that step (2) gives rise to the edge $T \rightarrow T_j$ that is missed in the case (7) of Figure 7.1.

(7) in Figure 7.1, $\mathcal{I}_T$ intersects with two triangles. Therefore, the outer approximation should include these two triangles, even though the images of the three vertices fall in the same triangle. Using both adaptive edge sampling and backward mapping, I can compute the outer approximation of this case correctly as follows. I first perform forward tracing, which will eventually generate an edge from triangle T to T_i. When I perform backward tracing, I first trace the two vertices of edge (v_1v_2) (step (1) of Figure 7.4) of the edge and determine whether the two triangles containing the images of the vertices of the edge are not neighbors. In here, they are not. I then choose the middle point of the edge and trace from it over time τ. It ends at triangle T. Therefore, I obtain the edge from T to T_j, since the edge (v_1, v_2) is shared by both T_i and T_j.

Algorithm 2: An efficient outer approximation computation

Routine: construct_multivaluemap(V, X, τ, L)
Input: V: vector field; X: surface; τ: integral time
L: maximum recursion level
Output: $\mathcal{F}_\tau$: the completed graph
Local variables: T: current triangle; e : current edge;
N_T: the triangle sharing the edge e with T;
v_1,v_2: the two vertices of e

```
Begin
for each vertex v of X
   T ← trace_forward(v, τ);
   new_edges(one-ring of v, T);
for each vertex v of X
   T ← trace_backward(v, τ);
   new_edges(T, one-ring of v);
for each triangle T of X
   for each edge e of T
      if e is visited
         continue;
      else
         e ← visited;
         v1, v2 ← two vertices of e;
         N_T ← the triangle sharing e with T;
         /*forward mapping*/
         call adaptive_edge_sampling(v1, v2, T, T, N_T, V, X, τ, L);
         /*backward mapping*/
         call adaptive_edge_sampling(v1, v2, T, T, N_T, −V, X, τ, L);
End
```

7.1.3 Complete Algorithm

The logic of the complete algorithm is shown in Algorithm 2. I first trace each vertex v of a triangle T forward for the time τ. If it falls in triangle T_i, I add the edges from the triangles of the one-ring neighbors of v to T_i in $\mathcal{F}_\tau$. Second, I trace each vertex v of T backward with τ and find the triangle T_i' containing the advected vertex of v. I then add the edges from T_i' to the one-ring neighbors of v. Note that if the image of v is located at a vertex v' or on an edge e, T_i (or T_i') becomes a set of one-ring triangle of v' or the two triangles sharing the edge e. Third, I compute the image of each edge following the original flow and inversed flow, respectively. The adaptive edge sampling algorithm is employed to produce

an outer approximation in a practical and effective manner. The directed edges are added accordingly during the process. Note that the algorithm doesn't deal with the interior triangles of an image (Figure 7.3) explicitly, since the backward tracing using the same manner of forward tracing stage (i.e. proceed the vertices and edges, respectively) will eventually take care of those interior triangles.

7.1.4 Results and Discussion

I have applied this algorithm to a number of analytical vector fields. Figure 6.2 provides the comparison of different Morse decompositions of a synthetic vector field using the geometry-based method (b) and the τ-maps with different time τ's (c) (d). The ECG of the vector field is shown (a). The corresponding MCGs of the obtained Morse decompositions (Figure 6.2(b-d)) are shown in the second row. From the results, we observe that the geometry-based approach is fast ($0.14s$), but tends to result in a Morse decomposition that is too coarse (only four Morse sets have been extracted), while the MCG derived from an $\mathcal{F}_\tau$ has finer Morse decomposition (Figure 6.2(d), ten Morse sets have been found). Note that the MCG in (d) matches the ECG (a), although they are labeled differently. We also observe that the larger the τ, the finer the Morse decomposition is (i.e. closer to the optimal). Larger τ can provide more detailed information of the flow behavior. On the other hand, larger τ requires more computation time to construct MCGs, and larger integration errors may be introduced as well.

As is indicated in Figure 6.4, the τ-map approach leads to a combinatorial

multi-valued map $\mathcal{F}$ with smaller images (than the geometry-based method) and hence a finer Morse decomposition. An important point that can easily be overlooked is the freedom of choice in the construction of $\mathcal{F}$. I have chosen an approach that is a compromise between accuracy of $\mathcal{F}$ and speed of computation. For problems in which computational time is not a concern we can expand on the adaptive sampling technique and the choice of τ to refine the images. Alternatively, if we know that the original vector field contains significant errors, and since the $\mathcal{F}$ needs only to be an outer approximation, these errors can be incorporated into the construction of the images of $\mathcal{F}$ (Figure 6.5). Thus, even in the presence of considerable small perturbation (Figure 6.5) we can ascertain that the resulting MCG is valid.

7.2 Temporal τ vs. Spatial τ_s

The τ-map introduced previously refers to a time discretization, i.e., every particle travels for a time τ. I refer to it as a temporal τ-map. In many scientific data sets, the vector field magnitude of the underlying flow varies significantly. If a constant time τ is used, the advection of some triangles corresponding to the flow region with a slow speed may not be advected far enough in order to construct the edges of $\mathcal{F}_\tau$. One solution is to choose a τ that makes sure every triangle is advected sufficiently far. However, this is likely to affect the overall performance and introduce errors. Similar problems have appeared in texture- and streamline-based flow visualization. One popular approach is to normalize the vector field before

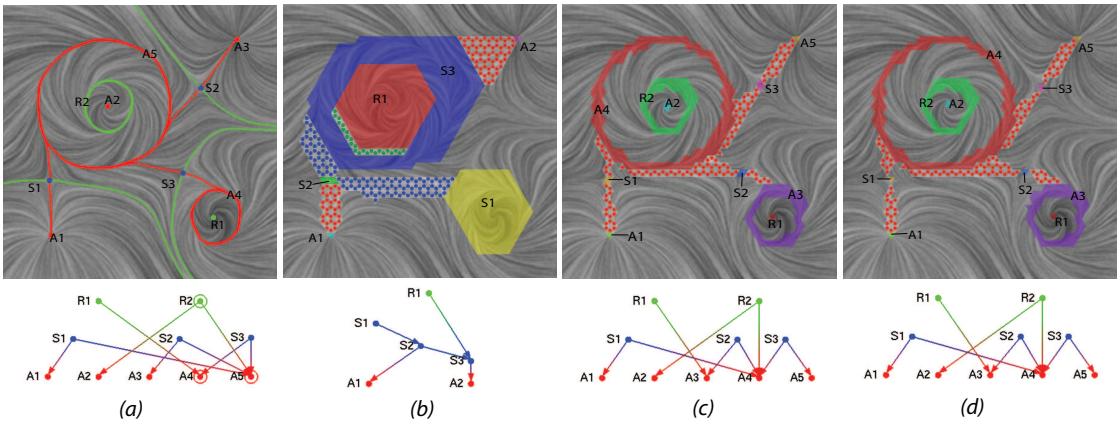


Figure 7.5: This figure shows various analysis results of an analytical data set: (a) ECG, (b) MCG (geometry-based method), (c) MCG (temporal $\tau = 12$) and (d) MCG (spatial $\tau_s = 0.049$). The computational time for (b)-(d) is $0.17s$, $2.42s$ and $1.57s$, respectively. Notice how the Morse sets are refined by using the idea of τ-maps. We also observe that using a spatial τ_s-map for the analysis of this field can give rise to a comparable Morse decomposition (having the same Morse sets) to the one using a temporal τ with a faster performance. The visualization scheme of ECGs and MCGs are described in Figure 6.1.

generating the streamlines or advecting the textures. Under these normalized vector fields, the vector values at the vertices are scaled to have the same magnitude except for fixed points. Therefore, the streamline computation can be executed efficiently. Motivated by this observation, I propose the idea of a *spatial τ-map*, which I refer to τ_s-map.

More specifically, a τ_s-map is defined on a spatial discretization τ_s. When computing a τ_s-map in the computational domain (a triangle mesh X here), for each sample of the triangle T in X I keep track of the integral length of the sample following the flow until the accumulated integral length reaches the spatial constraint τ_s. Since all the particles will travel the same distance in the same speed (e.g. the maximum speed) everywhere except for the neighborhoods of the fixed points, one can expect a faster computation than tracing with respect to the original (non-normalized) vector field. When considering spatial τ_s, we still can reuse the framework in Algorithm 2 to compute the $\mathcal{F}_\tau$ with only difference being that we now accumulate integral length instead of integral time. One important concern is how to compute the correct trajectory when the tracing enters the neighborhoods of the fixed points. The basic rule is that the trajectory should not cross any fixed points. Fortunately, the flow will slow down in those neighborhoods according to the properties of fixed points (where vector magnitude equals zero) and the continuous approximation of the flow guaranteed by the interpolation schemes that are used. Hence, I stop tracing when the vector magnitude is below a certain threshold (for instance, 0.01 times the uniform vector magnitude). I point out that after normalization, we have artificially introduced deviation to the

original vector field.

I apply the idea of spatial τ_s to a designed vector field (Figure 7.5). The geometry domain of the vector field consists of 6144 triangles. Ten Morse sets have been extracted using a temporal $\tau = 12$. The extraction took 2.42 seconds on a 3.0 GHz PC with 1.0 GB RAM. With a spatial τ_s-map ($\tau_s = 0.049$), I extract the similar Morse sets using only 1.57 seconds. The result of the geometry-based method is also shown (Figure 7.5(b)). The corresponding MCGs and ECG of the field are also shown in the bottom row of Figure 7.5. Based on the results, we observe that using a spatial τ_s, we can achieve faster Morse decomposition (Figure 7.5(d)). The use of τ_s also extends our understanding of τ-maps. In the previous section, I set a constant τ for all flow regions during the $\mathcal{F}_\tau$ computation. It is not necessary and may lead to distortion of the outer approximation when large τ is used. The success of τ_s-maps shows that it is possible to use different τ's in different flow regions. This is because given a constant distance τ_s and different flow speed v_s, we will obtain different tracing time $t = \tau_s/v_s$ in different flow regions. Therefore, more heuristic information from the dynamics of the flow can be employed to guide the choice of a proper τ for a specific flow region. This is the challenge I plan to address in future research.

7.3 Applications to Simulation Data

In this section, I provide the analysis results of the given vector fields using the efficient Morse decomposition framework for two engine simulation data sets. They

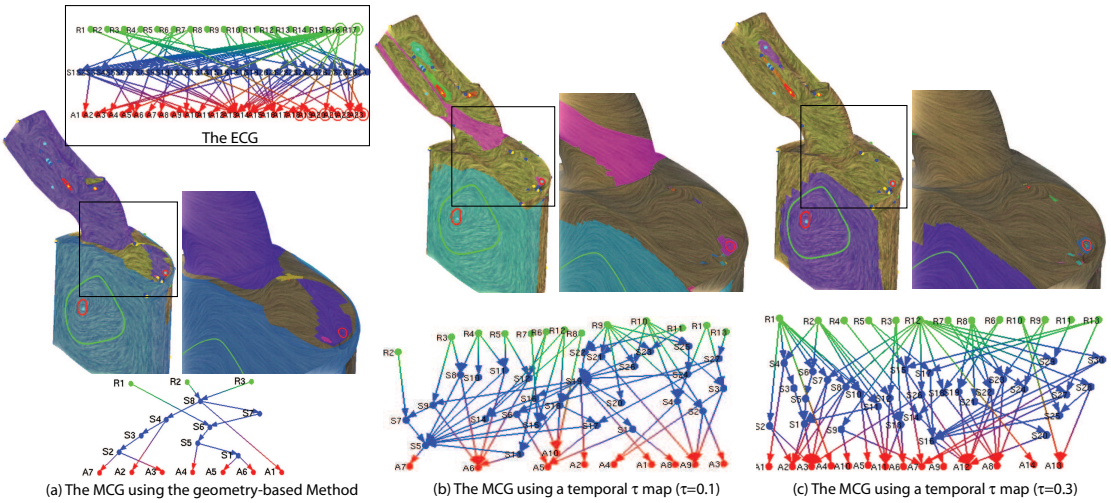


Figure 7.6: This figure compares the results of the Morse decompositions of the gas engine simulation data obtained using geometry-based method (a), a temporal τ-map with $\tau = 0.1$ (b) and a temporal τ-map with $\tau = 0.3$ (c), respectively. Note that the color disk-like region at the back of the cylinder bounds the area of recirculating flow corresponding to tumble motion which indicates an ideal pattern of motion with good mixing properties. Notice that using the τ-maps can greatly improve the quality of the Morse decomposition (the zoom in images). The corresponding MCGs of different Morse decompositions and the ECG of the data are also shown.

are the extrapolated boundary velocity fields that are obtained through simulation of in-cylinder flow. Engineers are interested in knowing whether or not the flows on the surface follow the ideal patterns (Chapter 3) [42].

Figure 7.6 shows the results of the gas engine simulation data. The first column shows the results using the geometry-based method. The second and third columns provide the results using the temporal τ-maps with $\tau = 0.1$ and $\tau = 0.3$, respectively. The corresponding MCGs are also displayed under the flow images. We observe that a Morse set has been extracted at the back of the chamber. It shows a recurrent pattern which indicates the flow starting to approximate the ideal tumble motion. The Morse sets obtained based on a τ-map capture regions that are more faithful to important features, while the approach using the geometry-based map could give rise to fewer Morse sets that cover large regions, which makes the identification of important features more difficult.

The results shown in Figure 7.7 are from the diesel engine simulation. The first column shows the results using the geometry-based method. Notice the rainbow-like regions indicate the recurrence behavior that does not actually exist. That is, the geometry-based method generates a Morse decomposition with misleading information. In the remaining columns, I provide two Morse decomposition results of the same data using a temporal τ-map ($\tau = 0.3$) and a spatial τ_s-map ($\tau_s = 0.08$), respectively. For the temporal case, the obtained Morse decomposition contains 200 Morse sets. It took $1,146.807$ seconds to obtain the result. For the spatial case, the number of the extracted Morse sets of the Morse decomposition is 201. The time for computing this Morse decomposition is 740.826 seconds. Either temporal

Table 7.1: The complexity and timing results for two CFD data simulating in-cylinder flow through a combustion engine (Figures 7.6 and 7.7). Times (in seconds) are measured on a 3.6 GHz PC with 3GB RAM.

dataset name(τ)	# polygons	# edges in $\mathcal{F}_\tau$	# Morse sets	time constructing $\mathcal{F}_\tau$	time extracting Morse sets	time computing MCG	time total
gas engine (temporal$\tau = 0.1$)	26,298	195,694	50	27.844	0.218	7.922	35.984
gas engine (temporal$\tau = 0.3$)	26,298	215,774	57	75.357	0.25	1.219	76.826
diesel engine (temporal $\tau = 0.3$)	221,574	2,035,133	200	1,101.323	7.781	37.703	1,146.807
diesel engine (spatial $\tau_s = 0.08$)	221,574	2,167,914	201	689.451	8.141	43.234	740.826

τ method or spatial τ_s method provides accurate information of the recurrence behavior of the bottom of the in-cylinder of the diesel engine, but the spatial τ_s-map shows faster $\mathcal{F}_\tau$ computation than temporal τ-map scheme.

Table 7.1 provides the performance information of the two data sets using different $\mathcal{F}_\tau$s.

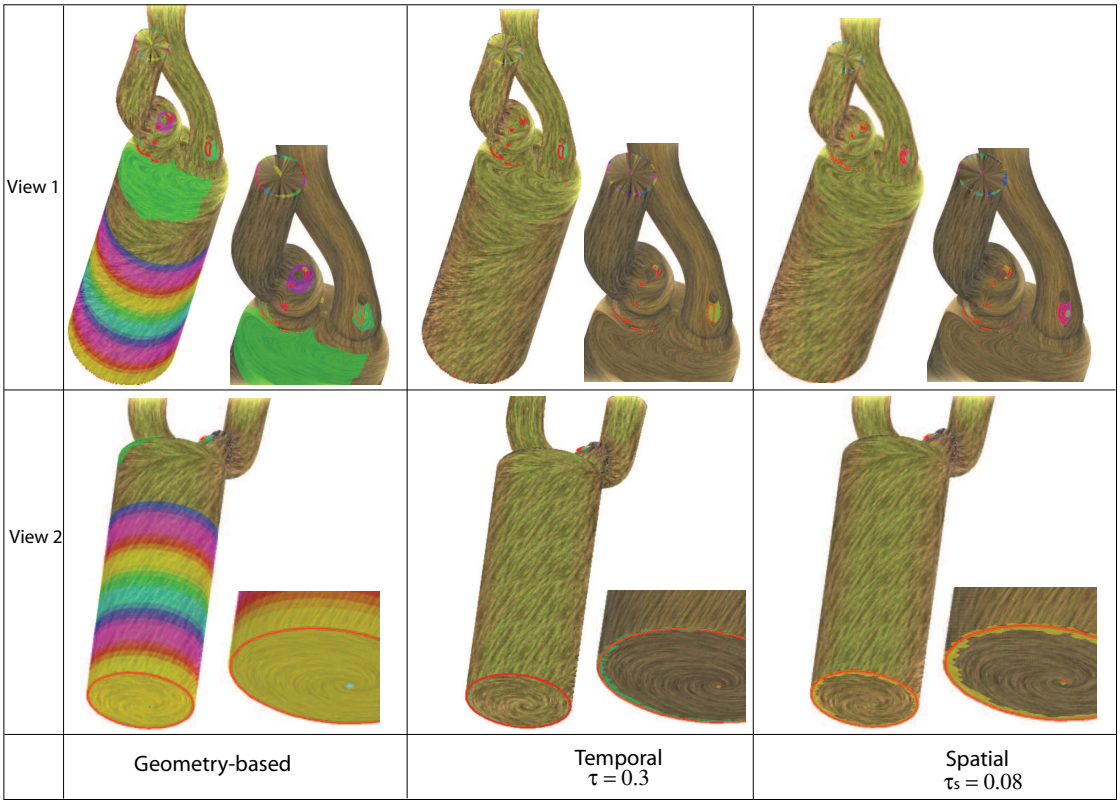


Figure 7.7: A comparison of various Morse decompositions of the diesel engine simulation data set. The first column shows the Morse neighborhoods obtained using the geometry based mapping. The color rainbow-like regions indicate the possible recurrent flow behavior. The second column provides the results using a temporal τ-map with $\tau = 0.3$, while the third column gives the results using a spatial τ_s map with $\tau_s = 0.08$. Note how much more refined the topological regions become. We also observe that using a proper spatial τ_s, we can obtain comparable Morse decomposition with higher performance (See Table 7.1).

Chapter 8 – Hierarchical Refinement of Morse Decompositions

In previous discussion, I show that given a vector field, the MCG is not unique. It is desirable to obtain fine MCGs (Figure 1.3, (e)). An MCG can be made finer by making use of the idea of a τ-map (Chapters 6 and 7) that is computed using particle tracing with respect to a time τ (see Chapters 6 and 7). Because an ideal τ value is typically not known for a given flow, the user must carry out multiple computations with different τ values before a satisfying result is achieved. Furthermore, each recomputation is conducted for the whole flow domain. This leads to a slow analysis process which can be prohibitive for large datasets. In this chapter, I re-examine the Morse decomposition with a focus on the τ-map computation of vector fields and propose an efficient Morse decomposition framework based on a hierarchical refinement process.

8.1 Overview

The slow performance of the previous τ-map approach stems from the repetitive computation during the *flow combinatorialization* stage which relies on a large amount of particle tracing operations. Flow combinatorialization encodes the flow dynamics into a standard directed graph where the nodes are the polygonal primitives of the space discretization (e.g. triangles) and the edges indicate the mapping

relations between polygons. τ-map computation is employed to determine these mapping relations. As mentioned, multiple trials of recomputing the whole flow combinatorialization (a directed graph) with different τ values are needed during the analysis. Furthermore, this approach does not exploit the results of previous computation as they are treated as separate experiments rather than an integrated process. In this chapter, I modify this procedure as follows. I first perform MCG computation using the *geometry-based method* [8] which is fast but coarse. Next, I enter an iterative process in which a Morse set in the current MCG is identified and refined through localized flow combinatorialization with the τ-map method of increasing τ values. This process repeats until none of the Morse sets can be further refined. See Figure 1.3 (a)-(d) for an example. The validity of the proposed approach is justified by Graph Theory. The presented framework is faster because the more expensive computations (higher τ values) are only performed in a small percentage of the mesh. Furthermore, particle tracing results from smaller tau values can be reused for higher τ values.

One of the key elements in my approach is the ability to identify the Morse sets that need further refinement. A simple approach is to let the user manually select a Morse set to refine that may contain more detailed information. This approach provides the user sufficient control but is labor intensive and error-prone when a large data set is investigated. Therefore, an automatic scheme is more preferable. The challenge of an automatic scheme lies in defining proper metrics needed to identify the Morse sets that require refinement. To handle this, I investigate a number of metrics including the *Conley indices* of Morse sets. I show

that the complexity of a Conley index typically indicates the complexity of the flow features inside the region of interest (Section 2.3). This makes Conley index a natural criterion to identify the Morse sets with more complicated flow behaviors. Due to the challenge posted by the flow combinatorialization graph, instead of computing the actual Conley index of a given Morse set I compute its upper bound. In practice, this upper bound is a sufficient approximation to the actual Conley index. In addition, when the flow combinatorialization graph is computed using a geometry-based method, the obtain upper bound is identical to the actual Conley index of a Morse set.

8.2 Hierarchical Morse Decompositions

Theory of dynamical systems shows that an isolating neighborhood (a polygonal region under discrete setting) exists for each Morse set [35]. In addition, in Chapter 7 I have demonstrated that computing the flow combinatorialization does not require a constant τ value everywhere in the domain [9]. This leads us to a local refinement scheme with various τ values. Next I describe the pipeline which is inspired but significantly different from the approach of shown in Chapters 6 and 7.

First, an MCG is computed from the Morse decomposition using a geometry-based flow combinatorialization. Denote the resulting Morse sets as $M(X, V)$ associated with their connectivity graph MCG. Second, I compute an upper bound of the Conley index of each detected Morse set (Section 2.3) and compute a priority

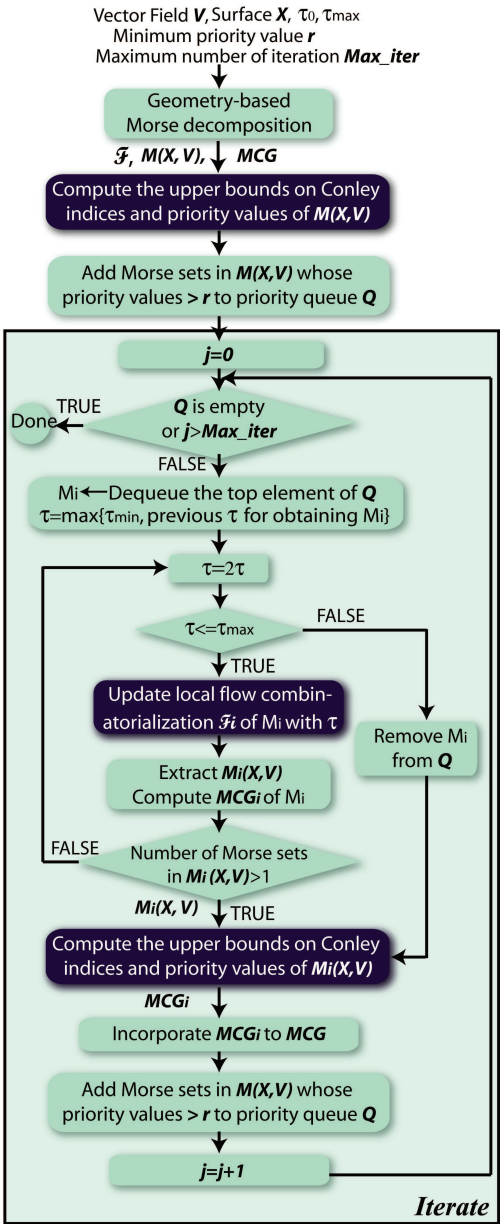


Figure 8.1: The pipeline of the proposed locally hierarchical refinement of Morse decompositions of vector fields.

value based on this upper bound, the area of the Morse set, and the variance of vector magnitude inside it (Section 8.3). Third, I add these Morse sets into a priority queue Q with the Morse set having larger priority values close to the top of the queue. I also initalize $\tau = \tau_{min}$ and set $num_{iter} = 0$. In practice, a good heuristic of τ_{min} is the ratio between the average edge length of local mesh of a Morse set and the minimum vector magnitude inside the Morse set. Fourth, if the first Morse set in Q has a priority value larger than the user specified minimal priority value r. I proceed as follows. Let M_i be the first Morse set in Q. I remove it from Q, and set $\tau = 2\tau$. Then, I perform a τ-map based flow combinatorialization inside the bounded Morse neighborhood of M_i to obtain an updated directed subgraph $\mathcal{F}_i$. Next, the strongly connected components $M_i(X, V)$ of $\mathcal{F}_i$ are extracted and a local MCG_i is computed. If the number of Morse sets in $M_i(X, V)$ is larger than 1, which means selected Morse set is refined, I compute the upper bounds of the Conley indices of the new Morse sets and their priority values. Then, I incorporate MCG_i into MCG and add $M_i(X, V)$ into Q based on their priority values. If the number of Morse sets in $M_i(X, V)$ equals 1 and $2\tau \leq \tau_{max}$ (where τ_{max} is a user specified maximum τ. A good example of τ_{max} is $2^3\tau_{min}$), I increase τ as $\tau = 2\tau$ and proceed as before until either $M_i(X, V)$ contains more than one Morse set or $\tau > \tau_{max}$. If the selected Morse set can not be refined even if $\tau > \tau_{max}$, I remove this Morse set from the priority queue and should not consider it in the future process. Then, I proceed to step 4 and iterate the process until Q is empty, i.e., no Morse sets have a larger metric than the threshold r or have been tested and found to be not refinable. Figure 8.1 illustrates this pipeline.

This pipeline proceeds in a hierarchical fashion and is expected to produce a MCG (Figure 1.3, (d)) similar to the one produced by previous manual τ-map approach with respect to globally applied τ (Figure 1.3, (e)).

8.2.1 Local Flow Combinatorialization

Similar to the general Morse decomposition, in the hierarchical Morse decomposition pipeline the local flow combinatorialization is a key step. In this section, I show that locally updating the flow combinatorialization will not affect the flow structure with respect to MCG outside of the bounded Morse neighborhood of interest. Before presenting the theorem on that, I first prove the follow lemma.

Lemma 8.2.1 *Given a directed graph $G = (V, E)$. Let $V = \cup_i V_i$ be the decomposition of strongly connected components of G. Let $E_i = \{(v_p, v_q) \in E | v_p, v_q \in V_i\}$. Select an integer j, let $G' = (V, E')$ such that $E' = (E - E_j) \cup E'_j$ where E'_j contains edges whose end points are both in V_j and $E_j \neq E'_j$ (i.e. they represent different sets of edges). We make two claims:*

1. *Any strongly connected component V_i in G with $(i \neq j)$ is contained in a strongly connected component in G'.*

2. *Any strongly connected component V'_i in G' is contained in a strongly connected component in G.*

If we can show 1 and 2 are true, then we immediately have the following: A strongly connected component V_i $(i \neq j)$ in G is also a strongly connected component in G'.

On the other hand, V_j corresponds to possibly more than one strongly connected components.

Proof: To show (1) is true, let a, b be two nodes in V_i ($i \neq j$), i.e., there are paths entirely contained in E_i from a to b and b to a. Given the construction of G', we know these paths are also contained in G'. Consequently, it is possible for a to reach b and vice versa using edges in G'. This means a and b are in the same connected component of G'.

To show (2) is true, assume that a and b are two nodes in the same connected component in G' but in different strongly connected components V_a and V_b with respect to G. This means that there is a minimal oriented loop γ containing a and b using edges in E'.

Since $V_a \neq V_b$, one of them is different from V_j. Without loss of generality, assume $V_a \neq V_j$, i.e., $a \notin V_j$.

Let $s \in V_j$ be the last node on the loop γ before reaching a, and $t \in V_j$ the first node on γ after a. Note that either s and t both exist or neither exists. In the first case, we have a path from s to t using edges in $E - E_j$ through a, i.e, such a path exists in G. Note that s and t both belong to E_j, so there is a path from t to s with edges in E. Consequently, there is a loop from s to a with edges entirely in E. Consequently, a and s belong to the same strongly connected component V_j since $s \in V_j$. However, this contrasts our assumption $V_a \neq V_j$. In the second case, i.e., there are no nodes on γ that belong to V_j. Consequently, edges on γ are entirely contained in $E - E_j \subset E$. This means a and b belong to the same connected component. However, this contradicts our assumption that $V_a \neq V_b$.

Q.E.D.

Theorem 8.2.2 *Consider a flow combinatorialization $\mathcal{F} = (V, E)$ of a vector field computed with respect to a τ. Let $M = \cup_i M_i$ be the set of extracted Morse sets from $\mathcal{F}$, $\mathcal{F}_i = (V_i, E_i)$ be the subgraph of $\mathcal{F}$ with V_i the set of triangles corresponding to the Morse neighborhood of M_i, and $E_i = \{(v_p, v_q) \in E | v_p, v_q \in V_i\}$. For an integer j, let $\mathcal{F}' = (V, E')$ such that $E' = (E - E_j) \cup E'_j$ where E'_j consists of edges that are computed using a $\tau' > \tau$ and whose end nodes are both in V_j and $E_j \neq E'_j$ (i.e. they represent different sets of edges). Then,*

1. *Any Morse set M_i of $\mathcal{F}$ with $(i \neq j)$ is a Morse set of $\mathcal{F}'$.*
2. *M_j is a) a Morse set, b) decomposed to be more than two separate Morse sets, or c) removed from M.*

Combining 1 and 2 of Theorem 8.2.2 I prove: By fixing all Morse sets but one and replacing edges of the local flow combinatorialization graph for that specific Morse set using a larger τ value, it is to obtain a likely refined Morse sets for that Morse set without changing the rest of $\mathcal{F}$ in practice. Consequently, the whole MCG is refined. Note that in some cases the selected Morse set needs not be a true Morse set (with non-trivial Conley index) and can be removed from the Morse set list after replacing its local flow combinatorialization graph with a more accurate one. This is because the flow combinatorialization constructed using a small τ or the geometry-based method is typically coarse and contains misleading information, e.g. inaccurate Morse sets (see the rainbow-like regions in Figure 7.7,

left column). Nonetheless, this error can be corrected by a more accurate graph obtained using a larger τ (Figure 7.7, middle and right columns). The proof of this theorem follows from Lemma 8.2.1. This theorem also provides the algorithm to realize the local MCG refinement which I will describe in the following section.

8.2.2 Implementation

Given a Morse set M_i, I now explain how to refine it locally. Let $\mathcal{F}_i$ be its local flow combinatorialization graph whose nodes are the triangles of the Morse neighborhood of M_i and directed edges encode the flow dynamics in M_i. First, I remove all the edges that are completely in $\mathcal{F}_i$, i.e, the edges whose both end points are nodes in M_i. Second, I reconstruct $\mathcal{F}_i$ by applying the τ-map approach locally inside M_i. Third, I extract strongly connected components from the updated $\mathcal{F}_i$. Finally, I compute the subgraph MCG_i based on the extracted Morse sets and incorporate it into the original MCG. Algorithm 1 provides the pseudo code of this process. The reconstruction of local flow combinatorialization is crucial. To accomplish that, I implement a function similar to the *construct_multivaluemap* routine (Section 7.1) but input a bounded flow region $X_i \subset X$. The obtained flow combinatorialization graph $\mathcal{F}_{temp}$ is incorporated into $\mathcal{F}$ with edges not completely falling in X_i excluded. This process is described in the routine $Reconstruct\_\mathcal{F}_i()$.

Figure 8.2 provides an example of the MCG refinement results using the proposed local updating scheme. I start from an MCG (left) computed from a flow

Algorithm 1: Locally refining one Morse set

Input: M_i: the given Morse set
X_i: the list of triangles of Morse set M_i
$\mathcal{F}_i$: the local flow combinatorialization of M_i
$V(M_i)$: the local flow inside Morse set M_i
L: the maximum level for adaptive edge sampling
Output: MCG_i: the local MCG with updated Morse sets
Begin
$Delete\_Edges\_in\_\mathcal{F}_i(\mathcal{F}_i)$;
$\mathcal{F}_i \leftarrow Reconstruct\_\mathcal{F}_i(X_i, V(M_i), \tau, L)$;
$Extract\_SCC\_\mathcal{F}_i(\mathcal{F}_i)$;
$MCG_i \leftarrow Construct\_Local\_MCG_i(\mathcal{F}_i)$;
End

Local flow combinatorialization using τ-map
Routine: $Reconstruct\_\mathcal{F}_i(X_i, V(M_i), \tau, L)$

Input: X_i: the list of triangles of Morse set M_i
$V(M_i)$: the local flow inside Morse set M_i
τ: a positive real number
L: the maximum level for adaptive edge sampling
Output: $\mathcal{F}_i$: the local flow combinatorialization of M_i
Local variables: $\mathcal{F}_{temp}$: temporary $\mathcal{F}_i$
e: an edge in $\mathcal{F}_{temp}$
$node$[2] : two nodes that connected by e
Begin
$\mathcal{F}_{temp} \leftarrow construct\_multivaluemap(V(M_i), X_i, \tau, L)$;
For each edge e in $\mathcal{F}_{temp}$
If $node[0] \notin X_i$ or $node[0] \notin X_i$
remove e from $\mathcal{F}_{temp}$;
EndFor
$\mathcal{F}_i \leftarrow \mathcal{F}_{temp}$;
End

combinatorialization using a geometry-based method. Next, I perform the local updating inside the obtained Morse sets with large area. Note that in addition to the refined Morse sets, the connection regions [9] between two Morse sets are refined as well due to the refinement of the underlying $\mathcal{F}$ which I use to compute the connection region (the dotted regions in Figure 1.3 (a)-(d)).

8.2.3 Conley index of Morse sets obtained with τ-map

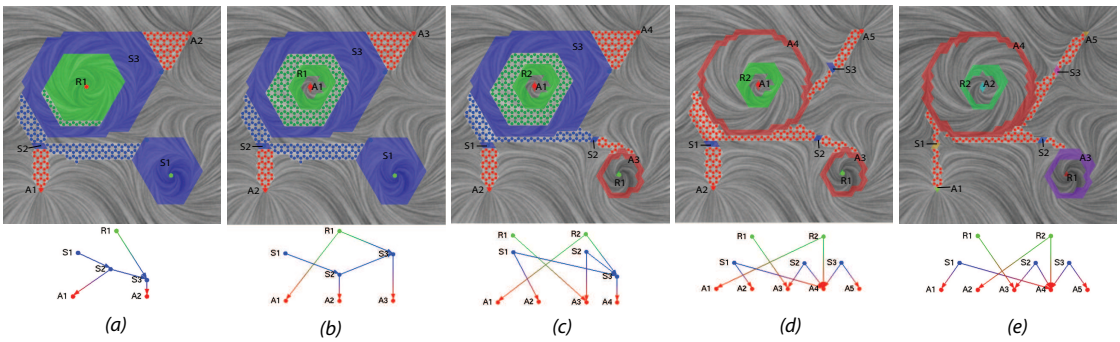


Figure 8.2: This figure provides an example of locally refining a Morse decomposition of an analytic flow data over different Morse sets (b): R1, (c): S1, and (d): S3 with $\tau = 7.8$, 10, 10, respectively. (a) provides the Morse decomposition using a geometry-based method. In addition, the MCG generated using a global τ-map idea with $\tau = 12$ is also shown in (e) for comparison. Note how our finest MCG is comparable to the one using global Morse decomposition. Different colors indicate different Morse sets. The color-dotted regions indicate the connection between Morse neighborhoods. Note that the connection regions are also refined during the process. In the MCGs, green dots stand for the source Morse sets, red dots for the sink Morse sets, and blue dots for the saddle Morse sets.

In this section, I introduce a simple and efficient algorithm to compute an upper bound on the Conley index of a Morse set M obtained using the τ-map. In all cases we looked at, the bound is identical to the Conley index (hence I call it the *estimate* of the Conley index) and is much simpler to compute. In addition, being an upper bound means that a Morse set containing complex dynamics will not be missed during the identification stage.

Because we use flow combinatorialization the detected Morse sets are not guaranteed to be isolating neighborhoods. Fortunately, they can still be used to relate the topological properties of the *map* ϕ_τ that moves any point by τ along its trajectory to the Conley index of the flow. This map can be viewed as a dynamical

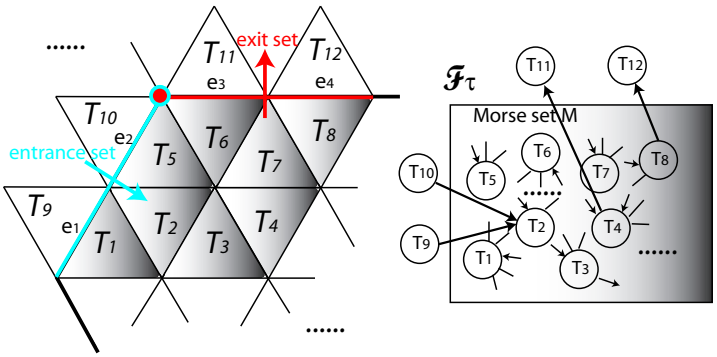


Figure 8.3: This illustrates the classification of boundary edges. Image to the left provides a portion of the mesh with a Morse set M_i inside the shadow region. Right diagram provides the configuration of a discrete map (i.e. a flow combinatorialization) $\mathcal{F}_\tau$. Note that I ignore the inner configuration of the graph inside the Morse set M_i because it does not affect the classification.

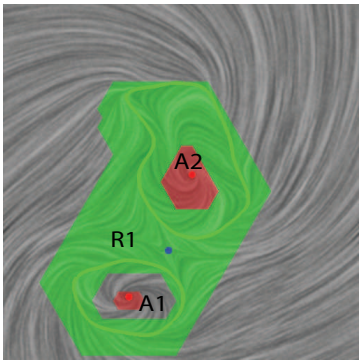
system with *discrete* rather than continuous time. Generalizations of the Conley index theory to flows with discrete time are described in [47, 58, 68].

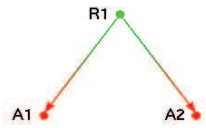
The analysis is based on the concept of an *index pair*, that is used to define the Conley index for dynamical systems with discrete time. For a continuous map f, an index pair (N, L) may be defined in a few ways. The key property that needs to be satisfied is the existence and continuity of the map induced by f on the quotient space N/L [58]. Recall that the quotient space is a pointed topological space obtained by collapsing all points in L to a single distinguished point. Thus, every point in the quotient space is represented by some point in N. Points of L represent the distinguished point. The point of N/L represented by $x \in N$ is denoted by $[x]$. The induced map takes a point $[x]$ into $[f(x)]$ if $f(x) \in N$. Otherwise, it takes $[x]$ into the distinguished point.

A relationship between the map induced by ϕ_τ on N/L and the Conley index

of the flow is described in [47]. Let (N, L) be an index pair for ϕ_τ and $\phi_{\tau,k}$ be the induced map on k-dimensional homology of N/L. The k-dimensional ($k \leq 2$ in our discussion) Betti number of the Conley index of the flow is equal to $\lim_{n\to\infty} \text{rank}\phi^n_{\tau,k}$. Typically, homology with coefficients in a field is used in this computation, which makes $\phi_{\tau,k}$ a linear transformation acting on $H_k(N/L)$. The rank of any iterate of $\phi_{\tau,k}$ is smaller or equal than the dimension of $H_k(N/L)$. We conclude that the k-dimensional Betti number of the Conley index of the flow is less or equal to the k-dimensional Betti number of the quotient space N/L. This is the upper bound I use in the rest of the thesis.

A procedure for computing an index pair for a continuous map based on a regular grid discretization of the domain is described in [69]. Even though I use a discretization based on an irregular triangle mesh in this work, the results still hold and the proofs are exactly the same. An index pair for a Morse set M is (M, L) where L consists of boundary edges of M that I still call *exit edges*. In order to decide if an edge e on the boundary of M is an exit edge, take a triangle T incident upon e and outside M. e is an exit edge if and only if there is an edge of the flow combinatorialization that *starts* at a triangle in M and *ends* at T. Figure 8.3 illustrates an example. Consider a boundary edge e_1 shared by triangles T_1 and T_9 with T_1 outside of M. There is a directed edge pointing from T_1 into a triangle T_2 of M in $\mathcal{F}$. Therefore, e_1 is classified as an entrance edge, so is e_2. Now consider edge e_3. Triangle T_{11} is incident to e_3 and out side of M. There is a directed edge from T_4 inside M to T_{11}. Hence, e_3 is an exit edge. Similarly, we classify e_4.





MCG

Morse Set	X(M)	Exit Set B(0)	Exit Set B(1)	Exit Set B(2)	X(L)	Upper Bound Of Conley Index B(0)	Upper Bound Of Conley Index B(1)	Upper Bound Of Conley Index B(2)
R1	-1	3	3	0	0	0	2	1
A1	1	0	0	0	0	1	0	0
A2	1	0	0	0	0	1	0	0

X(M): Euler characteristic of M
X(L): Euler characteristic of boundary exit set L
The left B(0)-B(2) are Betti numbers for L
The rightmost B(0)-B(2) are the Betti numbers of the upper bound

Figure 8.4: This figure illustrates an example on how the upper bound of the Conley index can help identify Morse set with complex flow. Note that Morse set $R1$ has an upper bound on its Conley index as $(0, 2, 1)$. In the meantime, the flow in this Morse set contains two repelling periodic orbits (green loops) and a saddle (blue dot). Therefore, based on its upper bound, we determine that $R1$ should be further refined.

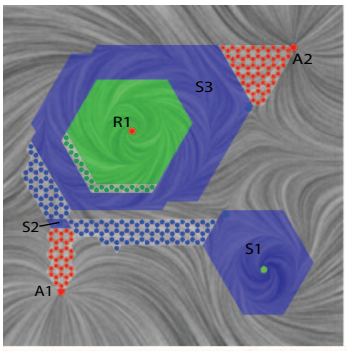
From the above discussion, I conclude the Betti numbers of the quotient space M/L, where (M, L) is the index pair for ϕ_τ described above, are an upper bound of the Betti numbers of the Conley index of the flow on Morse set M. It is worth noting that this computation is essentially identical to the examples of Conley index computation shown in Section 2.3.2.The only difference is the approach of determining the exit edges. In the examples of Figure 2.3, I make use of the flow at the edges to classify the edges, while in upper bound computation, the directed edges in the flow combinatorialization graph $\mathcal{F}$ are used for classification. Furthermore, the aforementioned computation on a discrete map (a flow combinatorialization graph) obtained using a geometry-based method returns the true Conley index of the given region based on the flow at the boundaries. Figure 8.4 provides such

an example. I have applied this computation algorithm of upper bound to two analytic datasets. Figure 8.5 provides the upper bounds of the Conley indices of the Morse sets extracted from two designed vector fields. The left image shows the Conley indices of the Morse sets from a geometry-based method, while the right displays those from a $\tau-$map based approach. Note that in the implementation, I include the strongly connected components whose Conley indices are trivial but with larger size (i.e. containing more than two triangles). This typically indicates regions with more than one recurrent features such that the sum of their Conley indices is trivial. $S1$ of Figure 8.5 (left) is such an example. Therefore, further refinement of these "Morse sets" is likely to produce finer decompositions.

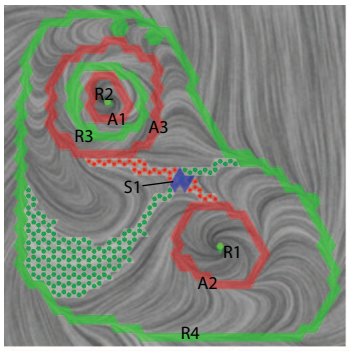
8.3 Identifying Morse Sets To Refine

In this section, I introduce a number of metrics that are used to identify Morse sets for refinement in the automatic framework.

The first intuition in the selection of a Morse set to refine is to consider the Morse set with complex flow inside the corresponding region. From the previous discussion on Conley index and its computation, we observe that the Conley index provides information on the complexity of the flow. For instance, if the Conley index of a Morse set is $(0, 2, 1)$ (Figure 8.4, $R1$), it is likely there could be separable periodic orbits and a saddle. Recall that Section 2.3 provides a number of important Conley indices. They also represent the Conley indices of the five simplest invariant sets. To measure the complexity of the Conley index of a Morse set M,



Morse Set	X(M)	Exit Set B(0)	Exit Set B(1)	Exit Set B(2)	X(L)	Upper Bound Of Conley Index B(0)	Upper Bound Of Conley Index B(1)	Upper Bound Of Conley Index B(2)
R1	1	1	1	0	0	0	0	1
S1	1	1	0	0	1	0	0	0
S2	1	2	0	0	2	0	1	0
A1	1	0	0	0	0	1	0	0
S3	0	1	0	0	1	0	1	0
A2	1	0	0	0	0	1	0	0



Morse Set	X(M)	Exit Set B(0)	Exit Set B(1)	Exit Set B(2)	X(L)	Upper Bound Of Conley Index B(0)	Upper Bound Of Conley Index B(1)	Upper Bound Of Conley Index B(2)
R1	1	1	1	0	0	0	0	1
R2	1	1	1	0	0	0	0	1
R3	0	2	2	0	0	0	1	1
A1	0	0	0	0	0	1	1	0
R4	0	2	2	0	0	0	1	1
S1	1	2	0	0	2	0	1	0
A2	0	0	0	0	0	1	1	0
A3	0	0	0	0	0	1	1	0

Figure 8.5: The computed upper bounds of the Conley indices of all Morse sets extracted from two analytical vector fields. (left) shows the results using a geometry-based method, while (right) provides the results of a MCG derived from a τ-map with $\tau = 24$. Note that the upper bounds for the Morse sets in the left example are their actual Conley indices. In addition, in my experience the obtained upper bounds for the Morse sets computed from a τ-map approach are typically equal to the ideal Conley indices, such as, in the example to the right.

we compute the distances between it and the five basic Conley indices as

$$dist(CH_*(M), CH_*(x_0)) = \sum_{k=0}^{2} |\beta_k(M) - \beta_0(x_0)|$$

The shortest distance then indicates how complex the flow inside the Morse set based on the flow behaviors on the boundaries of M. Therefore, I make use of this shortest distance of a given Conley index to the five basic Conley indices as one criterion. This should provide the necessary topological information of the region of interest. I refer to this metric the *topology metric*.

Still, there are a number of cases that examining Conley index is not sufficient. For instance, the Morse set $S3$ in Figure 8.5 (left) has a simple Conley index $(0, 1, 0)$ which is the same as region containing only a saddle, while detailed analysis shows a saddle and an attracting periodic orbit are enclosed. To handle this, I make use of the number of triangles in each Morse set as a heuristic for identification. I refer to this metric the *geometry metric*. It is intuitive that a Morse set covering a large portion of the flow domain may contain more detailed dynamics. For instance, Figure 8.7 (left) shows the result of the Morse decomposition of the gas engine dataset using a geometry-based approach. Note that there is a Morse set at the back of the cylinder of the engine which covers a large portion of the engine surface. I wish to point out that in our experiments some regions of flow recurrence have trivial Conley index $(0, 0, 0)$ (e.g. $S1$ in Figure 8.5, left). They consist of more than one triangle which indicates multiple features (i.e. invariant sets) included. This causes the Conley index of the region to be trivial, a property similar to Poincaré index which is particularly useful in topological simplification [80]. Therefore, in all experiments I include this kind of regions with trivial Conley index but more than one triangle in the MCG.

Combining the aforementioned topology and geometry factors, I define the formula for computing the priority value of each Morse set M_i as follows:

$$P(M_i) = (1 + min\,(dist(CH_*(M_i), CH_*(x_0)))) \times N(M_i) \tag{8.1}$$

where $N(M_i)$ represents the number of triangles in the Morse set M_i. This priority

value is used to determine the order of the refinement of Morse sets in current MCG. The larger the value is, the earlier the Morse set will be refined. Morse sets with priority value smaller than a threshold r will be discarded. In the implementation, I use $r = 2$ as the threshold, because any Morse set whose P value is larger than 2 could contain separable features by equation 8.1. Note that a ring-like region containing a periodic orbit may have larger P value, for instance, Morse sets $R1$ and $R4$ in Figure 8.5 (right). In this case, further refinement will discover that no more Morse sets can be extracted. The system then discards this Morse set from the list of Morse sets that are considered to refine according to the pipeline (Figure 8.1).

I make use of this metric to identify the Morse sets for refinement. Figure 8.7 provides the result of the consecutive refinement of an analytic data.

8.4 Applications

I have applied the proposed hierarchical refinement framework to a number of design datasets. Figure 8.6 (left) provides the result of such a dataset. This dataset consists of $6,144$ triangles. The experiment took 7.58 seconds to return the result given the parameters $r = 2$, $\tau_{max} = 28$, and maximum number of iterations equal 12. I compare the local refinement result with the one obtained using a manually refining process with $\tau = 7$, 14, and 28, respectively. This requires $11.15s$ computation time in addition to the user interaction time. Note that our result achieves the similar MCG structure to the one using a global τ-map

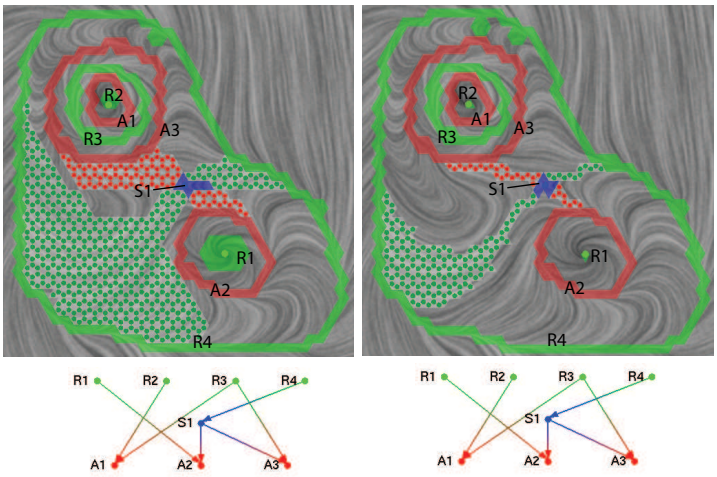


Figure 8.6: The figure provides the result of Morse decomposition using the presented hierarchical framework (left). The minimum priority value r, the maximum allowed τ_{max}, and the maximum number of iterations for the result of this data are 10, 28, and 12, respectively. The time for the computation is 7.58 seconds. Similar result using the previous global τ-map approach is given (right). It is obtained by experimenting $\tau = 7$, 14, and 28, respectively. The times spent on these computations are $2.63s$, $2.9s$, and $5.63s$. Therefore, totally I spend $11.15s$ computation time using manually selecting τ approach.

approach.

I also provide the Morse decomposition results of a gas engine simulation dataset using the proposed hierarchical framework. This data set is the extrapolated boundary velocity fields that are obtained through a simulation of an in-cylinder flow [42]. Figure 8.7 provides the analysis results of this data set composed of $26,298$ triangles. From (1)-(3)of Figure 8.7 I refine the circulated Morse set with $\tau = 0.1$, 0.3, and 0.3, respectively. The times spent on these refinement are $1.469s$, $27.845s$, and $42.094s$, respectively. The MCG obtained using a global $\tau = 0.3$ shows the similar results (right-most) which took around $227s$. to com-

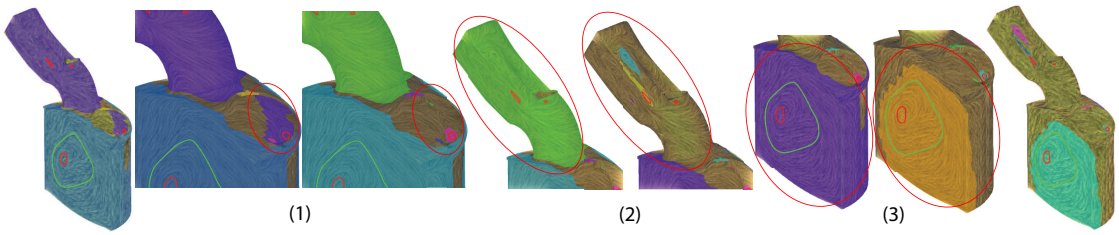


Figure 8.7: This figure illustrates the refinement process of the MCG of the gas engine simulation data. Left-most shows the MCG of a geometry-based approach. From (1)-(3) I refine the circulated Morse set with $\tau = 0.1$, 0.3, and 0.3, respectively. The times spent on these refinement are $1.469s$, $27.845s$, and $42.094s$, respectively. The MCG obtained using a global $\tau = 0.3$ shows the similar results (right-most) which took around $227s$ to compute. This is assuming that the user has already known $\tau = 0.3$ is sufficient for this dataset.

Table 8.1: The complexity and timing results for two CFD data simulating in-cylinder flow through a combustion engine. Times (in seconds) are measured on a 3.6 GHz PC with 2GB RAM. Note that I compare only the performance of the automatic refinement framework with the global τ approach with the $\tau = \tau_{max}$. Additional time spent on smaller τ values and the user interactions for the global τ scheme is not considered. Even though, our automatic refinement framework exhibits better performance in time.

Dataset	#	Local Update			Global Update		
name	polygons	τ_{max}	#Morse sets	time(s)	τ	#Morse sets	time(s)
gas engine	26,298	1	62	158.5	1	62	335.49
diesal engine	221,574	0.4	152	278.7	0.4	207	1,326.08

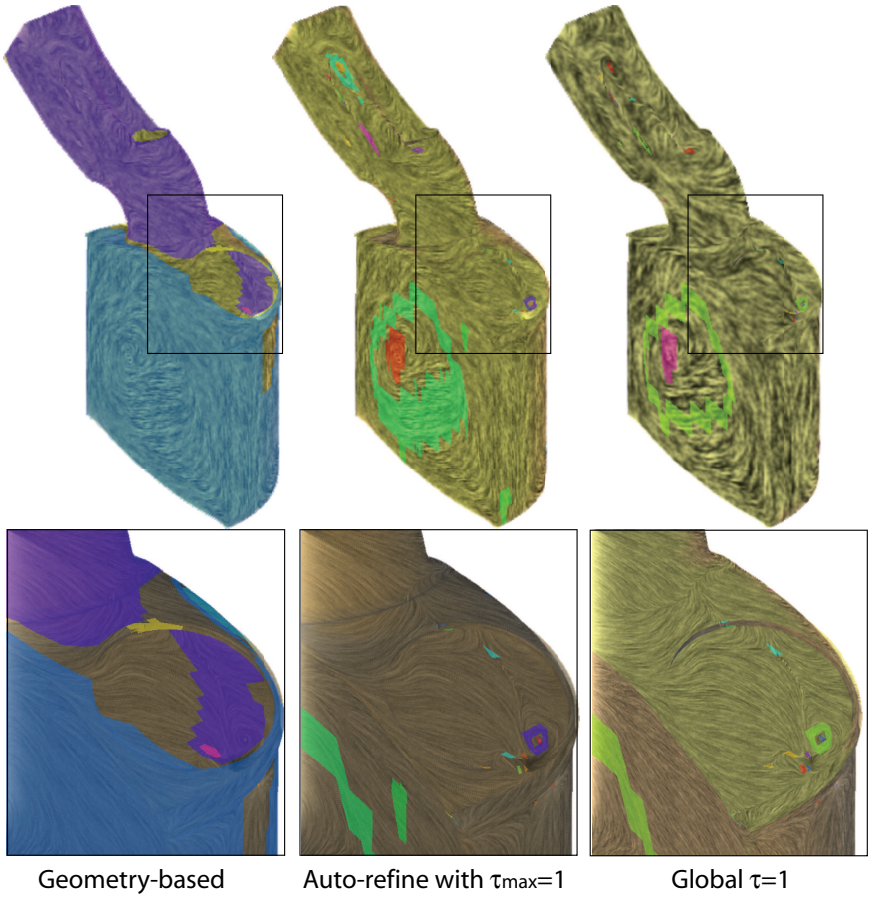


Figure 8.8: This figure compares the results of the Morse decompositions using local refinement (bottom) and global update of τ-map (top) for the gas engine simulation dataset. The setting for the hierarchical refinement is $r = 2$, $\tau_{max} = 1$, and maximum number of iterations equal 40.

pute. This is assuming that the user has already known $\tau = 0.3$ is sufficient for this dataset. Figure 8.8 compares the results of Morse decompositions using the presented scheme of automatic hierarchical refinement and the manually adjusting τ values approach. In the setting of $r = 2$, $\tau_{max} = 1$, and maximum number of iterations equal 40, my framework takes 158.5 seconds to return the results. On the other hand, the manually adjusting τ scheme uses $215.29s$ for $\tau = 0.25$, $277.24s$ for $\tau = 0.5$, and $335.49s$ for $\tau = 1$ before obtaining the final result. This takes $828.02s$ in total.

Figure 8.9 provides the automatic simplification results of a diesel engine dataset with $221,574$ triangles. The running time for this analysis is $278.7s$ under the setting: $r = 2$, $\tau_{max} = 0.4$, and maximum iterations is 60, compared to the $1,326.08s$ using a global $\tau = 0.4$. From these results, we see the proposed automatic refinement framework achieve the analysis results about four times faster than the global τ scheme.

Table 8.1 provides the timing information of the automatic refinement that is applied to the two engine datasets. Note that I compare only the performance of the automatic refinement framework with the global τ approach with the $\tau = \tau_{max}$. Additional time spent on smaller τ values and the user interactions for the global τ scheme is not considered. Even though, our automatic refinement framework exhibits better performance in time. Note all times (in seconds) are measured on a 3.6 GHz PC with 2GB RAM.

Figure 8.9: This figure shows the results of Morse decomposition before (left) and after automatic hierarchical refinement (right) for the diesel engine simulation dataset. The setting for the hierarchical refinement is $r = 2$, $\tau_{max} = 0.4$, and maximum number of iterations equal 60. The Morse sets are colored differently to emphasize their difference. Note that how the automatic framework produces the comparable results (middle) to the ones using a global τ (right).

Chapter 9 – Time-Varying Vector Fields in Graphics Applications

Chapter 5 has elaborated the importance of vector fields for a wide variety of graphics applications. The techniques of time-independent vector field design on surfaces have been well-studied. However, there is little work on the design of time-varying (i.e. time-dependent) vector fields on surfaces. Nevertheless, time-varying vector fields are more common in practice. Different from time-independent vector fields where given a fixed position the vector value remains constant for any time, the vector value of a fixed point in a time-varying vector fields could vary with respect to time. For instance, the velocity field associated with wind is a time-varying vector field. That is, at a given location the wind direction varies over time. Otherwise, weather prediction is no longer necessary. From this chapter, I turn my discussion to a more complicated subject: time-varying vector fields.

9.1 Applications and Impact

In many computer graphics applications, time-varying vector fields arise implicitly, such as the velocity fields in fluid simulation [65, 66], the force fields in crowd animation [77] and hair modeling [23], and the displacement fields in shape deformation [89] and video editing [99]. The ability of design and control of these time-varying vector fields will help researchers and artists achieve controllable fluid

and smoke animation, steerable interactive crowd simulation, temporally coherent video editing and painterly rendering of videos, controllable cloth and hair animation, and tractable object morphing.

Before starting the discussion on time-varying vector fields, it is important to realize the different functions of vector fields in various applications. A vector field can be used to describe many physical phenomena such as displacement, velocity, acceleration, orientation, and momentum. The definitions of features in a vector field is application-dependent. For example, in texture synthesis and animation, one vector field is used to guide the placement of texture patches, while another is used to move the patches. Therefore, understanding the differences between vector fields involved and separating their design tasks will not only ease the discussion of different design interfaces and primitives, but also enable us to achieve various effects. The following introduces two types of vector fields that are employed widely in different applications.

9.2 Orientation Field and Advection Field

In graphics applications such as texture synthesis and animation, two types of vector fields are required as the input. One is used to orient texture patches, which we refer to as an *orientation field*, while the other describes the advection of texture patches over time, which we call an *advection field*. Figure 9.1 shows the effect of both types of fields in texture synthesis and animation. The directions of strip-like pattern depict the orientation field, and the cyan arrows in the left

image illustrate the advection field (difficult to see with still images). We have specified the orientation field only for the first frame, which is then advected by the advection field using the technique proposed by Kwatra et al. [37]. Hence, the orientation field over time need not be stationary. This is reflected through the texture movement in the three frames from the animation (Figure 9.1). Note that the orientation field in this example does not convey any interesting effects on the surface. It is preferable to control this appearance to achieve meaningful effects, for example highlighting certain features of the shapes. Further, for the movement of the graphical properties (e.g. moving a texture patch from one place to the other), it can be valuable to allow the user to design their animation paths. Therefore, it is necessary to address the design of these two types of fields.

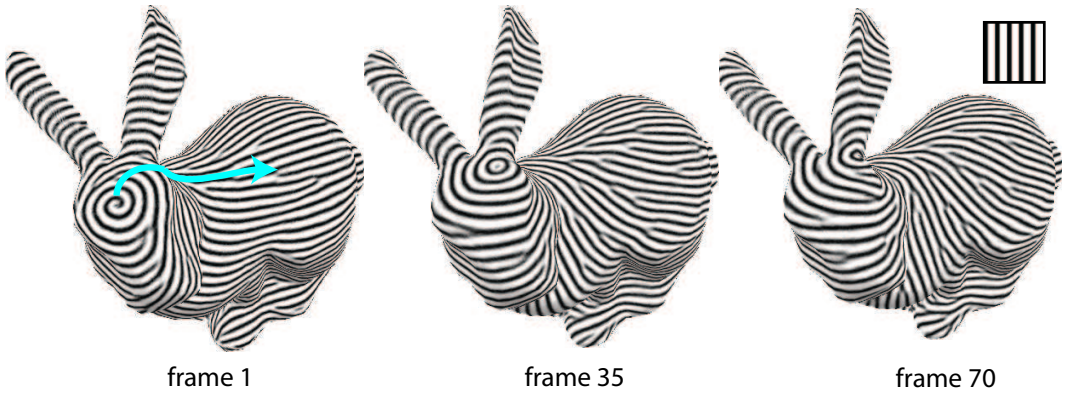


Figure 9.1: This example demonstrates the different utility of orientation field and advection field.

9.3 Requirements and Challenges

Design and control time-varying vector field poses a number of requirements and challenges as well. To be more precise, it has been observed that in time-varying graphics applications solving for a series of time-independent vector fields to produce each frame can lead to visual discontinuity artifacts. It has also been demonstrated that these types of artifacts are less prevalent when graphics applications are generated by solving a time-varying vector field. This suggests the desirability of designing time-varying vector field $V(\mathbf{x}, t)$ with specific properties. This poses a number of challenges to the design task. For example, a number of fundamental concepts, such as a singularity of a vector field, used in steady field design are no longer mathematically well-defined in the time-varying setting. More importantly, the theory for the analysis and control of time-varying dynamics is much more primitive than that for time-independent vector fields.

As previously mentioned, vector fields have different functions in various graphics applications. This requires distinct sets of design functionality and design primitives to pursue specific results that serve different purposes. For instance, the design of instantaneous appearance is the focus in orientation field design, while for advection field the emphasis is on the evolution of time-dependent features such as the trajectories of particles.

In many graphics applications structural changes of certain graphical properties are often observed, such as the splitting and merging of texture structures in texture synthesis and animation. In some cases, these structural changes may

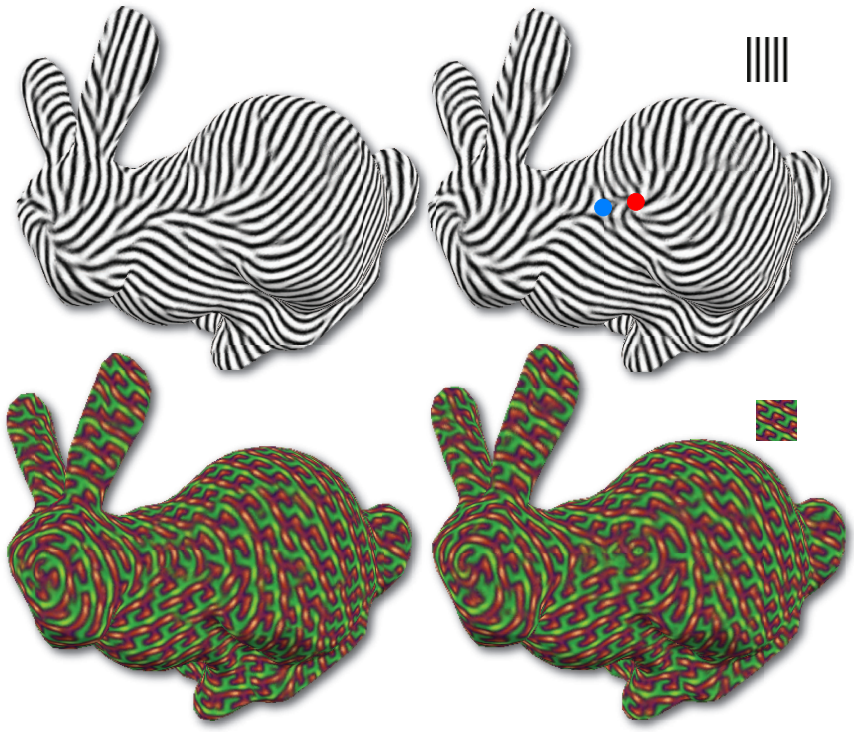


Figure 9.2: This figure shows an example of saddle-node bifurcation in an orientation vector field, the creation of a pair of saddle and sink, which causes the break of texture structure on the back of the bunny. Note that I sample the two frames before (left column) and after (right) bifurcation happens to reveal the discontinuity.

cause visual artifacts. Figure 9.2 shows an example where the break of texture structures could cause visual discontinuity in the animation. These structural variations are typically associated with the topological changes of the underlying fields, i.e. *bifurcations*, which I review later. This poses a requirement of the ability of manipulating them with desired properties.

Chapter 10 – Time-Varying Vector Fields

This chapter provides a brief review of a number of important concepts for the discussion of time-varying vector field design. More importantly, the concept of *parameterized vector fields* is introduced to enable the design of a time-varying vector field in the form of a collection of time-independent vector fields.

10.1 Time-Varying Vector Fields

In this section, I review the important concepts of time-varying vector fields for the design tasks.

10.1.1 Definition

Consider a manifold M and a subset $X \subset M$. The boundary of X is denoted by ∂X and closure by $\text{cl}(X)$. A time-varying vector field can be defined as follows.

Definition 10.1.1 *A* time-varying vector field *can be expressed in terms of a partial differential equation* $\dot{x} = V(x;t)$ *where* $x \in X$ *and* $t \in \mathbf{R}$. *Given* $t_c \in \mathbf{R}$, $V(x;t_c)$ *is a time-independent which is referred to as an* instantaneous vector field.

Remark: Time-independent vector fields is clearly a special case of time-varying vector fields such that $V(x;t) = V(x;s)$ for any $x \in X$ and all $t, s \in \mathbf{R}$.

10.1.2 Integral Curves in Time-Varying Vector Fields

Given the definition of a time-varying vector field, I now review a number of important curves which are the integral solutions under different initial conditions and integrand.

Definition 10.1.2 *Given $x \in M$ at time $t_0 \in R$, its* trajectory *in a time-varying vector field $\dot{x} = V(x;t)$ is $\cup_{t\in\mathbf{R}} x(t)$ where*

$$x(t) = x(t_0) + \int_0^{t-t_0} V(x(s); t_0 + s)ds \ .$$

It is also referred to as pathline.

Definition 10.1.3 *If V is time-independent, its trajectory through $x \in M$ is of the following form*

$$x(t) = x(t_0) + \int_0^{t-t_0} V(x(s); t_0)ds$$

which I referred to as streamline *(Section 2.2).*

Remark: [75] provides an example illustrating the difference between the streamlines and pathlines of the same time-varying vector field. Comparison and definitions of these two concepts shows that streamlines depict the instantaneous appearance of a time-varying vector field, while pathlines convey the real paths of the particles over time. Therefore, streamlines can be a design primitive to achieve desired instantaneous appearance in the orientation field design. Similarly, pathlines can be applied to control the moving paths of specific particles for the advection field design.

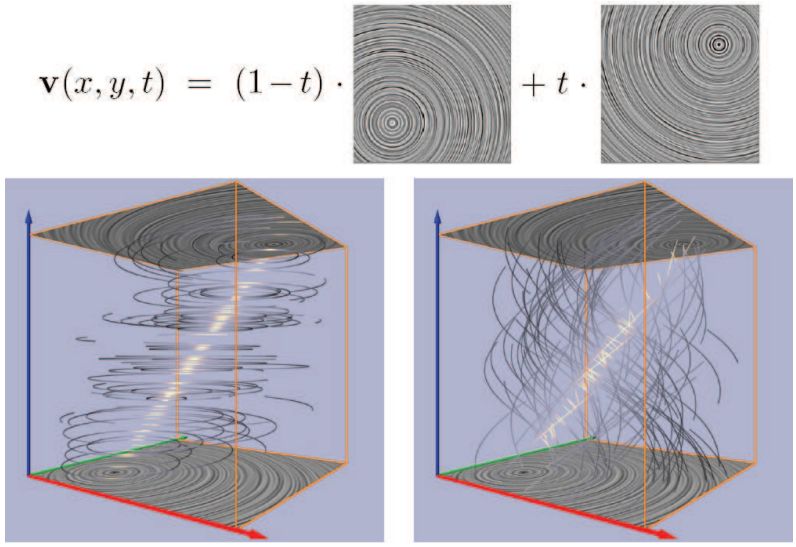


Figure 10.1: This figure demonstrates the difference between streamlines (left) and pathlines (right) [75].

In the later presented design system, streamlines and pathlines are the main curve primitives I will employ to accomplish the design tasks. However, there are other important feature descriptors that can help understand the time-dependent dynamics.

Definition 1 *Given $x \in M$, a* streakline *is defined to be the curve traced out by particles injected at x over time. It is computed as follows.*

$$x_{t_0}(t) = x(t_0) + \int_0^{t-t_0} V(x_{t_0}(s); t_0 - t + s)ds \ .$$

Remark: Different from pathlines who considers the trajectory of a single particle, a streakline consists of the current positions of multiple particles released at different time step but same location. If we are able to record the trajectories

of all particles of a streakline, we can produce a set of pathlines with different life span.

Definition 2 *Given $t \in \mathbf{R}$, a subset $X \in M$. The set of trajectories traced by particles injected at each point of X at t is called the* timeline *at t.*

Remark: Different from streaklines, we are tracking particles that are release at the same time but different locations. Similarly, from timeline, we can resemble a set of pathlines with same life span.

10.1.3 Instantaneous Topology

In this section, I briefly review the basic concepts of instantaneous topology which is equivalent to time-independent vector field topology (Section 2.2). I consider the instantaneous vector field $V(x;t_c)$ at time t_c. A point $\mathbf{p}$ is called a *singularity* at t_c if $V(\mathbf{p};t_c) = 0$. Note that from this chapter I will use the conventional notion for singularities instead of *fixed points* which I have used to define and analyze this vector field characteristic. According to the Jacobian analysis of a linearization of $V(\mathbf{p};t_c)$, a singularity can be classified as sources, sinks, and saddles. The streamlines of $V(x;t_c)$ that connect at least one saddle are referred to as separatrices. The closed streamlines that partition the vector field domain are called periodic orbits. The instantaneous topology of $V(x;t_c)$ is defined as the topological graph of the instantaneous field $V(x;t_c)$. It consists of singularities, periodic orbits, and their connectivity (i.e. ECG) (Section 2.2.2) and conveys the qualitative structure of the vector field at t_c. It should be pointed out that if all

singularities and periodic orbits in an ECG are hyperbolic, the ECG is structurally stable. That is, the ECG remains unchanged under small perturbation with certain bound [78]. I have not identified graphics applications in which design and control of periodic orbits is important. Consequently, I will focus on singularities and singularity-related bifurcations in this paper.

10.1.4 Bifurcation

Given the instantaneous vector field topology of $V(x;t)$, we are able to keep track of the evolution of it with respect to t by extracting instantaneous topological graph at each discrete step t_j and matching singularities and other features. This technique has been adapted to analyze time-varying flow datasets [82]. The movement of a singularity in the space-time domain gives rise to a curve which is referred to as a *singularity path*. Singularity characteristics, such as Jacobian property, can also be advected along the path. Therefore, the Jacobian can be computed at any time t given a continuous path and can be used to determine the local field structures near the singularity at t. Two singularity paths can intersect in domain $M \times \mathbf{R}$ only if they have opposite *poincaré indices*, for instance, the paths of a source and a saddle, or a sink and a saddle, respectively. Note that I have excluded higher-order singularities in my discussion. At the intersections, no hyperbolic singularities exist. Thus, these intersections indicate certain qualitative changes (i.e. the number and types of singularities vary, which results in the changes of the topological graph). We refer to these structural changes as *bifurcations*, and

the positions $(x; t_0)$ where these happen as *bifurcation points*. Figure 10.2 provides an illustration of such process in a saddle-source bifurcation. More comprehensive introduction of the bifurcation theory can be found in [26]. Bifurcation points for saddle-node bifurcations can be extracted [82].

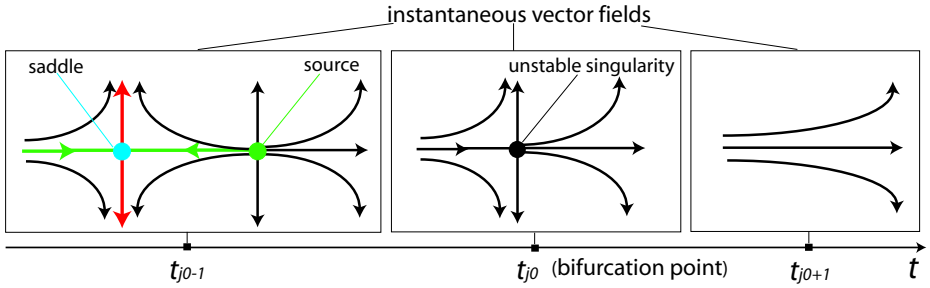


Figure 10.2: This example demonstrates a saddle-node bifurcation, i.e. a source-saddle cancellation. The directional curves illustrates the flow behaviors. Two singularities are shown in the left at t_{j_0-1}. They move toward each other when t evolves and collide at t_{j_0} (middle). The two singularities are cancelled after they meet, which results in a singularity-free vector field at t_{j_0+1} (right).

10.2 2D Parameterized Vector Field

To overcome the representation of time-varying vector fields and enable its design process, I now introduce the concept of *parameterized vector field*. Consider a 2-manifold M.

Definition 10.2.1 *A* parameterized vector field *is defined as a series of instantaneous vector fields with respect to a parameter* λ*, denoted by* $V(\mathbf{x}; \lambda)$*, where* $\mathbf{x} \in M$ *and* $\lambda \in \mathbf{R}$*. An instantaneous field of* V *at* λ_c *is denoted by* $V(\mathbf{x}; \lambda_c)$*.*

There is considerable freedom to move from the parameterized family of vector fields $V(\mathbf{x};\lambda)$ to a time-varying system. For example, let $g : \mathbf{R} \to \mathbf{R}$ be any smooth positive function. Then

$$\frac{d\mathbf{x}}{dt} = V(\mathbf{x},\lambda), \quad \frac{d\lambda}{dt} = g(\lambda)$$

is a time-varying system. As a first approximation the reader can assume that we are using the time-varying system $\frac{d\mathbf{x}}{dt} = V(\mathbf{x};\lambda), \quad \frac{d\lambda}{dt} = s$ where s is a positive constant. In other words, parameter λ and physical time t possess a linear relation such that $d\lambda = s \cdot dt$. Therefore, if $s = 1$, λ is equivalent to t. Further, it shows when $s \to 0$, the variation of the vector field is sufficiently small which guarantees a smooth transition. This is particularly useful for graphics applications where an input field with smooth transition over time is expected to achieve visually coherent results. It also reveals the relation between a time-varying vector field and a corresponding parameterized vector field which enables the discussion of certain concepts under the parameterized vector field framework.

Note that the rest of the thesis will discuss the design of the parameterized vector fields without further clarification.

Chapter 11 – Time-Varying Vector Field Design On Surfaces

I conduct the design of the parameterized vector fields, an approximation of time-varing vector fields using a three-stage pipeline. Figure 11.1 provides an illustrative diagram for this pipeline. In the first stage, the design system supports the creation of a parameterized vector field through a number of types of user specifications, such as *singularities*, *streamlines*, *singularity paths*, *pathlines*, and *bifurcations*. These specifications are either converted into basis fields and summed or treated as constraints of a relaxation process (Section 11.1). In the second step, the system analyzes the initial field and provides necessary feedback to the user for further editing. To enable control over unwanted flow behaviors such as singularities and bifurcations, I provide topological editing functionalities to remove them or move them to more desirable locations in spacetime in the third stage (Section 11.2). The techniques employed in vector field analysis (stage 2) is based on previous research. Thus, I will only describe the details of the initialization (stage 1) and editing (stage 3) of the design system. I demonstrate the utility of the proposed design system through the application of texture synthesis and animation.

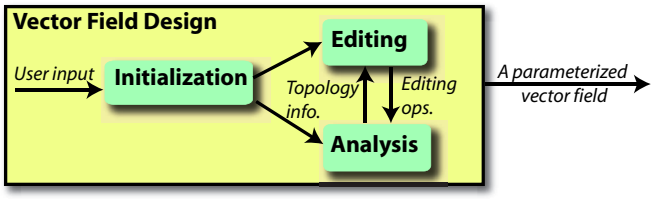


Figure 11.1: The design pipeline.

11.1 Initialization

In this section, I discuss two approaches that I employ to generate a parameterized vector field. First, instantaneous vector fields are created in key frames and propagated to the rest of the field. The advantage of this approach is that we can reuse past techniques in designing instantaneous (steady) vector fields [8, 21, 101]. However, this approach does not address features unique to time-varying vector fields such as bifurcations and pathlines. In the second approach, such features can be generated through the extended basis vector fields or constrained optimization in a *spatio-parameterized* domain.

11.1.1 Setting

Computation Domain: Given the definition of a parameterized vector field on a 2-manifold, I define the spatio-parameterized domain as $\mathbf{D} = M \times \mathbf{R}$. In the implementation, I am concerned with a sub-domain $\mathbf{X} \subset \mathbf{D}$ such that $\mathbf{X} = (X; \lambda)$ where X is a triangulation of a 3D surface, and $\lambda \in [0, 1]$ is a parameter that I use to approximate time (see the inlet

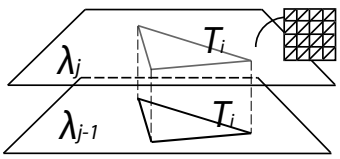


figure). For representing and storing the field, I discretize λ evenly. I denote these discretely sampled λ values as $\{\lambda_j\}$. A typical number for the discretization is 100 for the examples in this thesis. I then compute and store the instantaneous fields at these discrete $\{\lambda_j\}$ in order.

Interpolation scheme: I resort to the interpolation scheme that has been successfully applied by Zhang et al. [101], Palacios and Zhang [50], and Chen et al. [8] for continuous surface flow construction in the spatial subspace. Over $\mathbf{X}$, The similar interpolation technique proposed by Tricoche et al. [82] is employed to guarantee a linear field along the parameter λ dimension. Particularly, in planar case this configuration can be formulated as follows.

$$V(\mathbf{x};\lambda) = \mathbf{a}(\lambda)x + \mathbf{b}(\lambda)y + \mathbf{c}(\lambda) \quad (\mathbf{x};\ \lambda) \in X \subset \mathbf{D}$$

where $\mathbf{a} = (a_x, a_y)$ and $\mathbf{a}(\lambda) = \frac{\lambda_{j+1}-\lambda}{\lambda_{j+1}-\lambda_j}\mathbf{a}_{\lambda_j} + \frac{\lambda-\lambda_j}{\lambda_{j+1}-\lambda_j}\mathbf{a}_{\lambda_{j+1}}$, where $\mathbf{a}_{\lambda_j}$ and $\mathbf{a}_{\lambda_{j+1}}$ are the coefficient of the linearaztion of the vector field at λ_j and λ_{j+1}, respectively. $\mathbf{b}(\lambda)$ and $\mathbf{c}(\lambda)$ are similarly defined.

11.1.2 Designing Instantaneous Fields

I start with a review of the steady vector field creation using basis fields (Section 5.2.1) and constrained optimization (Section 4.2). To design an instantaneous field, the user can either specify the singular or regular design elements (locally) at desired locations [86, 101] or provide a set of streamlines indicating the flow

directions along the streamline and in nearby regions. The provided streamlines are eventually sampled and converted into polylines which are used to construct the regular elements. A regular element is an arrow pointing from a basis point to a certain direction. This idea has been employed by Chen et al. [10] to design street networks that follow the boundaries of natural features such as rivers. Each design element is associated with a Jacobian J_i, and gives rise to a basis field which I use to compute a weighted sum to obtain the global field. Equation 11.1 defines such a weighted sum.

$$V(\mathbf{x}) = \sum_i e^{-d\|\mathbf{x}-\mathbf{p}_i\|^2} J_i \tag{11.1}$$

where d is a decay constant, $\mathbf{x}$ is a point in space, J_i is the Jacobian corresponding to a design element, and $\mathbf{p_i}$ is the position of the design element in space.

The radial basis field approach (equation 11.1) cannot be applied to the design on surfaces without a global parameterization of the surface. Consequently, I resort to the *constrained optimization*, or *relaxation* to design a surface field, which possesses the following form:

$$\overline{V}(v_i) = \sum_{j \in J} \omega_{ij} \overline{V}(v_j) \tag{11.2}$$

where v_i is an interior vertex of a triangle region N, v_j's are the adjacent vertices of v_i that are either in the interior or on the boundary of N. $\overline{V}(v_i)$ represents the average vector value at vertex v_i.

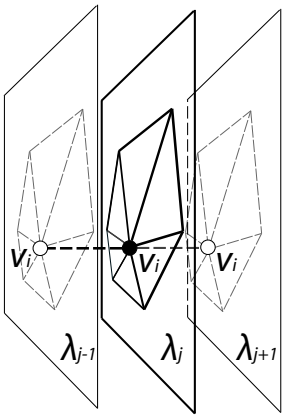
11.1.3 Designing Parameterized Vector Fields

I now describe the techniques that are used to produce a parameterized vector field. A natural idea is to set the designed instantaneous fields as *key frames* and derive a parameterized vector field from them.

11.1.3.1 Key Frame Design

In field design using key frames, the user first designs a sparse set of instantaneous fields that indicate the desired effects at specific frames (λ 's). The system then generates a parameterized vector field achieving these effects using an *extended constrained optimization*.

Extended Constrained Optimization: By taking into account the additional parameter λ, I introduce an extended constrained optimization technique to create parameterized vector fields on surfaces. Given a vertex $(v_i; \lambda_j)$ in the underlying mesh in domain $\mathbf{X}$, I consider a stencil of it shown in the figure to the right. In this stencil configuration, I assume there are (virtual) edges between $(v_i; \lambda_j)$ and $(v_i; \lambda_{j-1})$, and $(v_i; \lambda_j)$ and $(v_i; \lambda_{j+1})$, respectively. Therefore, the computation of discrete Laplacian under this setting needs to consider $(v_i; \lambda_{j-1})$ and $(v_i; \lambda_{j+1})$ as the direct neighbors of $(v_i; \lambda_j)$. The spatio-temporal dis-



crete Laplace can be expressed as follows:

$$\begin{aligned}\omega\overline{V}(v_i;\lambda_j) = \quad & \textstyle\sum_{l\in N(i)} \omega_{j,l}\overline{V}(v_l;\lambda_j) + \omega_{j,j-1}\overline{V}(v_i;\lambda_{j-1}) \\ & +\omega_{j,j+1}\overline{V}(v_i;\lambda_{j+1}) \end{aligned} \tag{11.3}$$

where $N(i)$ denotes the one-ring neighbors of $(v_i;\lambda_j)$, $\overline{V}(v_i;\lambda_j)$ represents the average vector value at position $(v_i;\lambda_j)$. $\omega_{j,j-1}$ and $\omega_{j,j+1}$ are positive weights determining how fast the relaxation process is. In the implementation $\omega_{j,j-1}$=$\omega_{j,j+1} = 10$. $\omega = \sum_{l\in N(i)} \omega_{j,l} + \omega_{j,j-1} + \omega_{j,j+1}$ is the normalization coefficient. I point out that this formula can be further extended by taking into account more sampled steps along λ axis to achieve smoother results as bi-Laplace smoothing does in time-independent case [21]. It is worth noting that similar spatio-temporal relaxation process has also been applied in the space-time surface reconstruction by Sharf et al. [64] where regular grid configuration is considered instead of irregular one used in this work.

Figure 11.2 shows a planar field generated using key frame design and the extended constrained optimization. Many surface fields (Figures 9.2, 11.8, 11.9, and 11.10) shown in this paper are also generated using this method.

It should be pointed out that other interpolation scheme can be employed to obtain a parameterized vector field from a set of instantaneous field as well, such as vector linear interpolation. This typically does not produce smooth results due to the potential degenerate vectors and discontinuity (Figure 11.3).

For planar field design, an *extended basis field approach* can also be applied to

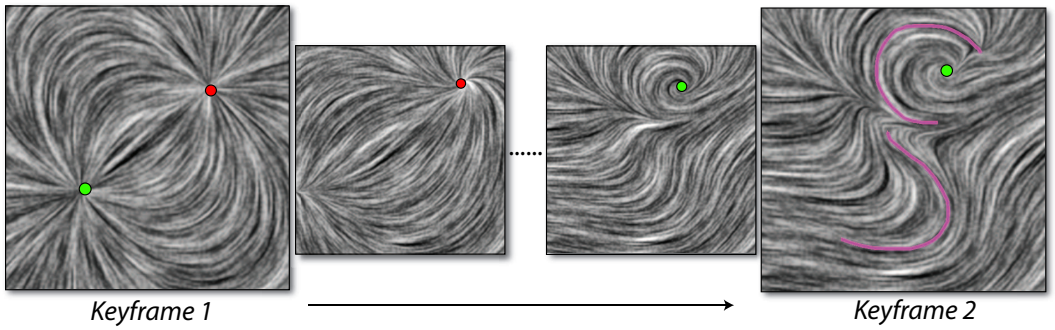


Figure 11.2: An key frame design example. The purple curves are the user specified streamlines. The flow-like textures shown in the paper are generated using IBFV(S) techniques of van Wijk [86, 87].

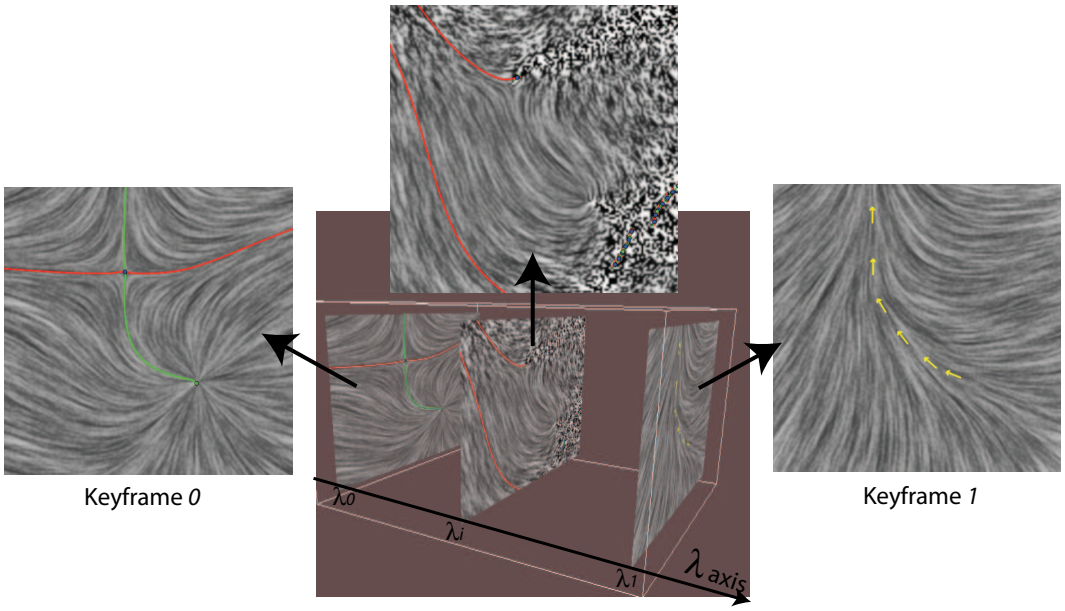


Figure 11.3: A time-varying vector field produced using simple linear interpolation from the specified key frames. Note that the discontinuity appear in the middle instantaneous field.

generate a parameterized vector field from the user specifications (singularities and streamlines) directly.

Extended Basis Field Approach: Similar to the instantaneous field design, we have the concepts of design elements with the additional parameter λ being considered. That is, the position of a singular or regular element in the computation domain $\mathbf{D}$ has the form of $(\mathbf{p}; \lambda)$. For instance, if the user inserts a singularity in the spatial position $\mathbf{p_i}$ at λ_i, its position in $\mathbf{D}$ is $(\mathbf{p_i}; \lambda_i)$. Similarly, all the regular elements stemming from a streamline defined at λ_i will be assigned this parameter value λ_i. Accordingly, the generated basis field will affect not only a single instantaneous field, but also the parameterized vector fields with the instantaneous field at λ_i as center. I update the basis field summation equation as follows:

$$V_I(\mathbf{x}; \lambda) = \sum_i e^{-b\|\lambda-\lambda_i\|^2} e^{-d\|\mathbf{x}-\mathbf{x}_i\|^2} J_i \tag{11.4}$$

where b is a decay constant along λ axis, the ratio of b/d reflects the ratio of the propagation speeds over the spatial and parameter spaces. In the experiments, I make use of $b/d = 10$ without considering particular physical constraints.

Further, a **brush streamline interface** inspired by the *brush interface* by my previous work (not included in this thesis) for tensor field design [10] is used for instantaneous vector field design. More specifically, the user sketches a curve. A local region with the curve as the skeleton is found [63]. The vector field inside the region is computed according to the derived regular elements from the curve. As

pointed out by Chen et al., the brush interface can easily introduce large variations along boundaries of the brush region which may be interesting to segmented texture synthesis. Figure 11.4 shows three examples using brush streamlines.

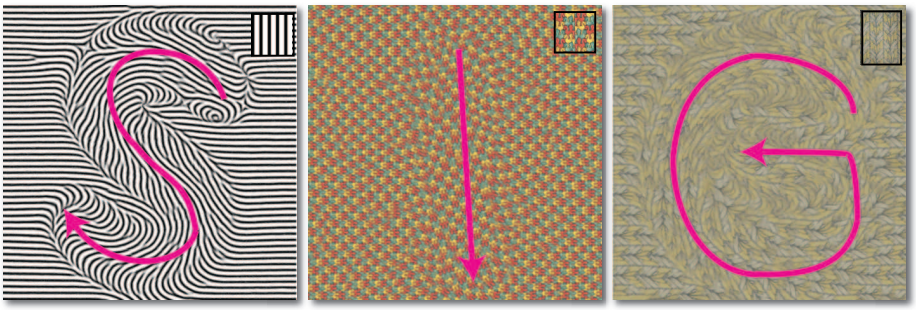
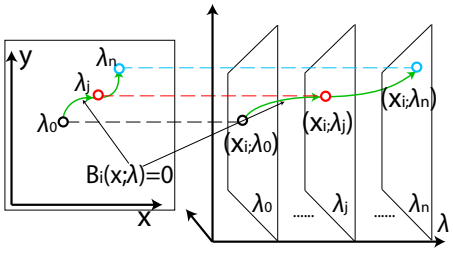


Figure 11.4: This figure provides some results of brush stroke design. They are the sampled frames from the accompany animations.

11.1.3.2 Singularity Path Design

In many cases, we want to animate the moving of certain *singular patterns* over surfaces, for instance, the moving of a storm system in environment modeling (Figure 11.8).



The design system supports this by allowing the user to specify the paths of the singular design elements along positive λ direction. The negative direction is similar. To do so, the user first specifies the path of s singular design element on the surface in the spatial domain. The parameter (λ) information associated with the path is then provided by the user,

including the parameter value λ_s at the start point and the value λ_e at the end point, or the parameter values corresponding to the discrete sample points along the paths if provided (see the hollow circles in the inlet figure). I denote the singularity path of i^{th} singular element as $B_i(\mathbf{p}_i;\ \lambda) = 0$ ($\lambda \in [\lambda_s, \lambda_e]$) . That is, given $\lambda_j \in [\lambda_s, \lambda_e]$), the position of this singular element $(\mathbf{p}_{ij}; \lambda_j)$ satisfies $B_i(\mathbf{p}_{ij};\ \lambda_j) = 0$. For simplicity, I evenly sample this specified path to obtain the position for the singular element corresponding to each λ_j. The system then induces a parameterized vector field by computing the positions of these singular elements along their paths at each sampled parameter value and summing up all the basis fields as follows.

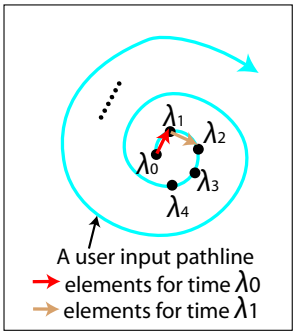
$$V_{S_p}(\mathbf{x};\ \lambda) = \sum_i V_i(\mathbf{p}_i;\ \lambda)|_{B_i(\mathbf{p}_i;\ \lambda)=0} \tag{11.5}$$

where V_{S_p} denotes the basis fields generated by the singularities that are currently at λ. $V_i(\mathbf{p}_i; \lambda) = e^{-d\|\mathbf{x}-\mathbf{p_i}\|^2} J_i$ is the basis field stemming from the i^{th} singularity at λ and $\lambda \in [\lambda_s, \lambda_e]$). The figure above provides an illustrative example of singularity path design.

Singularity path design on surfaces is handled differently compared to the design on plane. Due to the lack of a global parameterization, I resort to the constrained optimization to generate the individual instantaneous fields at the desired sampled λ_j.

11.1.3.3 Pathline Design

It is often necessary to specify trajectories of a particle according to a time-varying vector field. The trajectory, a pathline, can be designed using our system as follows. Note that a pathline is different from a singularity path despite the similar design mechanisms in our system for both. To create a field from a pathline, the user first specifies a curve to infer the desired pathline. The system then induces a parameterized vector field based on the sampled positions along the pathline. This technique is based on the following observation. Given a pathline P and a point $\mathbf{p}_j$ on it at time λ_j, it was advected from previous position on P, $\mathbf{p}_{j-1}$. If $\mathbf{p}_{j-1}$ and $\mathbf{p}_j$ are sufficiently close, the vector pointing from $\mathbf{p}_{j-1}$ to $\mathbf{p}_j$ approximates the true vector value at $(\mathbf{p}_{j-1}; \lambda_{j-1})$. We then can set this vector to be a regular element at λ_{j-1}. In the design system, I offer the user a similar interface to the streamline design for sketching the desired pathline. But during the discretization process, each sampled point will be assigned a unique λ value according to the parameter information associated with the corresponding pathlines. Assume the start value λ_s and end value λ_e are known. Then, for the i^{th} sampled point (out of n samples), its assigned λ value is computed as $\lambda_s + i \times (\lambda_e - \lambda_s)/(n-1)$. Figure to the right provides an illustrative example of pathline based design. Note that the i^{th} regular element stemmed from line segment $(i, i+1)$ is located at λ_i. The extended basis field approach can be adapted to induce a parameter-



ized vector field for the planar case (equation 11.4), while a constrained optimization process is needed for the generation of the individual instantaneous fields on surfaces.

11.1.3.4 Jacobian-Based Design

Matrix-based Design is also provided by our system for the user to determine how the vector field transforms (rotates, scales, stretches) over λ. The matrix (field) has the form of $M(\lambda) = \begin{pmatrix} M_{11}(\lambda) & M_{12}(\lambda) \\ M_{21}(\lambda) & M_{22}(\lambda) \end{pmatrix}$, where $M_{ij}(\lambda)$ are functions of λ. In the implementation, I require the user to provide an affine transformation matrix (i.e. the combination of 2D rotation and scaling) to act on the initial instantaneous field (at λ_s) to obtain the last field (at λ_e). The system then interpolates the rotation (rotation angles) and scaling (scaling factors). The vector field V at λ_j is computed as $V(\lambda_j) = M(\lambda_j)V(\lambda_{j-1})$, where $M(\lambda_j)$ is the combination of the rotation and scaling matrices at time λ_j. Figure to the right provides the texture synthesis results guided by an orientation field generated using matrix-based approach where the field is rotated $w.r.t.$ λ.

11.1.4 Bifurcation Design

As a significant and novel contribution, our system allows the user to insert a bifurcation at specific location in the spatio-parameterized domain. This is particually useful for the applications where the split or merge of the graphical primitives are desired. Figures 11.8 and 11.9 provide examples of desirable bifurcations in the texture synthesis and animation. Recall that we are only concerned with saddle-node bifurcations in this thesis. Equation 11.6 provides a formula that is used to create a saddle-node bifurcation (saddle and node pair creation) at position $(x, y; 0)$ in X (see Figure 11.5).

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \lambda - x^2 \\ -y \end{pmatrix} \tag{11.6}$$

Similarly, we can enforce a saddle and source pair cancellation at position $(x, y; 0)$ in X using equation 11.7.

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \lambda + x^2 \\ y \end{pmatrix} \tag{11.7}$$

In addition, we can scale the range of the bifurcation in both space and time, and re-orient the axis (a straight line in this case) to control where and how the bifurcation happens.

Other types of bifurcations [26] can be created in the similar manner. The global field induced from a set of bifurcations can be computed as the weighted

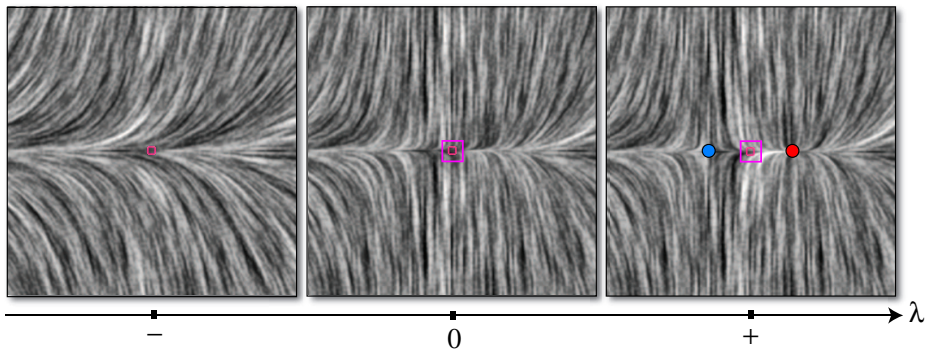


Figure 11.5: A saddle sink creation bifurcation happens at $(0.5, 0.5; 0.5)$ in the spatio-parameterized domain X using equation 11.6.

sum of the individual bifurcations (equation 11.8). In other words, each bifurcation is a design element.

$$V_B(\mathbf{x};\ \lambda) = \sum_i e^{-d\|\mathbf{x}-\mathbf{x_i}\|^2} V_i(\mathbf{x};\ \lambda - \lambda_i) \tag{11.8}$$

where $(\mathbf{x_i}; \lambda_i)$ is the position at which the i^{th} bifurcation happens. Note that the decay effect along λ axis is considered by the formula $V_i(\mathbf{x};\ \lambda - \lambda_i)$ (see equations 11.6 and 11.7). Accordingly, I define the global field as the weighted sum of the basis fields generated using singular and regular elements V_I (equation 11.4), and bifurcations V_B (equation 11.8).

$$V(\mathbf{x};\ \lambda) = \omega_B V_B(\mathbf{x};\ \lambda) + \omega_I V_I(\mathbf{x};\ \lambda) \tag{11.9}$$

where ω_I and ω_B are positive values which I use 0.5 for both in the implementation.

There are two alternative approaches of inserting bifurcations into the designed field. First, the user can generate a bifurcation through key frame design. Basi-

cally, the user sets two instantaneous fields as key frames before and after the λ value where the bifurcation point is desired. One of these fields is trivial (similar to the left field shown in Figure 11.5), while the other contains the saddle and source (or sink) singularity pair (the right field shown in Figure 11.5). Then, a bifurcation is enforced to occur by the extended constrained optimization. The second approach allows the user to design the split of a singularity path to indicate a saddle-node creation bifurcation or intersect the end points of two singularity paths to induce a saddle-node cancellation bifurcation. Both approaches were applied to insert bifurcations into the designed fields in the provided examples (Figures 11.8 and 11.9).

11.2 Editing

Editing functionality is required for a design system because of the appearance of undesired features such as singularities and bifurcations in the initialization phase. The design system provides the user with a number of options to edit a given parameterized vector field. First, the conventional editing operations for instantaneous vector field modification are provided. Second, the novel bifurcation removal and movement are introduced along with a general smoothing scheme in the spatio-parameterized domain.

11.2.1 Instantaneous Field Editing

First, the system extracts the instantaneous vector field topology. Then, the user can cancel two unwanted singularities at a particular λ value using the simplification techniques proposed in Chapter 4 [8]. This instantaneous field is then considered as a key frame for the regeneration of the field. Note that this editing process may potentially introduce more complex dynamics such as unintended bifurcations due to the weak constraint along the parameter (λ) axis. I relieve this by adding the constraint of maximum propagation distance along λ. That is, after modifying a certain key frame how far (in length) the user wishes the changes to affect the rest of the field along the λ axis.

11.2.2 Bifurcation Editing

I have demonstrated the relations between saddle-node bifurcations and the structural changes in texture animations. I now develop techniques to control them. To do so, we need to first know where the bifurcations occur. In my implementation, I keep track of singularities and extract bifurcations from the designed fields using the techniques proposed by Tricoche et al. [82].

Bifurcation Removal: The system allows the user to remove a bifurcation if the involving singularities do not participate in other bifurcations. I refer to these bifurcations as *isolated* bifurcations. If a bifurcation is not isolated, we can not cancel it without affecting other features. To remove an isolated bifurcation, I keep track of the involving singularities along the λ axis until their birth (saddle-node

cancellation) or their death (saddle-node creation). I assume the λ value of their birth (or death) is λ_c. Then, cancelling these singularities at λ_c will induce the removal of the corresponding bifurcation. Under our setting of $\mathbf{X}$, only boundary singularities (i.e. singularities at $\lambda = 0$ and 1 of X) will satisfy this requirement. Figure 11.6 shows an example of saddle-node bifurcation removal. More complex local control of connected bifurcations is possible which is beyond the scope of this thesis. Note that this operation is not valid in a real time-dependent vector field where the range of the physical time is infinite. In that setting, the more meaningful operation is *bifurcation movement*.

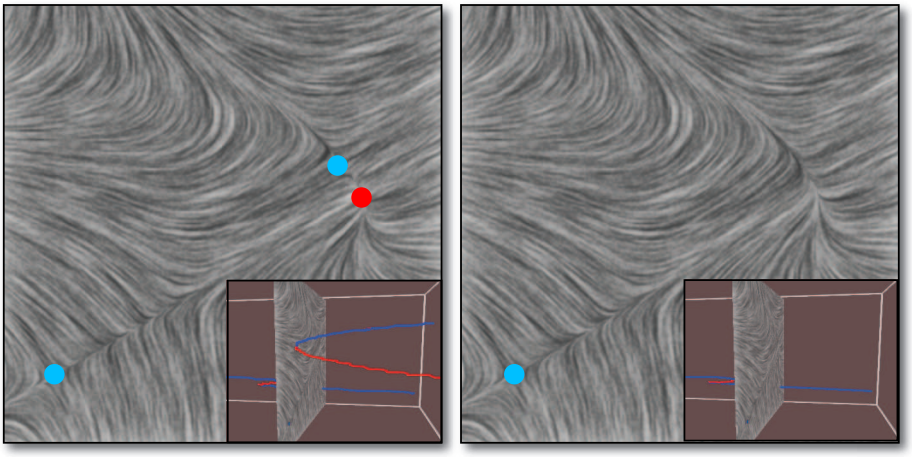


Figure 11.6: Example of bifurcation editing. Left column shows the effect before editing; right column shows the results after bifurcation removal.

Bifurcation Movement: Similar to the singularity movement functionality, a bifurcation can be moved. Moving bifurcation can be achieved by moving the involving singularities over space at particular λ value. The edited instantaneous field is then set as a key frame. The extended constrained optimization will smooth

the rest of the field. Note that the movement of these involving singularities should obey the topological constraints proposed by Zhang et al. [101] in their steady vector field design tool. This guarantees no other topological features will be affected during the movement.

General Smoothing: The two aforementioned bifurcation control techniques are typically too constrained for the design fields. I then introduce a more relaxed editing functionality to allow the user to modify the designed fields without concerning with the topological constraints. I refer to this technqiue as *general smoothing*.

General Smoothing is a spatio-parameterized smoothing in which the user defines a box in $\mathbf{X}$. The vector values at the inner vertices of the box are replaced with a hopefully smoother version computed using the extended constrained optimization (equation 11.3), with the boundary vertices as the constraints.

11.3 Application: Texture Synthesis and Animation

I have applied the designed parameterized orientation fields and advection fields generated using our techniques to create a number of synthetic texture animations (Figures 9.2, 9.1, 11.8, 11.4, 11.10, and 11.9). Flow-guided texture synthesis and advection has been introduced to visualization community for dense flow visualization by van Wijk [86, 87], Laramee et al. [41], and Neyret [48]. Kwatra et al. [38] present an optimization-based plane texture synthesis which can be used for flow-guided texture animation. Lefebvre and Hoppe's [44] introduce an

appearance-space texture synthesis technique that can handle texture advection over static surfaces. Later, Kwatra et al. [37] and Bargteil et al. [5] extend the advected texturing techniques onto the problem of fluid texturing on surfaces, respectively. In addition, Wiebe and Houston [96] and Rasmussen et al. [56] perform fluid texturing by advecting texture coordinates along the flow field using level sets and particles. In this work, I employ Kwatra et al.'s [37] texturing fluid techniques for the presented texture synthesis and animation examples.

Many applications may want the animated texture on surfaces, such as special effects, games, and digital arts. In addition, with the proper choice of texture exemplars and careful field design, other graphical effects can be resembled through advected textures, such as the ripple-like advection, the time-varying caustic reflection and the lava effect (Figure 11.7).

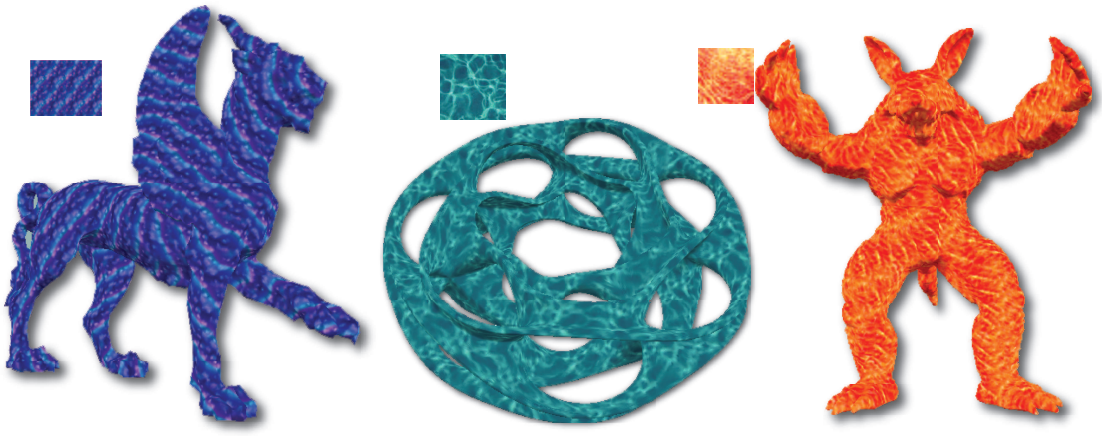


Figure 11.7: Different effects obtained using texture synthesis and animations: ripple advection (left), caustic reflection (middle), and the lava effect (right). All the texture synthesis and animations are driven by the time-varying vector fields created using our system.

Performance: The intialization of a planar field with 100 frames defined on a 65×65 regular grid typically takes less than 5 seconds on a 3 GHz PC with 1GB RAM. For the design on surface (up to $20,000$ vertices), it can take up to 4 minutes to generate the field with 100 frames without optimization. The times spent on synthesis vary from 8 hours to 20 hours on a 3.6 GHz PC with 2GB RAM depending on the number of sample points being put on the surfaces, the volume sizes of the models, and the sizes of the input texture exemplars [37].

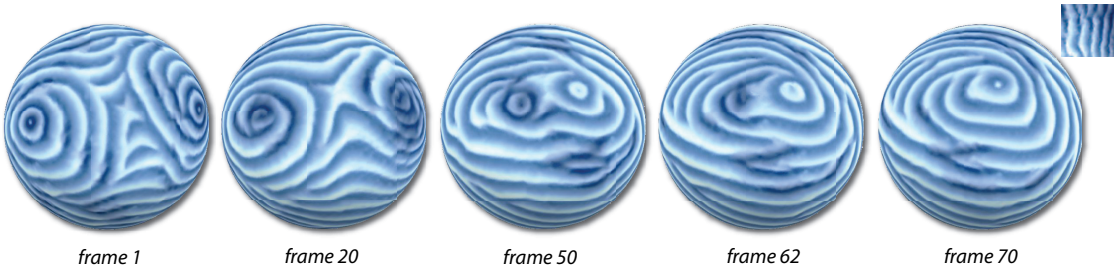


Figure 11.8: This image shows a number of frames from a texture animation on sphere which simulates the collision of two storm systems. The animation is driven by an orientation field and an advection field, both are designed using the techniques introduced in this paper. Note that two vortex-like patterns (frame 1) are displaced by the advection field at the middle of the squence (frames 20 and 50). The two storms are then merged (frame 62) and become one system (frame 70).

Figure 11.9: This image shows a number of frames from a texture animation on venus. The animation is driven by an orientation field and an advection field, both are designed using the techniques introduced in this paper. Note that a vortex-like pattern (frame 1) is displaced by the advection field at the middle of the model (frames 3 and 20). The vortex is then splitted into two (frame 70), and both of them continue moving upwards to the upper middle along the model (frame 100).

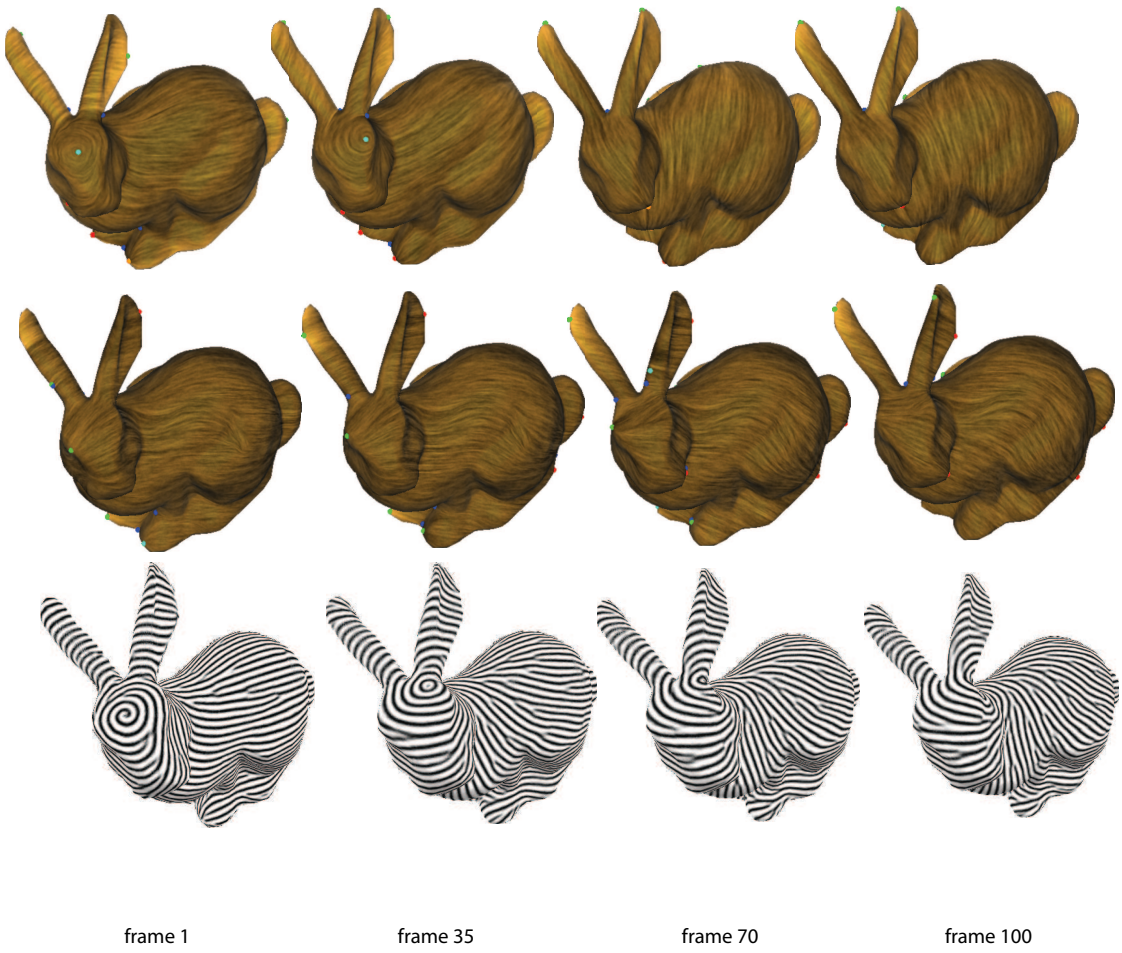


Figure 11.10: The designed results of an orientation field (first row) and and advection field (second row) on bunny. The sampled frames from the corresponding texture synthesis and animation results are provided below the fields. Particularly, the third row shows the advection of the orientation field of the first frame.

Chapter 12 – Conclusion and Future Work

Vector fields have become one of the major research subjects in both visualization and computer graphics communities. In engineering and science, vector fields are used to represent the simulation and experiment results for the evaluation of certain dynamical systems. In addition, vector fields are used to drive or orient certain graphical primitive and subjects for a wide range of important applications. In this work, I investigate the use of vector fields for both engineering and computer graphics applications. I summarize my contributions in this chapter.

12.1 Vector Field Analysis

Vector field topology plays an important role in vector field analysis. It provides the qualitative (or structural) information of the vector field of interest, which helps the domain experts identify the critical features and behaviors efficiently. Therefore, it has obtained a great deal of attention since 1989. However, a number of challenges remain unsolved, such as the completeness and stability. I have addressed both problems in this work which I summarize as follows.

12.1.1 Summary

First, I have identified periodic orbits as an important component of the topological graph (e.g. vector field topological skeleton) and proposed a more complete vector field topology called ECG with periodic orbits being included. With emphasis on periodic orbits, I introduce an efficient algorithm to extract periodic orbits from 2D flows. A robust algorithm for computing the ECG from the detected periodic orbits and fixed points is presented. With the assistant of ECG, I elaborate a number of basic scenarios of pairwise cancellation of features in ECG, which are applied to drive the vector field simplification with topological guarantee. The analysis and simplification results are visualized using the proposed improved evenly-spaced streamline placement technique with all separating features being highlighted. In addition, I introduce two approaches to create periodic orbits in 2D vector fields. To my best knowledge, this presented work is the first to address the design, analysis, and control of periodic orbits in surface flows.

Second, I analyze the instability of trajectory-based vector field topology including the proposed ECG and advocate the use of the Morse decomposition of a vector field as a more reliable computation for vector field topology analysis. MCG, the result of the Morse decomposition of a given vector field, is defined as a more stable vector field topological graph than ECG. With the focus on MCG computation, I first describe the pipeline of a Morse decomposition of a vector field. The key step of this pipeline is a directed graph construction process called flow combinatorialization. The obtained graph encodes the flow dynamics whose strongly

connected components correspond to the regions with critical (or recurrent) flow behaviors. To compute an accurate directed graph in flow combinatorialization, I introduce the idea of $\tau-$map which relies on tracing the image of each polygonal primitive of the flow domain according to the flow dynamics. The results show fine decompositions (i.e. detailed MCGs) of vector fields. Despite the desirable output of fine decompositions, the global $\tau-$map approach suffers from the problem of slow computation due to the demanding tracing process. To overcome that, I depict a hierarchical refinement framework which localized the $\tau-$map computation within the Morse neighborhoods. Each iteration, the program automatically selects a Morse set to refine where it is recognized that more complex flow is contained. This improvement greatly reduces the computation cost and leads to faster analysis. I should point out that the stable analysis of vector fields is one of the goals of vector field visualization area that was not resolved before. The presented work is the first to provide a practical solution to achieve such a goal.

12.1.2 Future Directions

The presented work also points out a number of future directions. First, the periodic orbit detection method depends on efficient extraction of separation and attachment points. While I have observed in the experiments that these points tend to be close to periodic orbits, a rigorous mathematical study on the subject is needed. Furthermore, other methods for extracting separation and attachment points, such as that of Peikert and Roth [52], may lead to more numerically

stable results. Second, current MCG and ECG construction methods assume closed surfaces. I plan to investigate means to extend them to handle surfaces with boundaries. Third, I am exploring more intuitive illustration of the ECG's and MCG's. In particular, I plan to explore graph and network visualization techniques developed by researchers in the Information Visualization community. Fourth, I expect to investigate automatic techniques for vector field simplification and make use MCGs to obtain a multi-scale representation of flow which can be used to guide vector field clustering, vector field compression, and automatic simplification. Fifth, more rigorous mathematical analysis and proofs need to be introduced to guarantee the minimal structure will be obtained after vector field simplification. Sixth, I plan to study the reconstruction of the original vector field from the obtained ECG/MCG graph, which can be applied to vector field compression. Finally, there is a need to extend the work to 3D volume vector fields and much larger scale (e.g. out-of-core) datasets, as well as the application to other types of data analysis including scalar fields [4] and tensor fields [104] and more advanced visualization problems beyond flow visualization [6, 85]. The research on the extension of the presented techniques to time-dependent vector field analysis [24,75] is also an important and non-trivial direction.

12.2 Vector Field Design

12.2.1 Summary

In the second part of the paper I address the problem of the design of time-varying vector field on surfaces. More specifically, I propose the use of the parameterized vector fields to approximate the solution. Two different types of vector fields are discussed for different purposes in texture synthesis and animation. Various design techniques are introduced to address the design of these two types of fields efficiently. The initial fields can be further edited to eliminate undesired effects. To my best knowledge, the presented design framework is the first in its kind for general time-varying vector field design with bifurcation control.

12.2.2 Future Directions

I do not currently provide an explicit solution to control the in-between field generation. Therefore, unexpected behaviors may arise which requires either post processing or re-configurating the initial setting and re-produce it again. This regeneration process is expensive compared to steady field design and does not guarantee that more satisfiable solutions can be found. In the future, I expect to study a more robust technique that can inform the user what could possibly be obtained given the specified constraints to address this problem. Second, I have only focused on a small set of vector field features for the creation of a parameterized vector field. There are other design primitives that may be important such

as streaklines and timelines. Third, the bifurcation control techniques proposed in this work are still limited due to the lack of the support of a systematic time-varying dynamics theory. Fourth, it will be interesting to experiment other vector field generation methods such as the one based on discrete calculus (DEC) [21].

This work opens a new range of the field design topic which can be extended to the design of *higher order fields*, such as time-varying tensor field design. In addition, it is likely that in the future more complicated vector field design problems will attract computer graphics researchers. For instance, 3D shape morphing and 3D fluid control will require the assistance of time-dependent 3D vector field design. Of course, 3D vector field design can help to verify existing or newly developed volume flow visualization techniques and serve for education purposes as well.

Bibliography

[1] Computer assisted proofs in dynamics group. http://capd.wsb-nlu.edu.pl/.

[2] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, 2003.

[3] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[4] Valerio Pascucci Attila Gyulassy, Peer-Timo Bremer and Bernd Hamann. A practical approach to morse smale complex computation: Scalability and generality. In *IEEE Trans. Vis. Comput. Graph. (IEEE Visualization 2008)*, volume 14, pages 1619–1626, 2008.

[5] Adam W. Bargteil, Funshing Sin, Jonathan E. Michaels, Tolga G. Goktekin, and James F. O'Brien. A texture synthesis method for liquid animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Sept 2006.

[6] Raghu Machiraju Bela Soni, David Thompson. Visualizing particle/flow structure interactions in the small bronchial tubes. In *IEEE Trans. Vis. Comput. Graph. (IEEE Visualization 2008)*, volume 14, pages 1412–1427, 2008.

[7] Jeff R. Cash and Alan H. Karp. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Trans. Math. Softw.*, 16(3):201–222, 1990.

[8] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, Jul./Aug. 2007.

[9] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient Morse Decompositions of Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):848–862, Jul./Aug. 2008.

[10] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Mller, and Eugene Zhang. Interactive procedural street modeling. *ACM Transactions on Graphics (Siggraph 2008)*, 27(3), 2008. Article 103: 1–10.

[11] Stephen Chenney. Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 233–242, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press, Cambridge, MA, 1990.

[13] W. de Leeuw and R. van Liere. Visualization of Global Flow Structures Using Multiple Levels of Topology. In *Data Visualization '99 (VisSym '99)*, pages 45–52. May 1999.

[14] W. de Leeuw and R. van Liere. Multi-level Topology for Flow Visualization. *Computers and Graphics*, 24(3):325–331, June 2000.

[15] Wim C. de Leeuw and Robert van Liere. Collapsing flow topology using area metrics. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 349–354, San Francisco, 1999.

[16] Thierry Delmarcelle and Lambertus Hesselink. The topology of symmetric, second-order tensor fields. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 140–147, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[17] Huong Quynh Dinh, Anthony Yezzi, and Greg Turk. Texture transfer during shape transformation. *ACM Trans. Graph.*, 24(2):289–310, 2005.

[18] Albrecht Dold. *Lectures on Algebraic Topology*. Springer-Verlag Berlin Heidelberg, New York, 1980. Proposition 5.9.

[19] H. Edelsbrunner, J. Harer, and A Zomorodian. Hierarchical morse-smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom*, 30:87–107, 2003.

[20] Michael Eidenschink. *Exploring global dynamics : a numerical algorithm based on the Conley index theory*. PhD thesis, Georgia Institute of Technology, 1996.

[21] Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. Design of tangent vector fields. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 56, New York, NY, USA, 2007. ACM.

[22] M. S. Floater. Mean value coordinates. *CAGD*, (20):19–27, 2003.

[23] Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. Sketching hairstyles. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 31–36, New York, NY, USA, 2007. ACM.

[24] Christoph Garth and Florian Gerhardt. Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1464–1471, 2007. Member-Xavier Tricoche and Member-Hagen Hans.

[25] Daniel Morse Stephen Sinder Sourabh V. Apte James A. Liburdy Guoning Chen, Zhongzang Lin and Eugene Zhang. Multiscale feature detection in unsteady separated flows. *International Journal of Numerical Analysis and Modeling*, 5, Supp:17–35, 2008.

[26] J. Hale and H. Kocak. *Dynamics and Bifurcations*. New York: Springer-Verlag, 1991.

[27] 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.

[28] J. L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11:36–46, May 1991.

[29] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, New York, NY, USA, 1998. ACM.

[30] Aaron Hertzmann and Ken Perlin. Painterly rendering for video and interaction. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12, New York, NY, USA, 2000. ACM.

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

[32] C.R. Johnson. Top Scientific Visualization Research Problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, July/August 2004.

[33] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2004.

[34] W. D. Kalies and H. Ban. A computational approach to Conley's decomposition theorem. *Journal of Computational and Nonlinear Dynamics*, 1(4), 2006.

[35] W. D. Kalies, K. Mischaikow, and R. C. A. M. VanderVorst. An algorithmic approach to chain recurrence. *Found. Comput. Math.*, 5(4):409–449, 2005.

[36] D. N. Kenwright. Automatic detection of open and closed separation and attachment lines. In *Proceedings IEEE Visualization 98*, pages 151–158, 1998.

[37] Vivek Kwatra, David Adalsteinsson, Theodore Kim, Nipun Kwatra, Mark Carlson, and Ming Lin. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):939–952, 2007.

[38] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

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

[40] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology Based Flow Visualization: The State of the Art. In *Topology-Based Methods in Visualization Workshop (TopoInVis 2005)*, pages 1–19. Springer–Verlag, 2007.

[41] R. S. Laramee, B. Jobard, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proceedings IEEE Visualization '03*, pages 131–138. IEEE Computer Society, October 2003.

[42] Robert S. Laramee, Daniel Weiskopf, Juergen Schneider, and Helwig Hauser. Investigating swirl and tumble flow with a comparison of visualization techniques. In *Proceedings IEEE Visualization 04*, pages 51–58, 2004.

[43] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 24(3):777–786, 2005.

[44] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 541–548, New York, NY, USA, 2006. ACM.

[45] Konstantin Mischaikow and Marian Mrozek. Conley index. In *Handbook of dynamical systems, Vol. 2*, pages 393–460. North-Holland, Amsterdam, 2002.

[46] M. Mrozek and P. Zgliczy Nski. Set arithmetic and the enclosing problem in dynamics. *Annales Pol. Math.*, 74:237–259, 2000.

[47] Marian Mrozek. Leray functor and cohomological conley index for discrete dynamical systems. *Trans. Amer. Math. Soc.*, 318:149–178, 1990.

[48] Fabrice Neyret. Advected textures. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, july 2003.

[49] Emil P., Adam F., and Hugues H. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, pages 465–470, July 2000.

[50] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55:1–55:10, 2007.

[51] Jacob Palis, Jr. and Welington de Melo. *Geometric theory of dynamical systems*. Springer-Verlag, New York, 1982. An introduction, Translated from the Portuguese by A. K. Manning.

[52] R Peikert and M. Roth. The parallel vectors operator a vector field visualization primitive. In *Proceedings IEEE Visualization 99*, pages 263–270, 1999.

[53] K. Polthier and E. Preuss. *Identifying vector fields singularities using a discrete hodge decomposition*. Ed: H.C. Hege, K. Polthier, 2003.

[54] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4):775–792, Dec. 2003.

[55] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[56] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–202, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[57] Nicolas Ray, Wan Chiu Li, Bruno Lvy, and Alla Sheffer an d Pierre Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 2006.

[58] J.W. Robbin and D. Salamon. Dynamical systems, shape theory and the conley index. *Ergodic Theory Dynamical Systems*, 8:375–393, 1988.

[59] Monika Jankun-Kelly Eugene Zhang Robert S. Laramee, Guoning Chen and David S. Thompson. Bringing topology-based flow visualization to the application domain. In *Topology-Based Methods in Visualization II (Proceedings of Topo-In-Vis 2007), Mathematics and Visualization*, pages 161–176. Springer-Verlag, 2009.

[60] A. Rockwood and S. Bunderwala. A toy vector field based on geometric algebra. In *Proceeding Application of Geometric Algebra in Computer Science and Engineering, (AGACSE2001)*, pages 179–185, 2001.

[61] G. Scheuermann, H. Hagen, H. Krüger, M. Menzel, and A. Rockwood. Visualization of Higher Order Singularities in Vector Fields. In *Proceedings of IEEE Visualization '97*, pages 67–74, October 1997.

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

[63] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci.*, volume 93, pages 1591–1595, 1996.

[64] A. Sharf, D. Alcantara, T. Lewiner, C. Greif, A. Sheffer, N. Amenta, and D. Cohen-Or. Space-time surface reconstruction using incompressible flow. In *ACM Transactions on Graphics (SIGGRAPH ASIA Conference Proceedings)*, 2008.

[65] J. Stam. Flows on surfaces of arbitrary topology. In *ACM Transactions on Graphics (SIGGRAPH 2003)*, volume 22(3), pages 724–731, July 2003.

[66] Jos Stam. Stable fluids. In Alyn Rockwood, editor, *Proceedings of ACM SIGGRAPH 1999*, pages 121–128, Los Angeles, 1999. Addison Wesley Longman.

[67] Guoning Chen Sourabh V. Apte James A. Liburdy Stephen Snider, Daniel Morse and Eugene Zhang. Detection and analysis of separated flow induced vortical structures, January 2008.

[68] Andrzej Szymczak. The conley index for discrete semidynamical systems. *Topology and its Applications*, 66(3):215–240, 1995.

[69] Andrzej Szymczak. *Index Pairs: from Dynamics to Combinatorics and Back*. PhD thesis, Georgia Institute of Technology, 1999.

[70] K. Mischaikow T. Kaczynski and M. Mrozek. *Computational Homology*. Springer, 2004.

[71] H. Theisel. Designing 2d vector fields of arbitrary topology. In *Computer Graphics Forum (Proceedings Eurographics 2002)*, volume 21, pages 595–604, July 2002.

[72] H. Theisel, Ch. Rössl, and H.-P. Seidel. Combining Topological Simplification and Topology Preserving Compression for 2D Vector Fields. In *Pacific Graphics*, pages 419–423, 2003.

[73] H. Theisel, Ch. Rössl, and H.-P. Seidel. Compression of 2D Vector Fields Under Guaranteed Topology Preservation. In *Eurographics (EG 03)*, volume 22(3) of *Computer Graphics forum*, pages 333–342, September 1–6 2003.

[74] H. Theisel, T. Weinkauf, H.-P. Seidel, and H. Seidel. Grid-Independent Detection of Closed Stream Lines in 2D Vector Fields. In *Proceedings of the Conference on Vision, Modeling and Visualization 2004 (VMV 04)*, pages 421–428, November 2004.

[75] Holger Theisel, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel. Topological methods for 2d time-dependent vector fields based on stream lines and path lines. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):383–394, 2005.

[76] Yiying Tong, Santiago Lombeyda, Anil N. Hirani, and Mathieu Desbrun. Discrete multiscale vector field decomposition. *ACM Trans. Graph.*, 22(3):445–452, 2003.

[77] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Trans. Graph.*, 25(3):1160–1168, 2006.

[78] X. Tricoche. *Vector and Tensor Field Topology Simplification, Tracking, and Visualization*. Schriftenreihe Fachbereich Informatik (3), University of Kaiserslautern, 2002.

[79] X. Tricoche, C. Garth, G. L. Kindlmann, E. Deines, G. Scheuermann, M. Ruetten, and C. D. Hansen. Visualization of intricate flow structures for vortex breakdown analysis. In *Proceedings IEEE Visualization 04*, pages 187–194, 2004.

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

[81] X. Tricoche, G. Scheuermann, and H. Hagen. Topology-Based Visualization of Time-Dependent 2D Vector Fields. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym '01)*, pages 117–126, May 28–30 2001.

[82] X. Tricoche, G. Scheuermann, and H. Hagen. Topology-based visualization of time-dependent 2d vector fields. In *Data Visualization 2001 (Joint Eurographics-IEEE TCVG Symposium on Visualization Proceedings)*, pages 117–126, 2001.

[83] Xavier Tricoche, Gerik Scheuermann, and Hans Hagen. A topology simplification method for 2d vector fields. In *Proceedings of IEEE Visualization 2000*, pages 359–366, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[84] G. Turk. Texture synthesis on surfaces. In *ACM Transactions on Graphics (SIGGRAPH 2001)*, pages 347–354, August 2001.

[85] Robin Vallacher and Andrzej Nowak. *Dynamical Systems in Social Psychology*. Academic Press, 1994. ISBN: 0127099905.

[86] J.J. van Wijk. Image based flow visualization. In *ACM Transactions on Graphics (SIGGRAPH 2002)*, volume 21(2), pages 745–754, Jul 2002.

[87] J.J. van Wijk. Image based flow visualization for curved surfaces. In *Proceedings IEEE Visualization '03*, pages 123–130. IEEE Computer Society, 2003.

[88] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proceedings IEEE Visualization 00*, pages 163–170, 2000.

[89] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *ACM Trans. Graph.*, 25(3):1118–1125, 2006.

[90] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Explicit control of vector field based shape deformations. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 291–300, Washington, DC, USA, 2007. IEEE Computer Society.

[91] L.-Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of ACM SIGGRAPH 2001*, pages 355–360, July 2001.

[92] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3d vector fields. In *Computer Graphics Forum (Eurographics 2004)*, number 3, pages 469–478, October 2004.

[93] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. Extracting higher order critical points and topological simplification of 3d vector fields. In *Proceedings IEEE Visualization '05*. IEEE Computer Society, October 2005.

[94] Daniel Weiskopf and Gordon Erlebacher. *Overview of flow visualization.* Elsevier, Amsterdam, 2005.

[95] J. Wejchert and D. Haumann. Animation aerodynamics. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 91)*, pages 19–22, 1991.

[96] Mark Wiebe and Ben Houston. The tar monster: creating a character with fluid simulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 64, New York, NY, USA, 2004. ACM.

[97] T. Wischgoll and G. Scheuermann. Detection and visualization of planar closed streamline. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):165–172, 2001.

[98] T. Wischgoll, G. Scheuermann, and H. Hagen. Tracking Closed Streamlines in Time Dependent Planar Flows. In *Proceedings of the Vision Modeling and Visualization Conference 2001 (VMV 01)*, pages 447–454, November 2001.

[99] L. Xu, J.N. Chen, and J.Y. Jia. A segmentation based variational model for accurate optical flow estimation. pages I: 671–684, 2008.

[100] Yan Qian Ye, Sui Lin Cai, Lan Sun Chen, Ke Cheng Huang, Ding Jun Luo, Zhi En Ma, Er Nian Wang, Ming Shu Wang, and Xin An Yang. *Theory of limit cycles*, volume 66 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI, second edition, 1986. Translated from the Chinese by C. Y. Lo.

[101] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Transactions on Graphics*, 25(4):1294–1326, 2006.

[102] Eugene Zhang. *Surface Topological Analysis for Image Synthesis.* Georgia Institute of Technology, 2004.

[103] Eugene Zhang, James Hays, and Greg Turk. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):94–107, 2007.

[104] Eugene Zhang, Harry Yeh, Zhongzang Lin, and R. S. Laramee. Asymmetric tensor analysis for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):106–122, 2009.