Feature-Based Uncertainty Visualization

Keqin Wu

Feature-based uncertainty visualization

By

Keqin Wu

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2012

FEATURE-BASED UNCERTAINTY VISUALIZATION

By

Keqin Wu

Approved:

Robert J. Moorhead Professor of Electrical and Computer Engineering (Major Professor)	Song Zhang Assistant Professor of Computer Science and Engineering (Director of Dissertation)
J. Edward Swan II Professor of Computer Science and Engineering (Committee Member)	Philip Amburn Research Associate Professor of Electrical and Computer Engineering (Committee Member)
James E. Fowler Professor of Electrical and Computer Engineering (Graduate Program Director)	Sarah A. Rajala Professor and Dean of the Bagley College of Engineering

Name: Keqin Wu

Date of Degree: August 11, 2012

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: Robert Moorhead

Director of Dissertation: Song Zhang

Title of Study: Feature-based uncertainty visualization

Pages in Study: 105

Candidate for Degree of Doctor of Philosophy

While uncertainty in scientific data attracts an increasing research interest in the visualization community, two critical issues remain insufficiently studied: (1) visualizing the impact of the uncertainty of a data set on its features and (2) interactively exploring 3D or large 2D data sets with uncertainties. In this study, a suite of feature-based techniques is developed to address these issues.

First, a framework of feature-level uncertainty visualization is presented to study the uncertainty of the features in scalar and vector data. The uncertainty in the number and locations of features such as sinks or sources of vector fields are referred to as feature-level uncertainty while the uncertainty in the numerical values of the data is referred to as data-level uncertainty. The features of different ensemble members are indentified and correlated. The feature-level uncertainties are expressed as the transitions between corresponding features through new elliptical glyphs. Second, an interactive visualization tool for exploring scalar data with data-level and two types of feature-level uncertainties — contour-level and topology-level uncertainties — is developed. To avoid visual cluttering and occlusion, the uncertainty information is attached to a contour tree instead of being integrated with the visualization of the data. An efficient contour tree-based interface is designed to reduce users' workload in viewing and analyzing complicated data with uncertainties and to facilitate a quick and accurate selection of prominent contours. This thesis advances the current uncertainty studies with an in-depth investigation of the feature-level uncertainties and an exploration of topology tools for effective and interactive uncertainty visualizations. With quantified representation and interactive capability, feature-based visualization helps people gain new insights into the uncertainties of their data, especially the uncertainties of extracted features which otherwise would remain unknown with the visualization of only data-level uncertainties.

ACKNOWLEDGEMENTS

I would like to start by thanking my advisors, Dr. Song Zhang and Dr. Robert Moorhead. Both went far beyond their duty as advisors and have inspired much more than just my research.

I would like to thank Dr. Edward Swan II and Dr. Philip Amburn, my committee members, and Derek Irby, my fellow researcher, who in one way or another helped me to complete this thesis.

I would like to acknowledge Dr. Jibonananda Sanyal, my former fellow graduate student, for providing the simulated weather data and constructive suggestions.

This work has been supported in part by the Northern Gulf Institute, a NOAA Cooperative Institute and NSF, Award No. NSF111787.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Uncertainty is a common and crucial issue in scientific data. The goal of uncertainty visualization is to provide users with visualizations that incorporate uncertainty information to aid data analysis and decision making; however, it is challenging to quantify uncertainties appropriately and to visualize uncertainties effectively without affecting the visualization effect of the underlying data information.

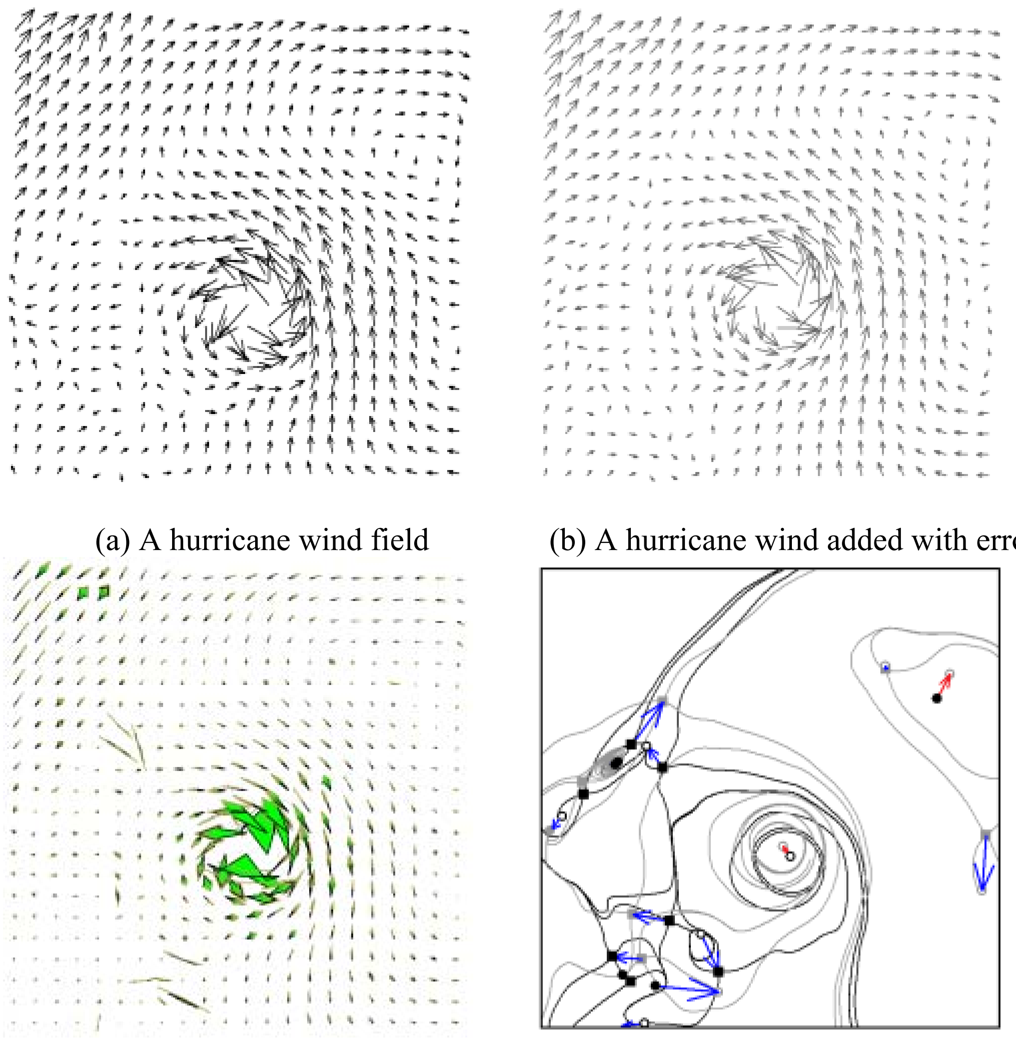
Uncertainty in scientific data can be broadly defined as statistical variations, spread, errors, differences, and minimum maximum range values, etc. [1] This broad definition covers most, if not all, of the possible types and sources of uncertainty related to numerical values of the data. We are particularly interested in the uncertain positional deviations of the features (e.g. extrema, sinks, sources, contours, and contour trees) in the data. In this thesis, these feature-related uncertainties are referred to as feature-level uncertainty while the uncertainties related to the uncertain numerical values of the data such as statistical variation, spread, errors, differences, and range, are referred to as data-level uncertainty. Most current uncertainty visualizations focus on encoding data-level uncertainty information into different graphics primitives such as color, glyph, and texture, which are attached to surfaces or embedded in volumes [1]. Those methods, in essence, give global insight into the data by differentiating the area of high uncertainty from that of low uncertainty; however, the impact of the uncertainty on the important features of the data is hard to assess in such visualizations.

1.1 The gap between data-level uncertainty and feature-level uncertainty

Knowing the uncertainty concerning features is important for decision making. Many uncertainty visualizations based on statistical metrics merely measure uncertainty on the data-level — the uncertainty concerning the numerical values of the data and introduced in data acquisition and processing. While these techniques achieved decent visualization results, they do not provide users an insight into how much uncertainty exists for the features in the data.

In scientific data, the difference between a known correct datum and an estimate is among the uncertainties most frequently investigated. To compare data-level uncertainty and feature-level uncertainty, we investigate two data sets. The first data set is a slice of a simulated hurricane Lili wind field (Figure 1.1a). The second data set is created by adding random noise to the first dataset (Figure 1.1 b). For a wind vector, its data-level uncertainty is represented as both angular difference and magnitude difference between vectors of the two fields. Figure 1.1c shows the arrow glyphs [2] for visualizing the uncertainty of vector fields with the angular uncertainty presented as the span of each arrow glyph and magnitude uncertainty as the two winglets around an arrow head. For details about designing arrow glyphs for vector field uncertainty, please refer to Wittenbrink et al.'s paper [2].

As shown in Figure 1.1c, with the uncertainty glyphs, users may notice that the area around the hurricane eye (the major vertex in the middle of Figure 1.1c) exhibits high data-level uncertainty which raises a question: does it affect the location of hurricane eye? Only an explicit comparison between the hurricane eyes in the two fields will answer this question. We therefore extract topologies of the two fields as shown in Figure 1.1d. The sink point (in black) inside the hurricane eye noticeably shifts northwest (indicated by a red arrow) from the original vector field to the new one (in gray). Another question is, is there anything hidden in the relatively low uncertainty area indicated by the small arrow glyphs? A quick look at the Figure 1.1d reveals that the upper corner vortex significantly shifts its position (indicated by a red arrow) though it is located at the region with relatively low data-level uncertainty.



(a) A hurricane wind field

(b) A hurricane wind added with error

(c) Uncertainty glyphs [2]

(d) Feature deviations

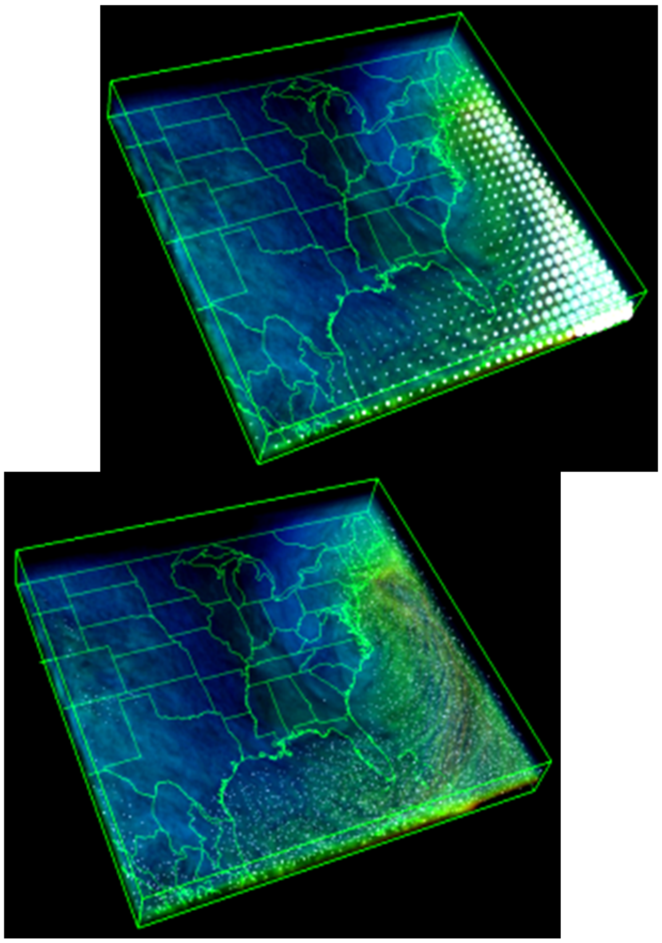
Figure 1.1 An uncertain vector field and its uncertain feature locations.

This example illustrates that merely investigating the data-level uncertainty may not tell the whole story about the data and that the feature-level uncertainty is an indispensable part of uncertainty that cannot be neglected. Besides, visualizing uncertainty of features, instead of that of the data, may provide a succinct and meaningful representation of the uncertainty and thus give a better interpretation of the data.

1.2 Challenges of visualizing uncertainty

Representation of uncertainty in 3D or large 2D data sets could encounter severe issues such as cluttered displays, information overload, and occlusions. Some uncertainty visualizations place glyphs that encode uncertainty within the visualization of the data. For instance, Sanyal et al. [4] visualized data-level uncertainties via circular or ribbon-like glyphs over a color-mapped image of the data. Due to the overlaps between the data and uncertainty glyphs, the number of the glyphs has to be limited and information loss for both the data and uncertainty is unavoidable. Other techniques which overlay or embed uncertainty representation in the data visualization face similar issues. For example, Figure 1.2 shows volume rendering data with uncertainty represented by circular glyphs and point clouds. In Figure 1.2a, the sizes of the glyphs vary according to the magnitude of the uncertainty. In Figure 1.2b, the more crowded the point cloud is, the higher the uncertainty is. As shown in Figure 1.2, with uncertainty representation embedded within the volumetric data, the details of the data are noticeably blocked by the crowed glyphs or points while the glyphs or points are overlapped with each other and appear to be blurred or buried in a 3D scene.

Meanwhile, interaction with the visualization of 3D or large 2D data sets could encounter issues such as geometry bandwidth bottleneck, depth perception, occlusion, and inefficiency in 3D object selection. These issues are inherent in interactive 3D graphics applications, and may be intensified in the integrated visualization of a 3D dataset and its uncertainty because there is more information to show in such visualizations.



(a) Circular glyphs (b) Point cloud

Figure 1.2 Volume rendering water-vapor mixing ratio data with uncertainty information represented by circular glyphs and point cloud.

1.3 Feature-based visualization

The challenge of understanding large and intricate data has made feature-based visualization attractive [3]. The feature-based methods both emphasize features and avoid visual clutter in the resulting visualizations. The abstract representation of features may provide a key to alleviate the perception and interaction issues when exploring 3D and large 2D data sets with uncertainties.

Several features such as maxima, minima, sinks, sources, contours, and contour hierarchies which are represented through contour trees, interest us the most. Contours, including iso-lines in 2D scalar data or iso-surfaces in 3D scalar data, are among features mostly used for exploring a scalar field and its uncertainty. Critical points such as sinks, sources, maxima, and minima are representative of topological features that carry significant physical meaning of the data. A contour tree stores the nesting relationships of the contours of a scalar field. It is a popular visualization tool for revealing the topology of contours [4], generating seed set for accelerated contour extraction [5], and providing users an interface to select individual contours [6].

1.4 Objective

The objectives of this thesis are (1) to bring awareness to the existence of feature-level uncertainties and (2) to design an interactive tool for exploring 3D and large 2D data sets with uncertainty. Particularly, this thesis presents (1) a comprehensive framework for feature-level uncertainty visualization, (2) quantified representations of feature-level uncertainties, and (3) an interactive contour tree-based interface for exploratory visualization of 2D and 3D data with uncertainty information.

In this thesis, we propose to investigate the uncertainty information on the data-level and the feature-level to provide users a more comprehensive view of the uncertainties in their data. We do not assume a specific distribution in our data. While our uncertainty representation can be adapted to different uncertainty models, in this thesis, we measure the uncertainty according to the differences between the data values, critical points, contours, or contour trees of different ensemble members.

CHAPTER II

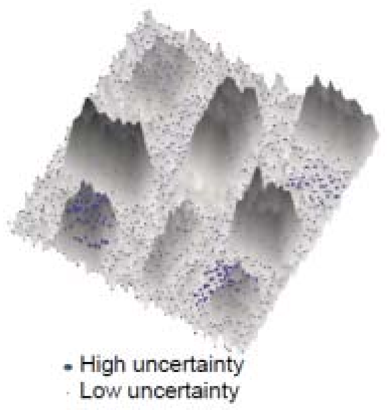
LITERATURE REVIEW

This literature review covers uncertainty visualization, feature-based visualization, and glyph-based techniques.

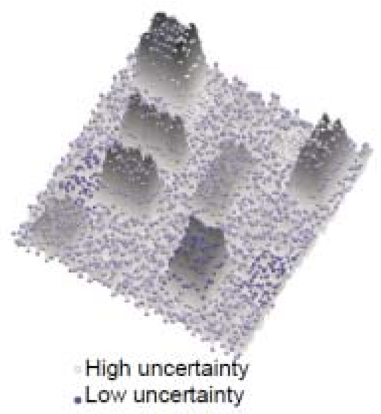
2.1 Uncertainty Visualization

A number of representation methods have been proposed for visualizing uncertainties. Pang et al. [1] roughly classified the techniques into adding glyphs, adding geometry, modifying geometry, modifying attributes, animation, sonification, and psycho-visual approaches.

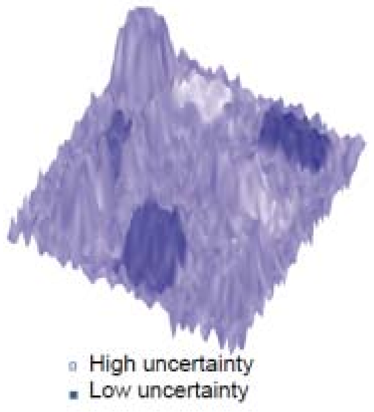
Several efforts have been made to identify potential visual attributes for uncertainty visualization. MacEachren [7] suggested the use of hue, saturation, and intensity for representing uncertainty on maps. Ehlschlaeger et al. [8] showed how animation could be used to depict uncertainty of elevation data. Hengl and Toomanian [9] showed how color mixing and pixel mixing can be used to visualize uncertainty in soil science applications. Davis and Keller [10] suggested value, color, and texture for representing uncertainty on static maps. Djurcilov et al. [11] used opacity deviations and noise effects to provide qualitative measures for the uncertainty in volume rendering. Sanyal et al. [12] conducted a user study to compare the effectiveness of four uncertainty representations: traditional error bars, scaled size of glyphs, color-mapping on glyphs, and color-mapping of uncertainty on the data surface. In their experiments, scaled sphere and color mapped sphere perform better than traditional error bars and color-mapped surfaces. Later, they proposed graduated glyphs and ribbons to encode uncertainty information of weather simulations [13]. Figure 2.1 is borrowed from Sanyal et al. [12] and shows the four uncertainty visualization techniques in their user study.



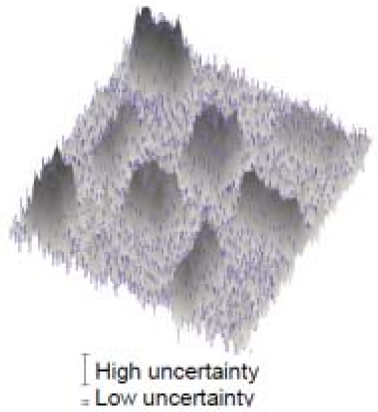
(a) Scaling the size of glyphs



(b) Altering the color attribute of glyphs



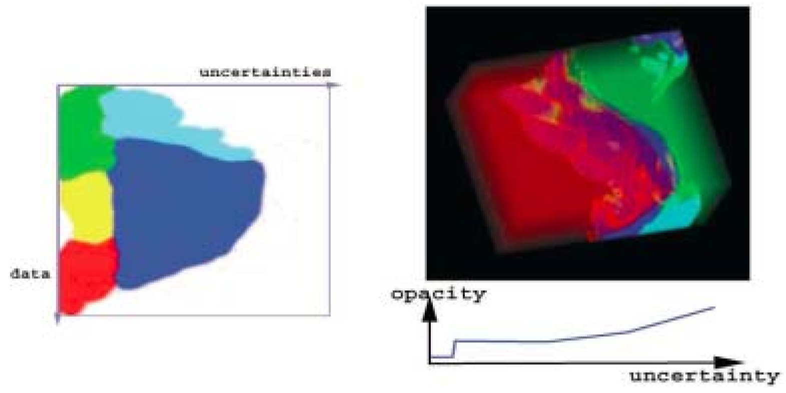
(c) Color-mapping on surface



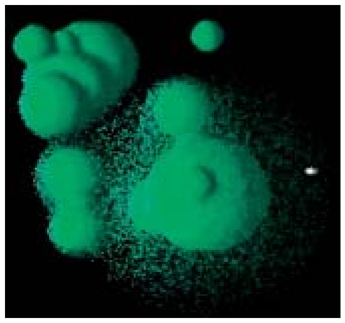
(d) Using error bars

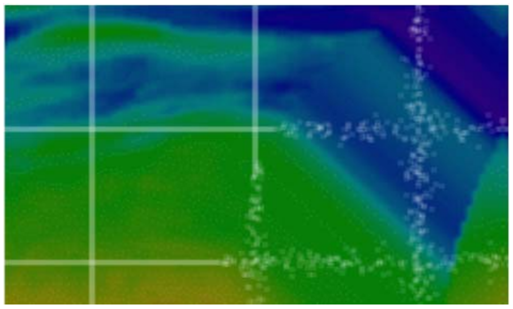
Figure 2.1 Samples of uncertainty visualization techniques for 2D datasets [12].

While the uncertainty visualization is application-dependent in many cases, two visualization schemes are widely used: using intuitive metaphors, such as blurry and fuzzy effects [11], [14], [15], which naturally implies the existence of uncertainty; and using quantitative glyphs [13], [16], which shows quantified uncertainty information explicitly. Both schemes have their own tradeoffs. In quantitative glyphs the uncertainty information has to be shown in a discrete way. By using uncertainty metaphors, people get less quantified information about uncertainty since they cannot tell levels of blur or fuzziness apart accurately [17]. To reveal uncertainty accurately, uncertainty glyph is preferred to uncertainty metaphors. Figure 2.2 shows examples of uncertainty visualizations using blurry or fuzzy effects. Figure 2.3 shows examples of uncertainty visualizations using quantitative glyphs.



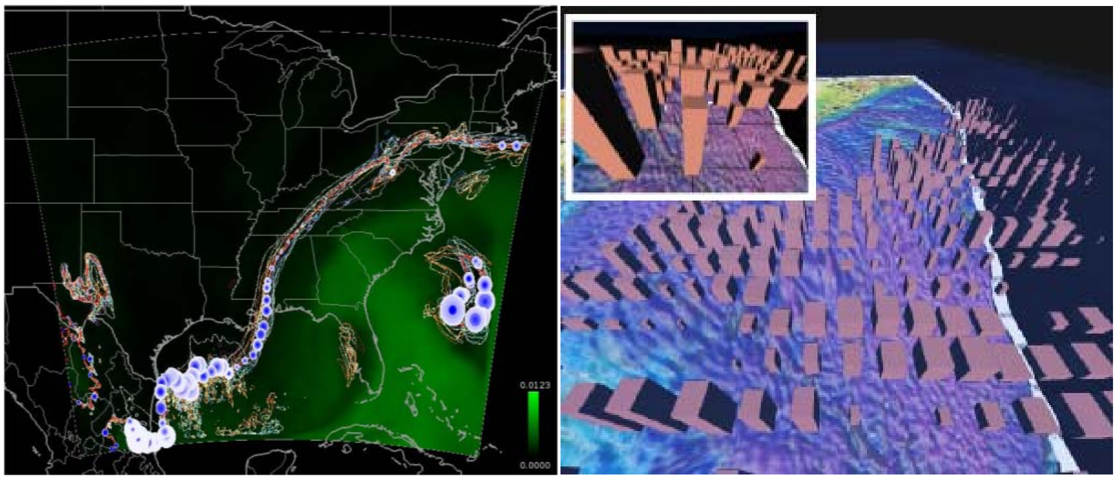
(a)





(b) (c)

Figure 2.2 Samples of uncertainty visualizations using blurry or fuzzy effects. (a) Visualizing scalar volumetric data with uncertainty [14]: low uncertainty data have low opacity while high uncertainty data have high opacity; low uncertainty data with low values are mapped to green while low uncertainty data with high values are mapped to red; the rest of the values are mapped to yellow, blue, and light blue. (b) Point-based probabilistic surfaces [15]: underlying polygonal model is shown with displaced points at each location on the surface according to the uncertainty value at the location. (c) Procedural annotation of uncertain information: uncertainty information is encoded in the lines of the grid which overlays the data [19].



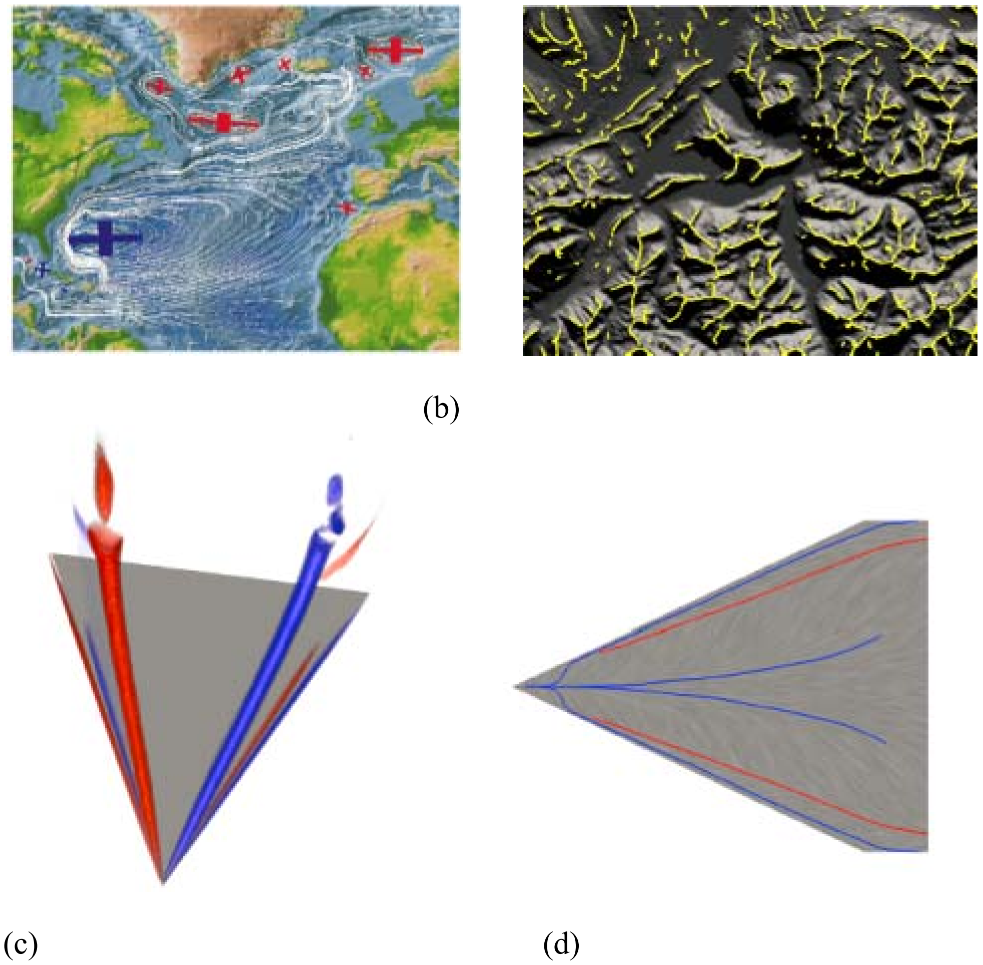
(a) (b)

Figure 2.3 Samples of uncertainty visualizations using quantitative glyphs. (a) Graduated glyphs [13] showing uncertainty along the perturbation pressure contour of the ensemble mean along with a spaghetti plot . (b) Box glyphs [16] scaled by a factor proportional to the uncertainty in each corresponding dimension and showing positional uncertainty on an underwater surface [19].

Several methods have been developed to address the uncertainty of the size, position, and shape of contours [15], [18], [19]. Pang et al. [1] presented fat surfaces that use two surfaces to enclose the volume in which the true but unknown surface lies. Pauly et al. [20] quantified and visualized the uncertainty introduced in the reconstructions of surfaces from point cloud data. Pfaffelmoser et al. [19] presented a method for visualizing the positional variability around a mean iso-surface using direct volume rendering. A method to compute and visualize the positional uncertainty of contours in uncertain input data has been suggested by Pöthkow and Hege [21]. Assuming certain probability density functions, they modeled a discretely sampled uncertain scalar field by a discrete random field. Pöthkow et al. [22] extended their model to correlated random fields.

Among uncertainty methods, only a few are proposed to address the topological features. Otto et al. [23] studied the uncertain topological segmentation of a vector field by introducing the probability density that a particle starting from a given location will converge to a considered source or sink. The uncertainty related to the topology structure of a scalar field, is barely studied.

2.2 Feature-Based Visualization



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

Figure 2.4 Samples of feature-based visualization methods. (a) Flow in the Atlantic Ocean with streamlines and ellipses indicating vortices [24]. (b) Height ridges [25] indicated by yellow lines. (c) Volume rendering of the vortices above a delta wing [26]. (d) Separation and attachment lines of the shear flow on the wing surface [26].

Feature-based visualization methods generate images that depict features of particular interest in complex data with a limited set of points, lines, and volumes for the considered application [26]. Features are defined in various ways. There are application-dependent features, such as height ridges, vortices, separation and attachment lines, and more general features, such as contours and topological features. This work focuses on the latter in seeking a general solution for feature-based visualization. The concise representation generated by topological methods has become a popular tool for visualization. Figure 2.4 shows examples of feature-based visualizations.

Flow topology characterizes a flow in that the relatively uniform flow behavior in each topological region can be deduced from its boundary [27], [28]. After taking the gradient derivative of a scalar field, one may visualize the scalar field via the topology of the resulting vector field [29]. In addition, topology simplification [28], [30] provides an even more compact way to depict data.

As for scalar fields, two structures for storing topological information are Contour Tree (CT) and Morse-Smale (MS) complex. A contour tree [4] is a loop free case of a Reeb graph [31] which describes the hierarchical relationship of contours. A MS-complex decomposes a scalar field into quadrangular cells with uniform gradient flow behavior [32], [33]. In this thesis, we used a contour tree rather than a MS complex because people are more interested in contours than the partition of gradient flow. In latter sections, we will specifically discuss the uncertainty of contours and contour trees.

The contour tree is a powerful visualization tool for abstract data representations [34], contour extractions [35], [36], transfer function design [37], etc. A number of algorithms have been used to compute contour trees [36], [38], [4]. Several contour tree based topology simplifications were proposed to either remove noise or extract important contours hierarchically. Usually, a simplification method removes paired critical points with increasing importance which is measured by persistence or other geometric measures [39], [40]. We adopt the usual bottom-up contour tree simplification by removing pairs with ascending persistence. The computation of the geometric properties, such as length, surface area, or volume of individual contours is discussed in [6], [41], [42]. The overlapped volume or area between contours is often used to compare or map contours of different scalar fields [41], [42]. Utilizing a contour tree as an interface to explore data was first suggested by Bajaj et al. [6]. They introduced the "contour spectrum" to facilitate iso-value selection based on iso-surface properties such as iso-line length and iso-surface area. Carr et al. [40], [5] proposed "path seeds" to facilitate a selection of distinct contours and "flexible iso-surface" with different levels of simplification to highlight the fundamental structure of data. These methods demonstrated a powerful paradigm of using a simplified contour tree for contour selection. To present a contour tree graph that is large in size and cluttered due to self-intersections, Pascucci et al. [39] proposed a 3D orrery-like layout based on a novel branch decomposition scheme. However, although there is no intersection between branches in a 3D sense, there is still notable overlapping among branches in a side view of their contour tree layout. In addition, a planar display allows for easier selection of an object and therefore is more desirable for an interface design. Heine et al. [43] compared several planar contour tree layouts and identified a orthogonal layout as one of the most effective layouts in terms of representing branch hierarchy, minimizing self-intersections, and associating ancillary information such as geometric properties of contours.

Feature tracking is a way to investigate the feature evolutions of time-varying data. Some feature-tracking methods correlate features by measuring their positional distance or overlapped area while others utilize an intermediate data field derived from two neighboring time slices and trace paths along critical points within it [28], [44], [45]. Theisel's Feature Flow Field (FFF) method [44] is especially interesting to us because it provides a generic approach to feature tracking. Independent of underlying grids, the FFF method captures the temporal evolution of a feature using stream object integration in a derived Feature Flow Field. The concept of FFF has been successfully applied to tracking critical points [46], extracting Galilean invariant vortex core lines [47], topology simplification [48], and topology comparison [49].

2.3 Glyph-Based Techniques

Glyphs, also referred to as icons, encode meaningful information into illustrative symbols. Because glyphs are generally not placed in dense packings, the free space between them allows the visualization of additional information.

Many glyphs are employed in data and uncertainty visualization. Glyphs used for uncertainty representation are error bars, spheres, graduated glyphs, etc. One important consequence of this process is a vast data reduction. The original data field is replaced by a usually small number of attribute sets, visualized by simple geometric objects.

The effectiveness of a glyph design is usually validated by user studies. For instance, Sanyal et al. [12] conducted a user study to compare effectiveness of uncertainty representations, and conclude that compact glyphs, scaled spheres, and color mapped spheres perform better than traditional error bars and color mapped surfaces. Figure 2.1 show the uncertainty representations in their user study. Meanwhile, perceptual and cognitive theories [50], [51], [52] provide valuable guidelines for glyph design.

CHAPTER III

TOPOLOGY BACKGROUND

This chapter discusses the necessary mathematical concepts and definitions in topology, which will be used throughout this thesis. Section 3.1 mainly deals with scalar field topology, Section 3.2 with vector field topology.

3.1 Scalar field topology

The Contour Tree (CT) and the Morse–Smale (MS) complex are two important data structures for storing topological information of a scalar field. Both types of structures have been extensively used for hierarchical representation and feature extraction of scalar fields [30], [53].

3.1.1 Morse function and critical points

Consider a smooth function $f : M \rightarrow R$ defined on a manifold M.

Definition (Morse Function) f is a *Morse function* if all critical points are non-degenerate and have distinct function values.

One usually deals with piecewise linear functions given at the vertices of a triangulation or tetrahedralization. Within each simplex of M, the function f is the linear interpolation of its values at the vertices. As stated in [4], defining f as a linear interpolation over a simplicial mesh with unique data values at vertices ensures that f is a Morse function. (Note that one may need to perturb the actual data to guarantee uniqueness.)

Definition (Scalar Field Critical Point) A point $v \in M$ is a *critical point* when its gradient $\nabla f(v) = 0$.

Qualitative information about the behavior of the gradient field near a critical point is obtained by analysis of the Hessian matrix $H(f)$ of f.

2D Hessian matrix:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{3.1}$$

3D Hessian matrix:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} \tag{3.2}$$

The eigenvalues and eigenvectors of the above matrix determine the behavior of the gradient field; hence the scalar field near the critical point is much the same as for the behavior of a general vector field.

Definition (Degenerate) A critical point v of a function $f : M \rightarrow R$ is called *non-degenerate* if the Hessian matrix of f at v is non singular $\det H(f) \neq 0$; it is called *degenerate* otherwise.

A critical point is characterized by its index that is equal to the number of negative eigen-values of the Hessian of f. In a 2D case, minima, saddle, and maxima have indices 0, 1, and 2, respectively. The minima, 1-saddles, 2-saddles, and maxima in a

3D case have indices equal to 0, 1, 2, and 3, respectively. For a piecewise linear function, criticality of a vertex is classified via its link graph [54].

Given a triangulation K of 2-manifold M, the star of an interior vertex v consists of all simplices (vertices, edges, and triangles) that contain v and the link of v consists of all faces of simplices in the star that are disjointed from v. The link of every vertex is a circle (Figure 3.1). The lower link contains all simplices in the link whose endpoints are lower than v, while the upper link contains that with endpoints higher than v.

The lower link is used to classify a vertex as regular or critical. As illustrated in Figure 3.1, the lower link of v consists of $k+1 \geq 1$ connected pieces, each being an arc or a single vertex. In $k=-1$ case, v is an extremum. The lower link of maximum is full while that of a minimum is empty. In all other cases, the vertex v is regular if $k=0$, and a *k*-fold saddle if $k \geq 1$. As illustrated in Figure 3.1, for $k \geq 2$, a *k*-fold saddle can be split into k simple or 1-fold saddles.

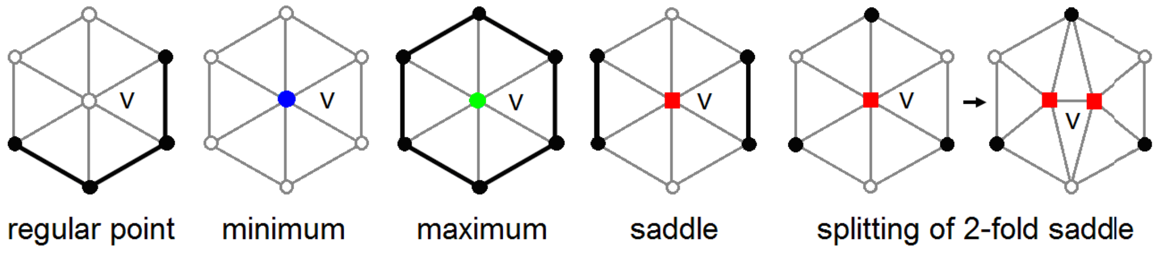


Figure 3.1 Classification of vertices based on relative height of vertices in its link. The lower link is marked black.

3.1.2 Morse-Smale complex

Definition (Stable/Unstable Manifold) Given a Morse function $f: M \to R$, the *stable manifold* $S(x)$ of a critical point x is defined as: $S(x)=\{x\} \cup \{y \in M \mid y \in im\gamma,$

$dest(\gamma) = x\}$. Symmetrically, the *unstable manifold* $U(x)$ of x is defined as:

$U(x) = \{x\} \cup \{y \in M \mid y \in im\gamma, org(\gamma) = x\}$

Definition (Morse-Smale Complex) The descending manifold *D(v)* of a critical point *v* is the union of *v* and all integral lines ending at *v* in the gradient field of f. Symmetrically, the ascending manifold *A(v)* of *v* is the union of *v* and all integral lines that start at v. If no integral line both starts and ends at a saddle, one can superimpose the descending and ascending manifolds of all critical points to obtain the MS complex of f (see Figure 3.2 for an example). Within a MS complex cell, all integral lines start at the same minimum and end at the same maximum and f is monotonic [55].

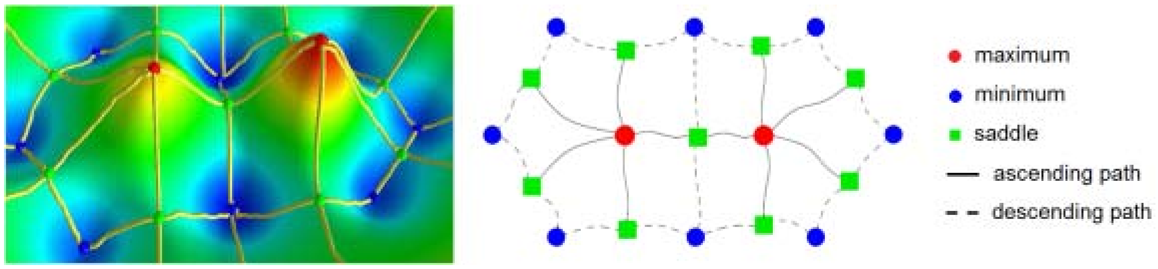


Figure 3.2 A function (left) and corresponding MS complex (right) [56].

3.1.3 Contour tree

Consider a smooth function $f : M \rightarrow R$ defined on a manifold M.

Definition (Level Set) For a given value *h*, the *level set* of *f* at *h* is the subset $L(h) = \{x \in M | f(x) = h\}$1.

Definition (Contour) Each connected component of the level set *L*(*h*) is a *contour*.

As *h* increases in the level set of *L(h)*, contours appear at local minima, join or split at saddles, and disappear at local maxima of *f*.

Definition (Contour Tree) The *contour tree* for a Morse function is a graph that tracks the evolution of contours. Each leaf node corresponds with the creation or deletion of a component at a local extremum; each interior node represents the joining and/or splitting of components at a saddle; each arc represents a component in the level sets for all values between the values of the arc's ends.

As stated by Takahashi et al [57], for a contour tree with virtual minimum, if critical points are non-degenerate and k-fold saddles are unfolded into simple ones, a contour tree has the following properties:

(1) A maximum has only one arc whose opposite endpoint is a saddle or minimum lower than it.

(2) A minimum has only one arc whose opposite endpoint is a saddle or maximum higher than it.

(3) A saddle has three incident arcs, one is lower than it and the other two are higher than it, or one is higher and the other two are lower than it.

There is a one-to-one mapping from a point in the tree to a contour of the scalar field, that is, each contour can be represented uniquely by a single point in the contour tree. A contour tree example is shown in Figure 3.3.

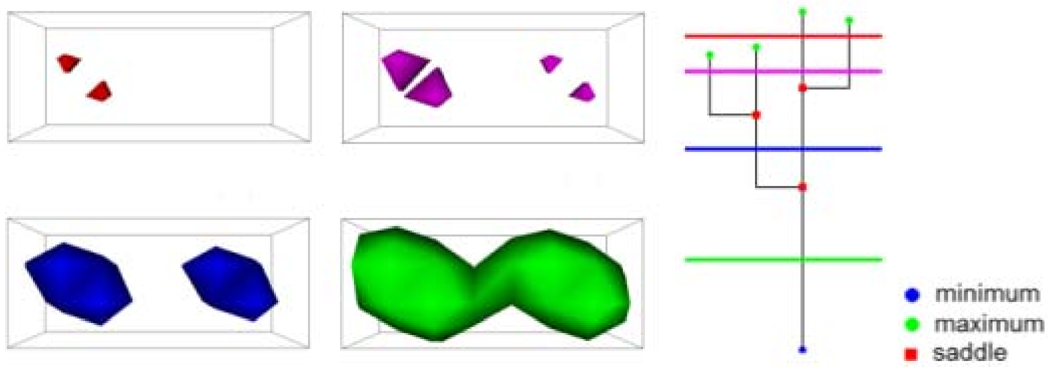


Figure 3.3 The contour tree and contours of a 3D scalar field. Each horizontal line cuts exactly an edge of the tree for every contour at the corresponding iso-value. Color is used to indicate the correspondence between a line and its corresponding contours.

Definition (Fully Augmented Contour Tree) A *fully augmented contour tree* is the contour tree augmented by all vertices in the mesh.

In this thesis, the term contour tree refers to un-augmented contour tree whose nodes are merely critical points.

3.2 Vector field topology

Definition (Vector Field Critical Point) A *critical point* in a vector field is a singularity in the field such that $v(x) = 0$.

Critical points are classified by eigenvalues of the Jacobian matrix, J, of the vector function at their positions.

2D Jacobian matrix:

$$J_{x0,y0} = \begin{vmatrix} \partial v_x / \partial x & \partial v_x / \partial y \\ \partial v_y / \partial x & \partial v_y / \partial y \end{vmatrix}_{x0,y0} \tag{3.3}$$

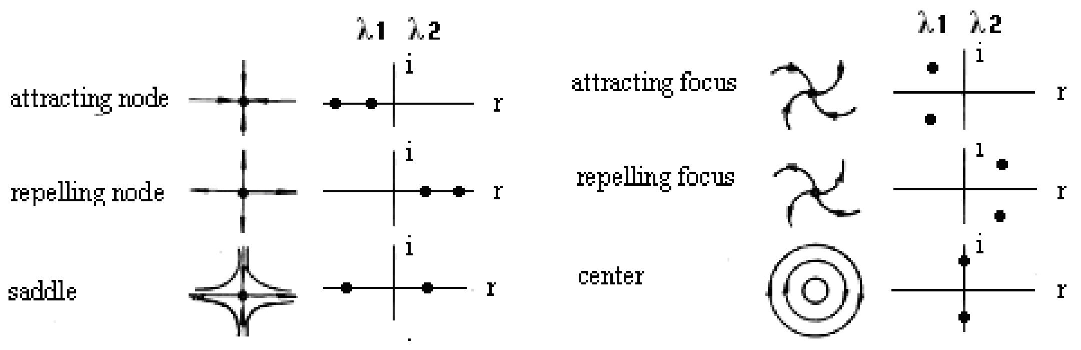
3D Jacobian matrix:

$$J_{x0,y0,z0} = \left| \begin{matrix} \partial v_x/\partial x & \partial v_x/\partial y & \partial v_x/\partial z \\ \partial v_y/\partial x & \partial v_y/\partial y & \partial v_y/\partial z \\ \partial v_z/\partial x & \partial v_z/\partial y & \partial v_z/\partial z \end{matrix} \right|_{x0,y0} \quad (3.4)$$

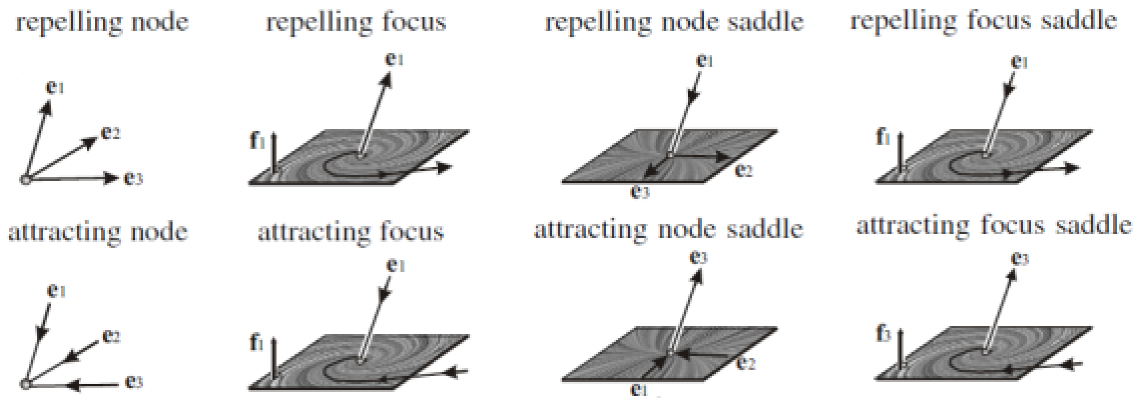
If J has full rank, the critical point is called linear or first-order.

Definition (Vector Field Topology) *Topology* consists of critical points, periodic orbits, and separatrices [28].

Figure 3.4 shows critical point classifications [27], [58]. Figure 3.5 and Figure 3.6 show examples of 2D and 3D topologies which consist of critical points connected with separatrices.



(a) 2D critical points [27]



(b) 3D critical points [58]

Figure 3.4 Critical point classification.

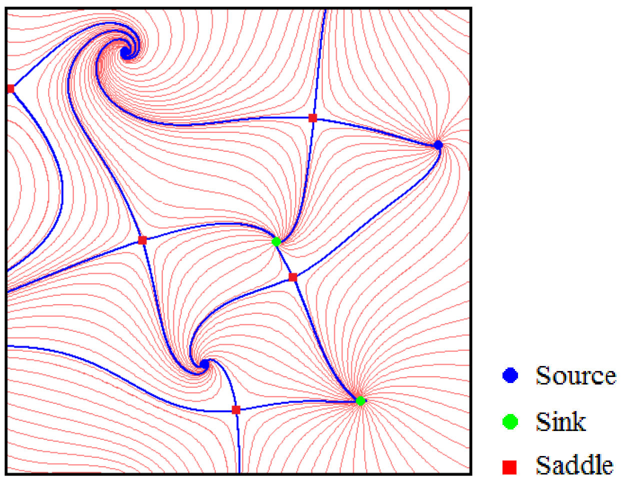


Figure 3.5 The topology of a 2D flow field. Separatrices are in blue and other streamlines are in light red.

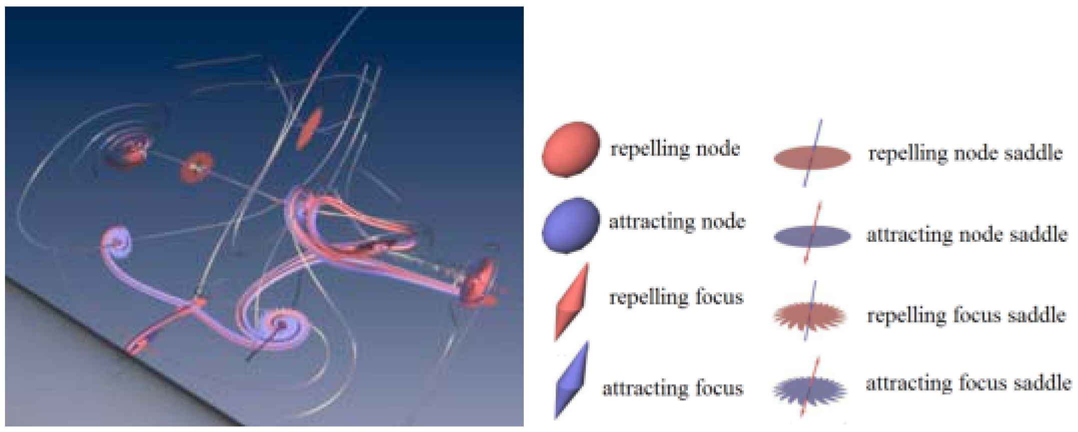


Figure 3.6 The topology of a 3D flow field [58].

CHAPTER IV

FEATURE-LEVEL UNCERTAINTY VISUALIZATION

Visualizing data uncertainties facilitates scientific studies in different areas. Though many current methods visualize uncertainty as errors that arise from different measurements, calculations, and numerical models, we study the uncertainty of features within the data.

The uncertain deviations of a feature, e. g. a minimum, a maximum, or any user defined features, are referred to as feature-level uncertainty, while the uncertainties related to the uncertain numerical values of the data, such as statistical variation, spread, errors, differences, and range, are referred to as data-level uncertainty. Visualizing feature-level uncertainty reveals the potentially significant impacts of the data-level uncertainty, which in turn helps people gain new insight into data-level uncertainty itself. For example, the uncertainty of the ocean temperature data may result in the uncertain deviation of the center of an important warm eddy. The uncertainty of the hurricane wind data may cause the uncertain location of a hurricane eye (see example in Section 1.1). This kind of uncertainty is neglected by most current methods but needs to be quantified so viewers are aware of the uncertainty.

In this chapter, we first conduct a case study showing that the uncertainty of a data set may result in remarkable positional deviations of its features; therefore, there is a clear need to visualize the feature-level uncertainty. Then we present a method to visualize feature-level uncertainties.

This chapter is organized as follows. Section 4.1 describes a case study of modeling scattered oceanic data to explore the gap between the data-level uncertainty and feature-level uncertainty. Section 4.2 describes a framework of the feature-level uncertainty visualization.

4.1 A case study of modeling scattered oceanic data

Modeling scattered field data [59] is a specific interpolation problem which reconstructs an unknown function from a collection of scattered data points. It addresses the essential problem of modeling unevenly distributed sample data onto a uniform grid so that it can be rendered using conventional grid-based visualization techniques. This section describes a case study of scattered oceanic data modeling which identifies the existence of feature-level uncertainties.

It is often not possible to establish a ground truth to estimate uncertainty. Similarly, the lack of ground truth is often a problem found in measuring the modeling precision of scattered data. Scientists therefore estimate the errors between standard functions and their reconstructed fields [59]. It is not possible to investigate all kinds of features as well. Among various features, critical points have been long recognized as important features for data analysis and for being closely related to physical features.

Usually, the precision of a modeling method is estimated by testing Root-Mean-Squared (RMS) error between certain standard test functions that exhibit a variety of behaviors with different number of critical points and their reconstructed functions from a set of scattered data [59]. There is no previous precision estimation method addressing the feature-level precision — the difference between the features in the known correct datum and an estimate. We choose to study the uncertainty related to critical points in the reconstructed field. Through this case study, we launch a probe into the uncertainty related to features, which is often a missing part in uncertainty visualizations.

4.1.1 A real-world scenario

Consider the following scenario: we have a set of scattered values of ocean temperature and need to reconstruct a continuous scalar field from it. We could never know the exact reconstruction error—the difference between the reconstructed temperature field and the actual temperature field other than at the sample locations. However, there are two ways to estimate such uncertainties when a comparison to a ground truth is impossible. (1) We can test the modeling precision through standard functions. Given a sample point p in scattered data set, assign it a value from the standard function $f(p)$. Then, we use an interpolation method to rebuild a continuous field from it. The reconstruction precision can be estimated as the difference between the reconstructed field and the standard function. We call this the vertical comparison method. (2) We can calculate the difference among the reconstructed data from different modeling methods. We call this the lateral comparison method. In our study, two interpolation methods, Modified Quadratic Shepard's method (MQS) [60] and Multiquadratic method [59], are used to reconstruct a continuous data field from scattered data sets. The reconstruction precision can be estimated as the difference between the two reconstructed fields from these two methods. Commonly, RMS (Root-Mean-Squared) error is computed as modeling precision. We use it as data-level errors. The feature-level errors are estimated by the difference in the number and position of the critical points between the original field and rebuilt field before we introduce a more sophisticated measure of feature-level difference in section 4.2.

This section briefly discusses the interpolation methods and studies the reconstruction precision in both data-level and feature-level.

4.1.2 Scattered data interpolation

Let Ω denote a continuous 2D domain, $\Omega = \{P = (x, y) \mid a \le x \le b, c \le y \le d\}$. $P_i = (x_i, y_i)(i = 1,\ldots,n)$ are discrete sampling points distributed over Ω with associated scalar values f_i. A continuous function $F(P) = F(x, y)$ is constructed such that $F(P_i) = F(x_i, y_i) = f_i (1 \le i \le n)$.

(1) Multiquadric Method [59]: Due to its high precision and easy implementation, the multiquadric method is a preferred choice for a scattered data set with less than 500 samples. The form of modeling function is $F(P) = \sum_{i=1}^{n} a_i \times \sqrt{R^2 + \| P - P_i \|^2}$ where $\| P - P_i \| = \sqrt{(x - x_i)^2 + (y - y_i)^2}$. The typical value of R is 0.1. The coefficients $a_1, a_2, \ldots, a_n$ are given by the solution of the following equation: $(\sqrt{R^2 + \| P_i - P_j \|^2})$ $(a_1, a_2, \ldots, a_n)^T = (f_1, f_2, \ldots, f_n)^T \ (1 \le i, j \le n)$.

(2) Modified Quadratic Shepard's (MQS) Method [60]: MQS Method is a major interpolation method for a scattered data set with a size larger than 500 that has relatively high precision. The form of modeling function is

$$F(x, y) = \frac{\sum_{k=1}^{n} W_k(x, y) Q_k(x, y)}{\sum_{i=1}^{n} W_i(x, y)}$$ where $Q_k(x, y)$ is a quadratic polynomial, a local estimate of f at (x_k, y_k) with $Q_k(x, y) = f_k$; $W_k(x, y) = \left[\frac{(R_w - d_k)_+}{R_w d_k} \right]^2$ with $(R_w - d_k)_+ = \begin{cases} R_w - d_k, (d_k < R_w) \\ 0, (d_k \ge R_w) \end{cases}$, $d_k(x, y) = \sqrt{(x - x_k)^2 + (y - y_k)^2}$, and R_w is radius of influence of (x_k, y_k).

4.1.3 Error estimation

(1) Data-level error: RMS error is used to compute the modeling precision. Nielson provided several standard functions [59] to test RMS error. A set of scattered sample values are computed by the standard functions. The interpolation error is $RMS = \sqrt{\frac{\sum_{i=1}^{Ni} \sum_{j=1}^{Nj} [F(\frac{i}{N_i}, \frac{j}{N_j}) - F^{'}(\frac{i}{N_i}, \frac{j}{N_j})]^2}{N_i N_j}}$ where F is a test function, and $F^{'}$ is the interpolation function. Assume $N_i = N_j = 100$, they give the resolution of the evenly re-sampled data across the original and rebuilt continuous data fields. F and $F^{'}$ could be two interpolation functions if one intends to compare between two interpolation functions.

(2) Feature-level error: In this section, feature-level errors are estimated by the difference between the topological features, namely, the number and position difference between the critical points of the original and rebuilt fields. Let F be a test function and $F^{'}$ be the interpolation function. The critical points of F and $F^{'}$ are $C = \{c_0, c_1, ..., c_n\}$ and $C^{'} = \{c_0^{'}, c_1^{'}, ..., c_m^{'}\}$, respectively. The difference in critical point numbers is $|n - m|$.

Similar to the concept of Earth Mover's Distance (EMD) [61], which estimates the amount of work necessary for transforming one object into the other, the position differences of the critical points are defined as:

$$Distance(C, C^{'}) = \begin{cases} \sqrt{\sum_{i=0}^{n} distance(c_k, C^{'})}, n \geq m \\ \sqrt{\sum_{i=0}^{m} distance(c_k^{'}, C)}, m \geq n \end{cases} \quad (4.1)$$

$distance(c_i, C)$ is the shortest Euclidean distance found between a critical point c_i to a critical point in *C*. The length and width of the data field are normalized to [0,1].

4.1.4 Experiment results and discussion

We use a layer of temperature data from a 1998 ocean field investigation data set. The samples are taken along a ship. We implement two methods for interpolating scattered samples. We would like to know the modeling error of the reconstructions by using vertical comparison and lateral comparison.

(1) Vertical Comparison Method: Vertical comparison compares the standard functions with reconstructed data. The experiment is conducted on 5 standard functions [59] listed below. The sample locations are provided by one layer of a field investigation data (Figure 4.1). The number of sample points is 86. The data-level error and feature-level error are shown in the table 4.1 and 4.2.

$$\begin{aligned}F_1(x,y) &= 0.75\exp(-((9x-2)^2+(9y-2)^2)/4)\\ &+0.75\exp(-(9x+1)^2/49-(9y+1)/10)\\ &+0.50\exp(-((9x-7)^2+(9y-3)^2)/4)\\ &-0.20\exp(-(9x-4)^2-(9y-7)^2)\end{aligned} \tag{4.2}$$

$$F_3(x,y) = (1.25+\cos(5.4y))/(6+6(3x-1)^2) \tag{4.3}$$

$$F_5(x,y) = \exp(-20.25((x-0.5)^2+(y-0.5)^2))/3 \tag{4.4}$$

$$F_7(x,y) = 2\cos(10x)\sin(10y)+\sin(10x) \tag{4.5}$$

$$\begin{aligned}F_9 &= ((20/3)^3\exp((10-20x)/3)\exp((10-20x)/3))^2\\ &*((1/(1+\exp((10-20x)/3)))(1/(1+\exp((10-20y)/3))))^5\\ &*(\exp((10-20x)/3)-2/(1+\exp((10-20x)/3)))\\ &*(\exp((10-20x)/3)-2/(1+\exp((10-20y)/3)))\end{aligned} \tag{4.6}$$

Table 4.1 Interpolation Error of Multiquadric Method

Test functions	F_1	F_3	F_5	F_7	F_9	Mean
RMS	0.0132	0.0083	0.0017	0.0156	0.0051	0.0088
n	4	3	1	23	5	
m	8	8	1	22	14	
$\|n-m\|$	4	5	0	1	9	5
$Distance(C,C')$	0.1933	0.1530	0.0081	0.1207	0.1367	0.1216

* n is the number of critical points in a test function F; m is the number of critical points in the interpolation function F'; C and C' are the sets of critical points of F and F', respectively.

Table 4.2 Interpolation Error of MQS Method

Test functions	F_1	F_3	F_5	F_7	F_9	Mean
RMS	0.0162	0.0141	0.0029	0.0176	0.0089	0.0119
n	4	2	1	23	5	
m	25	15	35	51	30	
$\|n-m\|$	21	13	34	28	25	26.2
$Distance(C,C')$	0.1629	0.1718	0.1744	0.1601	0.1657	0.1669

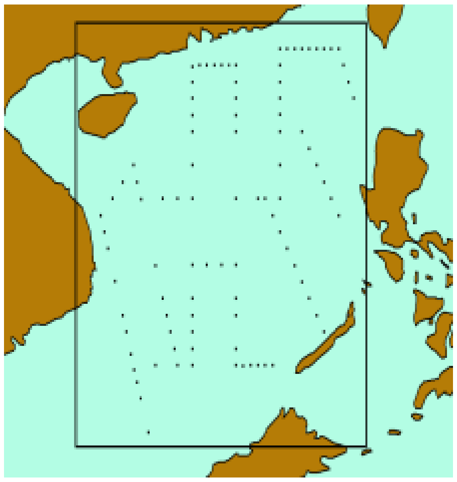


Figure 4.1 1998 field investigation data set of the South China Sea. The test sample locations are extracted from the layer at a depth of 150 meters.

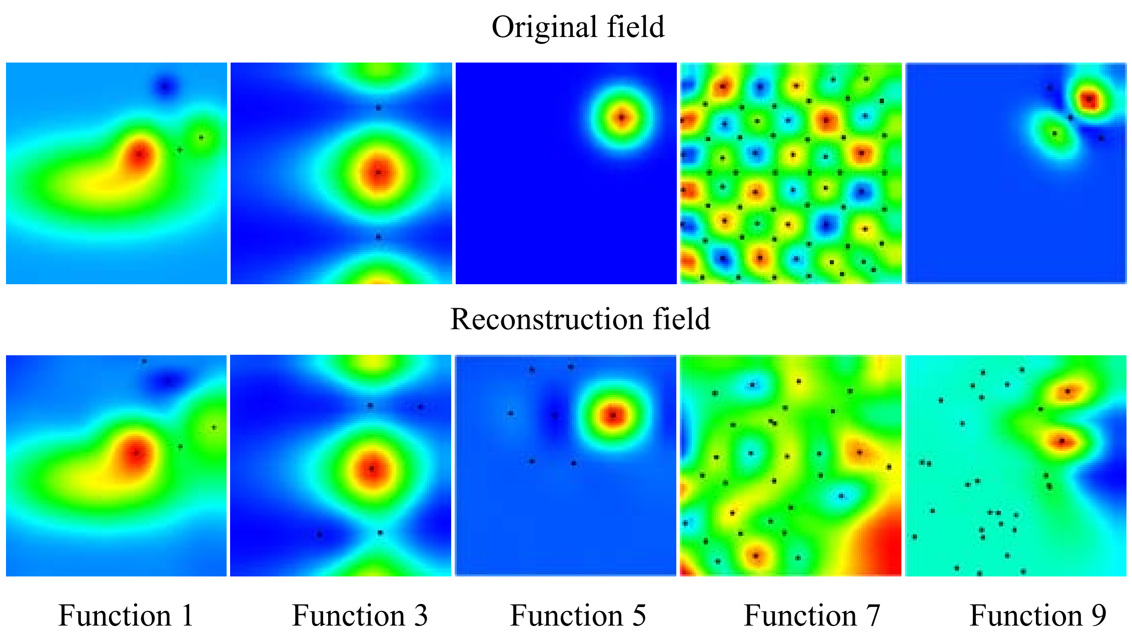
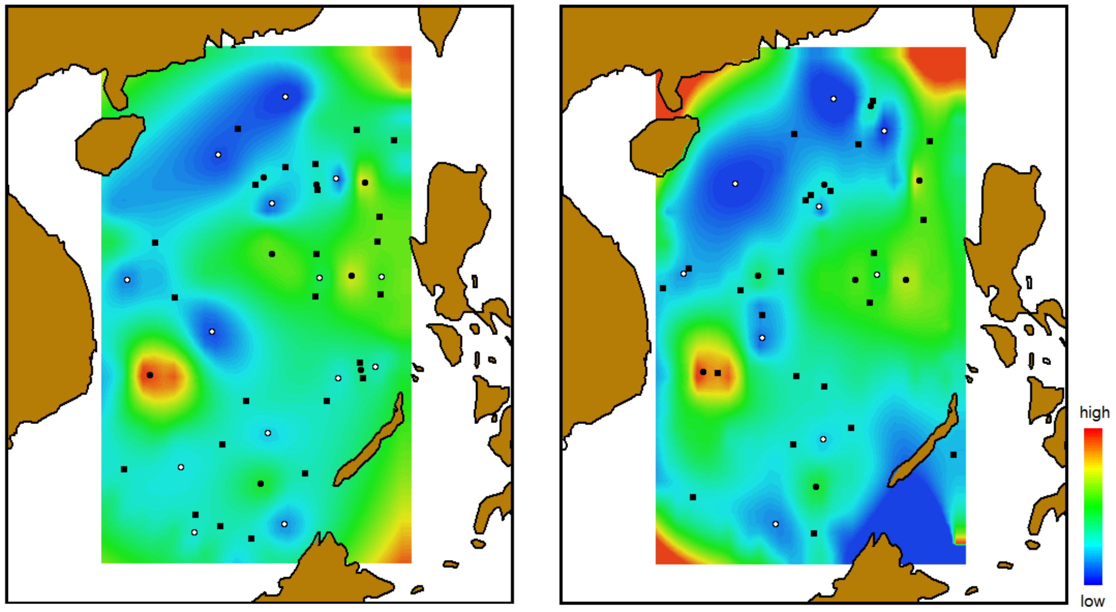


Figure 4.2 Vertical comparison: compare original scalar fields with reconstructed scalar fields. The function values are mapped onto a rainbow colormap, where each function value is mapped to a color value interpolated between blue and red. The black points indicate critical points.

(2) Lateral Comparison Method: Lateral comparison estimates the difference between reconstructed fields by different interpolation methods. Given a set of sample data (Figure 4.1) from field study, one does not know the ground truth but only estimates it from reconstructed fields using different interpolation methods. In this case, the 5 standard functions are not needed. The data-level error and feature-level error are shown in Table 4.3.



Multiquadric reconstructed field MQS Reconstructed Field

Figure 4.3 Lateral comparison: compare two temperature fields reconstructed by MQS method and Multiquadric method.

Table 4.3 Comparison between Multiquadric Method and MQS Method

RMS	m_1	m_2	$\lvert m_1 - m_2 \rvert$	$Distance(C_1, C_2)$
0.023	46	40	6	0.073

* m_1 is the number of critical points in field F_1 (reconstructed by Multiquadric method); m_2 is the number of critical points in field F_2 (reconstructed by MQS method); C_1 and C_2 are the sets of critical points of F_1 and F_2, respectively.

As indicated in Figures 4.2 and 4.3 and Tables 4.1 and 4.2, no matter what kind of comparison, vertical comparison or lateral comparison, is used, the amount and position deviation of the critical points (the black points in the figures) are considerable though the RMS error appears to be fairly low. The average position deviations between critical points are as high as 0.1669. The feature-level errors show that the locations of critical points are uncertain, depending on which interpolation methods are used.

This case study reveals that the feature-level errors could be significant and that they need to be revealed to users.

4.2 A framework for feature-level uncertainty visualization

In many cases, locations of features matter more than the data-level uncertainty. For instance, the locations of warm eddies are important in ocean fishery, and the locations of hurricane eyes or the peaks in pressure field are important in weather analysis. The case study in section 4.1 reveals that significant deviations of the features may exist in the data with seemingly low data-level uncertainty, and thus indicates a pressing need to measure the feature-level uncertainty.

It is believed that the feature-based technique is desirable when the size and complexity of the data increases [3]. Visualizing the feature-level uncertainty instead of the data-level uncertainty of the whole data may provide a solution to the uncertainty visualization of the large scale data. Feature tracking methods are proposed to map the evolving features over time. Two features are considered the same feature at different time slices if they share the same tracking path [28], [44] or the biggest similarity [42]. This inspires us to evaluate the uncertainty related to features by comparing the deviation of those feature pairs in different data sets.

4.2.1 Method overview

The impact of uncertainty on the features is quantified as feature-level uncertainty which is measured by feature deviation. The feature deviation is obtained through a three-step procedure — feature identification, feature mapping, and uncertainty representation. Given a set of data members, e. g. multiple simulation runs, this method first identifies the features within all the data members and the mean field given by averaging all the members. Second, feature tracking is implemented to map the features of each data member to that of the mean field. The mapped features are then assumed to have the same feature with slight position deviation in each of the individual data members. Finally, the feature-level uncertainties are expressed as the deviations of the features.

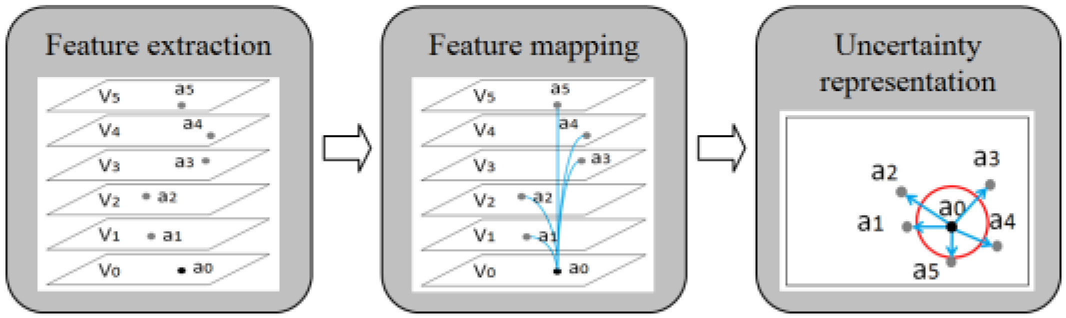


Figure 4.4 The pipeline for feature-level uncertainty visualization.

The features that we currently study are vector field critical points. They are intuitive features closely related to physical features [3]. Scalar fields can be analyzed through their gradient fields.

Section 4.2.2, 4.2.3, and 4.3.4 discuss feature extraction, feature mapping, and uncertainty glyph design respectively; Section 4.2.5 demonstrates results; Section 4.2.6 concludes this chapter with possible improvements.

4.2.2 Feature extraction

The computation of critical points in a vector field can be found in [27].

For a scalar field, its gradient field can be used to extract critical points so that the features of scalar fields and vector fields are analyzed in the same way. A vector field V can be constructed out of scalar field f using the gradient operator $\nabla : V = \nabla f = (\partial f / \partial x, \partial f / \partial y)$. The maxima of f appear as sinks and minima appear as sources in its gradient field V. Figure 4.5 illustrates an example of the critical points extracted from the gradient field of a temperature field.

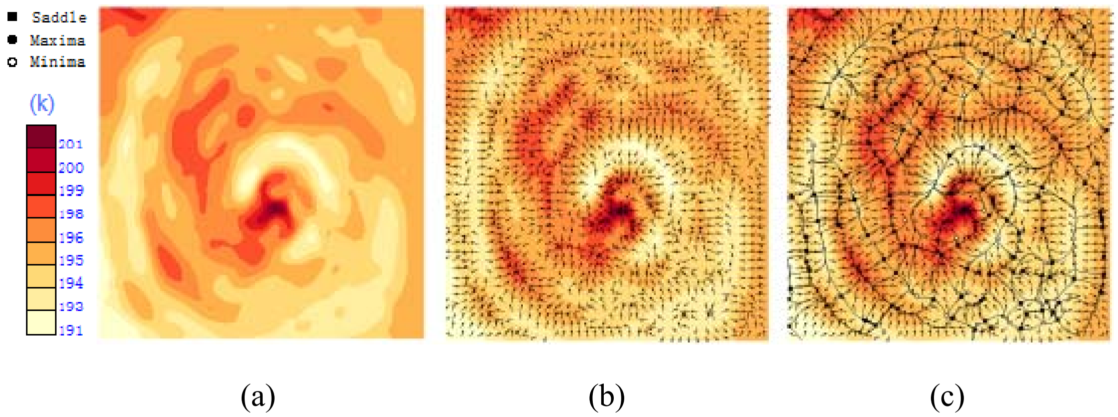


Figure 4.5 Topology extracted from the gradient field of a temperature field.(a) Color-mapped temperature. (b) Gradient field represented with arrows. (c) Gradient field with extracted vector topology.

4.2.3 Feature mapping

Feature tracking is a way to investigate feature evolution of time-varying data. Some feature-tracking methods correlate features by measuring their positional distance or overlapped area while others utilize an intermediate data field derived from two neighboring time slices and trace paths along critical points within it [28], [44], [45]. Theisel's Feature Flow Field (FFF) method [44] is especially interesting to us because it provides a generic approach to feature tracking. Independent of underlying grids, the FFF method captures the temporal evolution of a feature using stream object integration in a derived Feature Flow Field. The concept of FFF has been successfully applied to tracking critical points [46], extracting Galilean invariant vortex core lines [47], simplification [48], and comparison [49].

Feature Flow Field (FFF) [44] is adopted to couple critical points of different fields by tracing streamlines within it. It is used to identify the same feature that appears in different data members at different positions. The uncertainty of this feature is then expressed as the deviation between all of its counterparts in different data members.

4.2.3.1 Feature flow field construction

The main idea of the FFF approach is to introduce an appropriate vector field f in space-time, such that a feature tracking in a 2D time-dependent vector field V corresponds to a streamline integration in f. Consider tracking critical points in a 2D time-dependent vector field, which is given as $V(x,y,t)=\begin{pmatrix} u(x,y,t) \\ v(x,y,t) \end{pmatrix}$. V can be constructed by applying a linear interpolation of V_1 and V_2: $V(x,y,t)=(1-t)V_1(x,y)+tV_2(x,y)$.

To get FFF f, one searches in space-time for the direction in which both components of V locally remain constant. This direction is perpendicular to the gradients of the two components of V. This leads to $f(x,y,t)=grad(u)\times grad(v)=\begin{pmatrix} u_x \\ u_y \\ u_t \end{pmatrix}\times\begin{pmatrix} v_x \\ v_y \\ v_t \end{pmatrix}=\begin{pmatrix} \det(v_y,v_t) \\ \det(v_t,v_x) \\ \det(v_x,v_y) \end{pmatrix}$. Figure 4.6a illustrates that a feature is tracked by integrating a streamline within a Feature Flow Field. Let two vector fields, V_0 and V_i, be two slices of a time-varying flow. A streamline which starts from one feature a_0 in vector field V_0 reaches another feature a_i in vector field V_i. a_0 and a_i are therefore recognized as one feature that evolves over time between V_0 and V_i

4.2.3.2 Feature-level uncertainty measurement

Given k data members $V_i, (i = 1, ..., k)$, a mean field V_0 is first computed as the average of them. V_0 is then paired with each data member V_i. Feature mapping is implemented for each data pair V_0 and V_i. With the FFF method, features of different vector fields could be correlated.

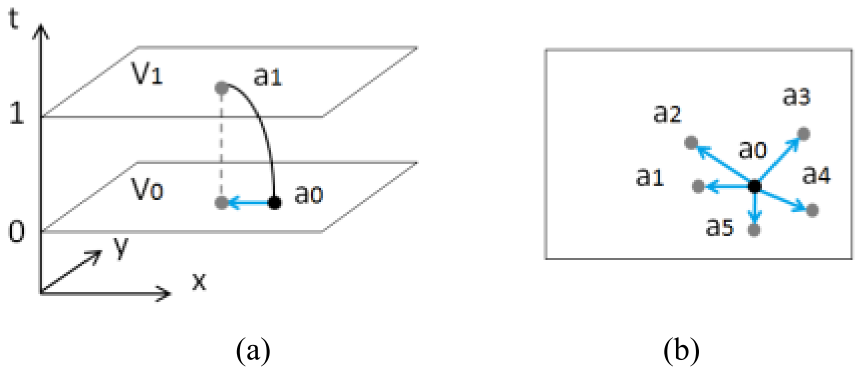


Figure 4.6 Feature-level uncertainty measurement. (a) Feature deviations detected by tracing critical points within FFF between a data member V_1 and the mean field V_0. (b) Feature-level uncertainty measured by the deviations (indicated by arrows) of features (indicated by dots) in all the data members $V_n (n = 1, ..., k)$ from the features in the mean data field V_0.

Figure 4.6 demonstrates how to measure feature-level uncertainty related to a feature. For a data member V_1 and the mean field V_0, we trace a streamline from a critical point a_0 in V_0 until it reaches a critical point a_1 in V_1. After tracing critical points between all the pairs, the feature-level uncertainty is measured by the distances between $a_i (i = 1, ..., k)$ and a_0. Figure 4.6 illustrates the feature mapping between a pair of data.

Figure 4.6b shows a straightforward representation of the uncertainties related to individual features by arrows. Given a data member $V_n (1 \le n \le k)$, it is possible that the streamline starting from a_0 reaches the boundary of the FFF or ends at V_0 instead of reaching a critical point in V_n. In these cases, we assume that the mapped critical point a_n for a_0 in this data member exists outside the domain. Therefore, we set its distance from a_0 a large value — the maximum distance found between a_0 and all the mapped critical points a_i.

4.2.4 Uncertainty glyph design

Glyph design addresses the central problem of how uncertainty information is processed into knowledge. For the detected deviations of a critical point, a quantitative glyph is designed to indicate uncertainty level related to the critical point. The new uncertainty glyph is inspired by both graduated circular glyph [13] and elliptical glyph [62].

4.2.4.1 Graduated circular glyph

Sanyal et al. [12] identified that glyphs altered by size are effective in depicting uncertainty in 2D datasets. They introduced a graduated circular glyph that encodes the deviation of each ensemble member from the ensemble mean. A glyph that has a dense core with a faint periphery indicates that ensemble members have a few outliers and mostly agree. A mostly dark glyph indicates that large differences exist among individual members. The size of a glyph indicates the variability of a location with respect to other locations on the grid. Consequently, graduated glyphs provide a straightforward way to visualize uncertainty locally and globally. For more details on graduated glyphs, we refer the reader to [13].

4.2.4.2 Elliptical glyph

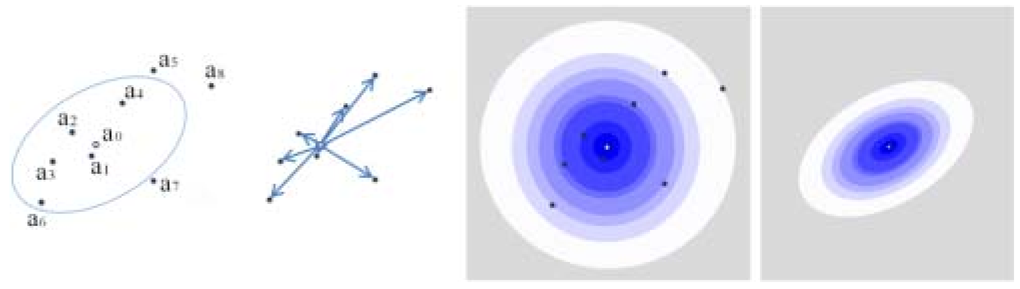
A glyph that can be used at different levels is the elliptical glyph [62]. It depicts the covariance between multiple real-valued random variables X_i. In probability theory and statistics, covariance measures how much two variables change together. The covariance matrix $\sum$ generalizes the notion of variance to multiple dimensions. $\Sigma_{i,j} = Cov(X_i, X_j) = E[(X_i - E(X_i))(X_j - E(X_j))]$ where $E(X_i)$ is the expected value of X_i, and the ellipse axis length is $\lambda = eig(\Sigma)$. Ellipse axis directions are given by $d = eigvec(\Sigma)$. An elliptical glyph can be applied to visualize tensors, but can also show a simplified representation of the spatial distribution of a set of 2D or 3D data [24].

4.2.4.3 2D graduated elliptical glyph

Before using the graduated ellipse, we considered using arrows to indicate the uncertainty by showing directions and distances of individual deviations. Nevertheless, it is found that arrows, though showing the deviations in an authentic way, could cause severe information overload when the number of ensemble runs increases. Contrarily, the graduated ellipse possesses the elliptical glyph's ability to depict the overall deviation of a feature and the graduated glyph's intuitive way to depict inner deviation of individual ensemble members.

Let $a_0\ (x_0, y_0)$ be a critical point in the mean data field V_0. Its counter-parts in data members V_i are $a_i\ (x_i, y_i)$ $(i = 1,...,k)$. A graduated elliptical glyph consists of k nested ellipses and is placed at the location of a_0. The nested ellipses share the same orientation and axis ratio. The detail of rendering each nested ellipse is as follows:

First, sort a_i according to its distance from a_0, $d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$, in descending order. Next, assign the outmost ellipse E with axes A and B computed according to the relative locations of a_i towards a_0, $(x_i - x_0, y_i - y_0)$. Let variable X be $x_i - x_0$, and Y be $y_i - y_0$. The lengths and directions of A and B are given by $eig(\sum(X,Y))$ and $eigvec(\Sigma(X,Y))$, respectively. $(\Sigma(X,Y))$ is the Covariance Matrix of X and Y. Then, the rest of nested ellipses are produced by fitting them into ellipse E. Let the biggest distance among d_i be D. Each nested ellipse is given axis lengths $|A| \times d_i / D$ and $|B| \times d_i / D$. Finally, in a way similar to producing a graduated circular glyph [13], assign saturation level s_i for the ith glyph as $s_i = (i-1)/(k-1)$ and overlay the ellipse so that the smaller one is over the larger one. The graduated elliptical glyph shows the overall deviation of a feature and the inner distribution among the feature deviations of individual data members. When placed across an image, the overall size and orientation of the glyphs indicate the variability of features while the very shape of and color distribution within an individual glyph give a quick statistical summary of uncertain deviations of a feature.



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

Figure 4.7 Feature-level uncertainty glyph design. (a) Ellipse. (b) Arrows. (c) Graduated circular glyph. (d) Graduated ellipse

Figure 4.7 gives a comparison between using a simple ellipse, arrows, graduated circular glyph, and graduated elliptical glyph for feature-level uncertainty. The individual features $a_i, (i = 1, \ldots, k)$ (which are not displayed in a final visualization) are shown as well to better illustrate the design concept of each glyph. The simple ellipse (Figure 4.7a) only characterizes the overall deviation of a feature. Arrows (Figure 4.7b) indicate the exact locations of all the deviated features while inviting visual clutters when k increases. The graduated circular glyph (Figure 4.7c) shows the individual deviation of a feature but no direction information is revealed. However, the graduated elliptical glyph (Figure 4.7d) summarizes overall and individual distribution of a feature in a succinct way. Figure 4.8 shows a set of graduated ellipse with varying color distributions, sizes, axis length ratios, and orientations.

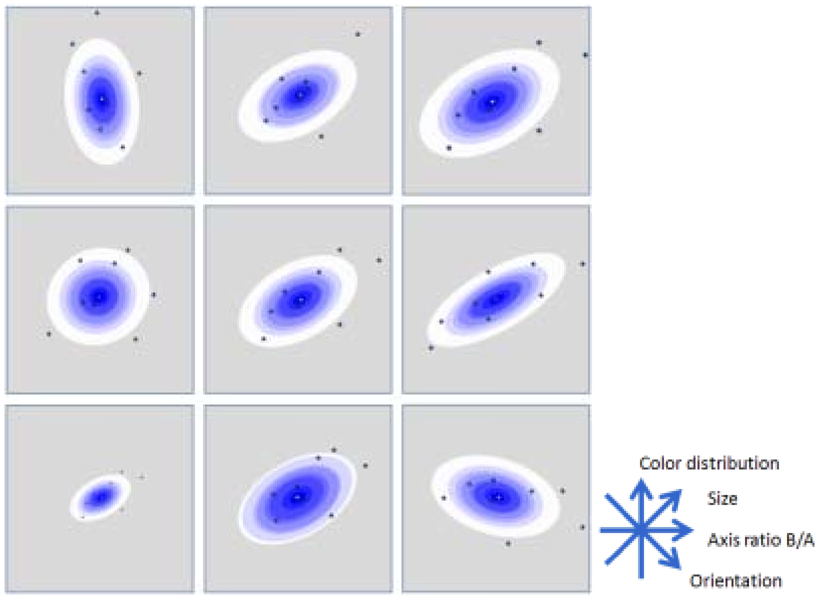


Figure 4.8 Graduated ellipses with varying orientation, saturation distribution, size, and axis length ratio B/A. 8 data members are used.

4.2.4.4 3D graduated elliptical glyph

A 3D elliptical glyph consists of k nested ellipsoids and is placed at the location of a_0. The nested ellipsoids share the same orientation and axis ratio. To show the distribution among data members, a wedge is clipped out of it. Assign the outmost ellipsoid E with axes A, B, and C computed according to the relative locations of a_i

towards a_0, $(x_i - x_0, y_i - y_0, z_i - z_0)$. Let variable X be $x_i - x_0$, Y be $y_i - y_0$, and Z be $z_i - z_0$. The lengths and directions of A, B and C are given by $eig(\sum(X,Y,Z))$ and $eigvec(\Sigma(X,Y,Z))$, respectively, where $\Sigma(X,Y,Z)$ is the Covariance Matrix of X, Y, and Z. Then, similar to the manner of producing a 2D graduate elliptical glyph, the rest of the nested ellipsoids are produced by fitting them into ellipsoid E and are rendered in blue with different saturation levels. The ellipsoids are opaque because we found in the experiment that transparency adds confusion and blur into the images, and is not effective in conveying depth information. Figure 4.9 shows 3D graduated elliptical glyphs in different sizes and orientations.

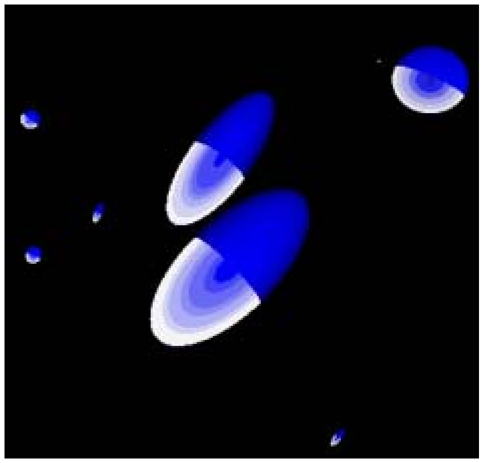
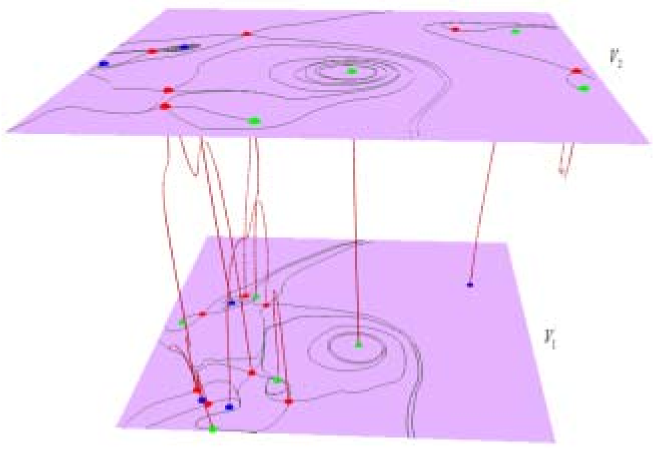


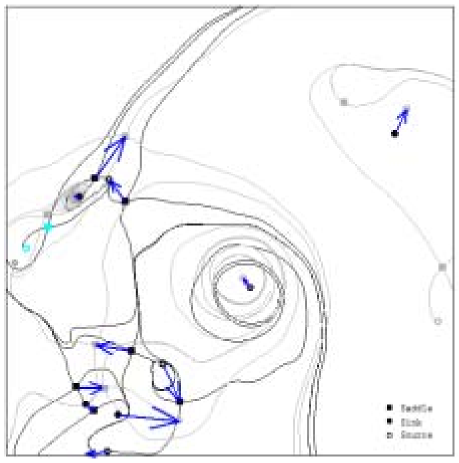
Figure 4.9 3D graduated elliptical glyphs.

4.2.5 Results and discussion

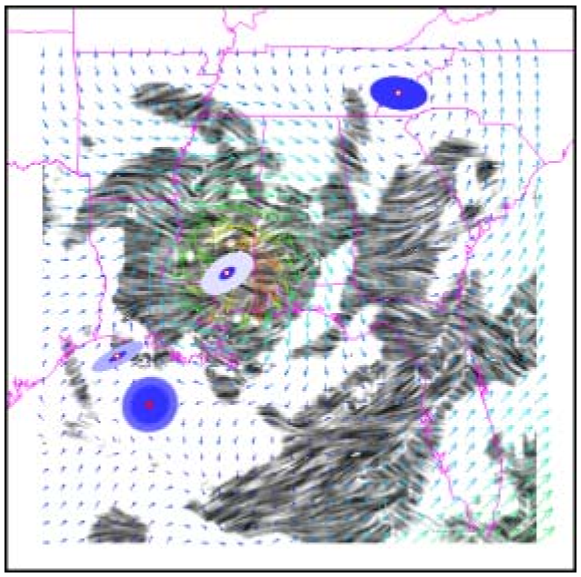
The method is applied to a 2D vector dataset and 3D scalar dataset. The first dataset includes 5 simulated hurricane wind fields (Figure 4.10). The second dataset contains water vapor data (Figure 4.11) from 8 WRF (Weather Research and Forecast Environmental Modeling System) simulation runs. The uncertainty glyphs are placed in critical point locations. The results demonstrate how much features are affected by the uncertainty within the data. Through mapping and comparing the critical points between different ensemble members, the shifts between critical points become perceivable. The graduated elliptical glyphs effectively indicate the magnitude and overall orientation of the uncertain deviations of vortices. Especially in the hurricane wind data (Figure 4.10), the uncertain position of the hurricane eye reflects the impact of the uncertainty directly. A side-by-side display of different component data or the visualization of the data-level uncertainty may not give viewers such insight.



(a)



(b)



(c)

Figure 4.10 Feature-level uncertainty of a hurricane wind field. (a) Feature tracking within FFF of two vector fields V_1 and V_2. (b) Feature deviations (indicated by arrows) between two vector fields. Overlapped topology of V_1 (black) and V_2 (gray) are shown as well. (c) Uncertain location of hurricane eye and vortices in a hurricane wind field (5 ensemble members).

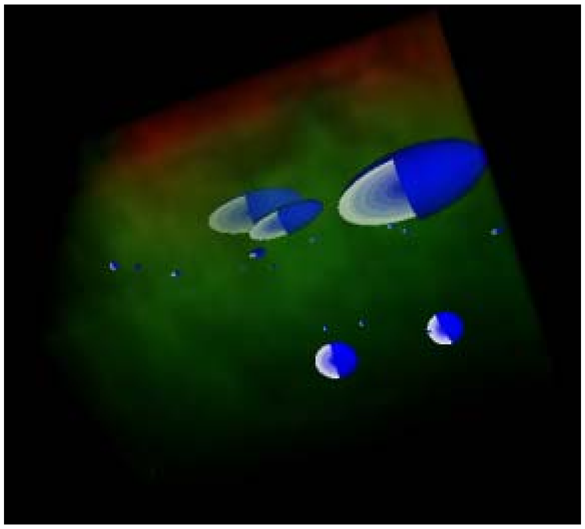


Figure 4.11 3D WRF water vapor data and uncertainty glyphs.

4.3 Conclusion

We presented a framework of feature-level uncertainty for both scalar and vector data, which is composed of feature extraction, feature deviation computation, and uncertainty representation. An elliptical glyph is designed to represent feature level uncertainty. The compactness of the feature-level uncertainty representation may provide a way to ease the perception issue of 3D visualization. The presented framework has potential for broader applications which include different types of features. Hopefully, this work will bring awareness to the existence of feature-level uncertainty.

Although the result of this feature-level uncertainty visualization is positive, there are a few limitations and areas that need further study. Most notably, more features could be considered in the future. Second, other feature-mapping methods may be included depending on the feature type since the current feature-mapping method, FFF, mainly tracks topological features.

CHAPTER V
AN INTERACTIVE CONTOUR TREE BASED VISUALIZATION FOR EXPLORING DATA WITH UNCERTAINTY

This chapter presents an interactive visualization tool based on contour trees for exploring 2D and 3D data with intuitive and quantitative uncertainty representations.

Section 5.1 summarizes the background, main idea, and contributions of the proposed method. Section 5.2 discusses contour tree simplification. Section 5.3 describes the contour tree layout and tree view graph design based on contour tree simplification. The visualization of three levels of uncertainties is discussed in section 5.4. Section 5.5 discusses the interface design. Section 5.6 demonstrates experimental results. Section 5.7 draws some conclusions and discusses possible areas of improvement.

5.1 Method overview

Many current uncertainty visualizations focus on encoding uncertainty information into different graphics primitives, such as color, glyph, and texture, which are attached to surfaces or embedded in 3D volumes [1]. These techniques may be subject to cluttered display, occlusion, or information overload due to the large amount of information and interference between the data and its uncertainty (as shown in Figure 1.2). We believe that one promising direction to cope with this challenge is to allow users to explore data interactively and to provide informative clues about where to look.

Contours, including iso-lines and iso-surfaces, are features frequently investigated for exploring data with uncertainty [13], [15]. For instance, uncertainty in climate modeling is often represented by ensembles that contain multiple results for the simulated quantities. Rendering contours from all ensembles in a single image, known as spaghetti plots [63], is a conventional technique used by meteorologists for observing uncertainty in their simulations. Users are often interested in the uncertainty of the contours, e. g. how different are the contours of different ensemble members? This chapter focuses on new methods for visualizing the uncertainties in scalar fields. Though many uncertainty visualization techniques have been developed for scalar fields, the uncertainty related to global features, e. g. the topological structure of the scalar data, is barely studied. We classify uncertainty information into three categories: data-level uncertainty that indicates the uncertainty of the data, contour-level uncertainty that represents the positional variation of the contours, and topology-level uncertainty that reveals the uncertainty of the topology in the data.

The contour tree has been exploited as an efficient data structure to guide exploratory visualization. Although it has rarely been used for uncertainty visualization before, we identify it as a desirable tool for an interactive visualization of data with uncertainty. First, a compact and clutter-free uncertainty visualization is achieved by attaching uncertainty glyphs to simplified contour trees. This provides an effective solution to the long-standing perception issues such as clutter and occlusion in many uncertainty visualizations in 3D or large 2D scenes. Secondly, a contour tree stores the information related to the geometry of individual contours, which can be utilized to compute the contour-level uncertainty. Thirdly, investigating the unstable structure of the contour tree reveals the uncertain topology of the data. Further, a contour tree provides a flexible interface that allows users to interactively select contours that interest them, e. g. those with high or low uncertainty. Moreover, contour tree simplification facilitates a high level overview of a scalar field along with its uncertainty. Particularly, a simplified contour tree attached with uncertainty glyphs reduces the workload in viewing and analyzing 3D data or complicated 2D data with uncertainty.

The core of this method is the use of contour trees as a tool to represent uncertainty and to select contours accordingly. First, a new contour tree simplification with efficient top-down sub-tree decomposition is proposed as an alternative to the traditional bottom-up simplification with branch pruning. Second, based on the new contour tree simplification, an easy-to-use contour tree-based interface takes the form of a tree view graph which is constructed by a new balanced planar hierarchical contour tree layout. The new contour tree layout emphasizes height information with few self-intersections. The new contour tree display allows a user to quickly navigate through the level of details of the data. Further, attaching uncertainty information to the planar layout of a simplified contour tree is a key to avoiding the visual cluttering and occlusion of viewing uncertainty within volume data or complicated 2D data.

The main contributions include (1) a planar contour tree layout which suppresses the branch crossing and integrates with tree view interaction for a flexible navigation between levels of detail for contours of 3D or large 2D data sets and (2) a new paradigm of investigating and visualizing uncertainty based on a contour tree that integrates the uncertainty representations on the data-level, the contour-level, and the topology-level.

5.2 Contour tree simplification

Simplification is introduced to deal with the contour trees that are too large or complicated to be studied or displayed directly [39], [40]. This applies to the data with uncertainty. Moreover, contour tree simplification facilitates a high level overview of a scalar field along with its uncertainty. Particularly, a simplified contour tree attached with uncertainty glyphs reduces the workload in viewing and analyzing 3D data or complicated 2D data with uncertainty.

This section first discusses issues of contour tree simplification in section 5.2.1; then introduces a new top-down simplification in section 5.2.2 as a competitive alternative to the usual bottom-up simplifications.

5.2.1 Contour tree simplification criteria and de-strangulation

Usually, a contour tree simplification is conducted in a bottom-up manner by successively removing branches that have a leaf node (extremum) and an inner node (saddle). Takahashi et al. [64] identified similar candidates in 3D data (maximum—2-saddle, minimum—1-saddle, and 1-saddle—2-saddle) to be removed. There exist several criteria for contour tree simplification such as topology integrity [15], importance ranking [40], and strangulation avoidance.

5.2.1.1 Contour tree simplification criteria

Topology integrity ensures that the simplified topology is consistent with the original one. Takahashi et al. [57] stated that the critical points must maintain topological integrity by satisfying the Euler formula. They suggested connecting all the boundary vertices to a virtual minimum with value $-\infty$. Figure 5.1 shows 2D and 3D examples of contour tree with virtual minimum. A 2-manifold M added with virtual minimum is a topological 2D sphere [57] whose Euler formula states that the number of critical points of a 2D sphere satisfies #{maxima} – #{saddles}+#{minima}=2. By adding a virtual minimum to the volume function, a volume dataset becomes a topological 3D sphere [64] whose Euler formula satisfies #{maxima} − #{2-saddles} + #{1-saddles} − #

{minima}= 0. It is easy to prove that, for a given contour tree that satisfies the Euler equation, a simplified contour tree through cancellation of saddle-extremum pairs (saddle-maximum or saddle-minimum pairs in 2D cases and 2-saddle-maximum or 1-saddle minimum pairs in 3D cases) satisfies the Euler equation as well.

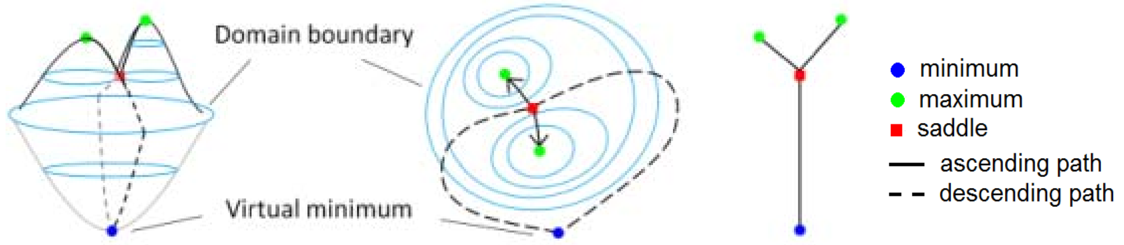


Figure 5.1 2D (left) and 3D scalar (middle) fields and their corresponding contour tree (right) with virtual minima. Contours are shown in light blue.

Importance Ranking is used to decide which pair should be removed before others. A frequently used importance measure is persistence — the absolute difference in function value spanned by a feature which is usually a pair of saddle and extremum [40]. Optionally, other measures, such as enclosed area or volume [40], can be adopted as importance measures. Bremer [56] referred to persistence as topological error norm to measure the error introduced in simplification.

Strangulation is avoided in MS-complex simplification [30] so that any other critical points are unaffected after a pair cancellation. In an MS-complex, strangulation happens when a saddle is incident to an extremum twice. Figure 5.2a shows a strangulation where saddle s is connected to u twice. There are similar strangulation cases in a contour tree when an extremum u appears to be the only upper or lower node incident to a saddle s in the contour tree. We give the following statement and a short proof of it.

Statement 1. The case that an extremum u is the only upper or lower node incident to a saddle s in a contour tree indicates a strangulation in a MS-complex, where s is connected to u twice.

Proof. (by contradiction): Without loss of generality, assume u be a maximum. It is the only upper node incident to s in the contour tree. Assume to the contrary that there is another node v higher than s and incident to s in the MS-complex. Let the monotone ascending integral line connecting s and u be path (s, u), and the monotone ascending integral line connecting s and v be path (s, v). Let the union of all the contours crossing path (s, u) be C_s^u. Let the union of all the contours crossing path (s, v) be C_s^v. $C_s^u \neq C_s^v$ and $s \in C_u^s \cap C_s^v$, thus different contour components merge at s. Therefore, the upper node v is incident to s in the contour tree, contradicting the fact that u is the only upper node incident to s in the contour tree. □

5.2.1.2 De-strangulation

Though usual contour tree simplification methods do not address strangulation cases, we found handling it unavoidable in practice. Removing a strangulation pair (s, u) results in the dilemma of finding a local node to connect the nearby minimum v, as illustrated in Figure 5.2b and e. The isolated node v cannot be reconnected to the local saddle w since it is higher than w. One way to avoid this dilemma is to remove its closest neighbor, pair(s, v), first as demonstrated in Figure 5.2c and f. Removing pair(s, v) successfully solves the strangulation while the other parts of the topology are unaffected.

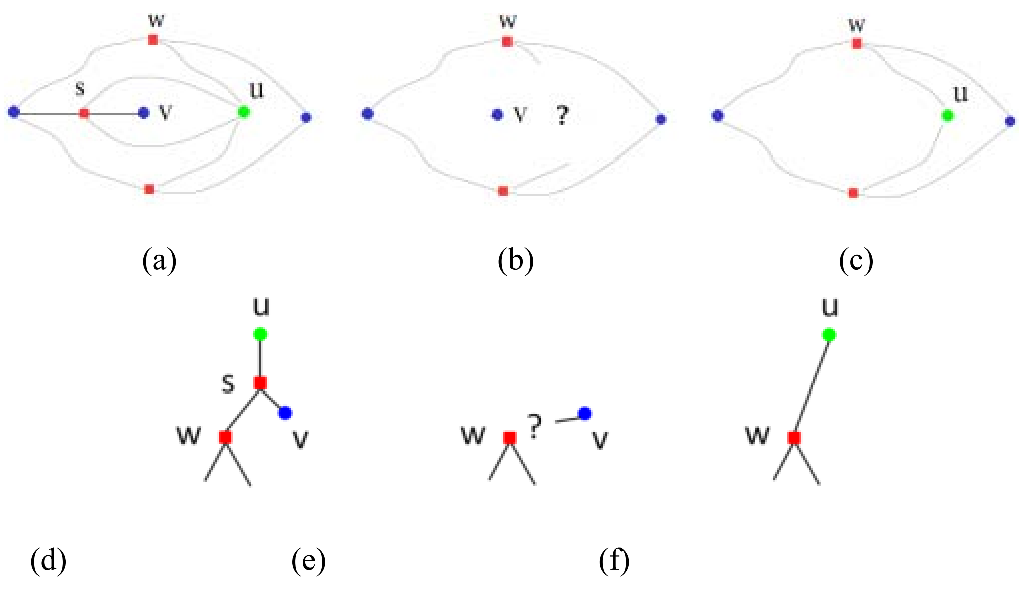


Figure 5.2 Strangulation and de-strangulation. (a) MS-complex strangulation case. (d) Contour tree strangulation case. (b) and (e) Removing strangulation pair s and u leaves no way for v to be reconnected to the nearby saddle. (c) and (f) Solving strangulation by removing the pair s and v.

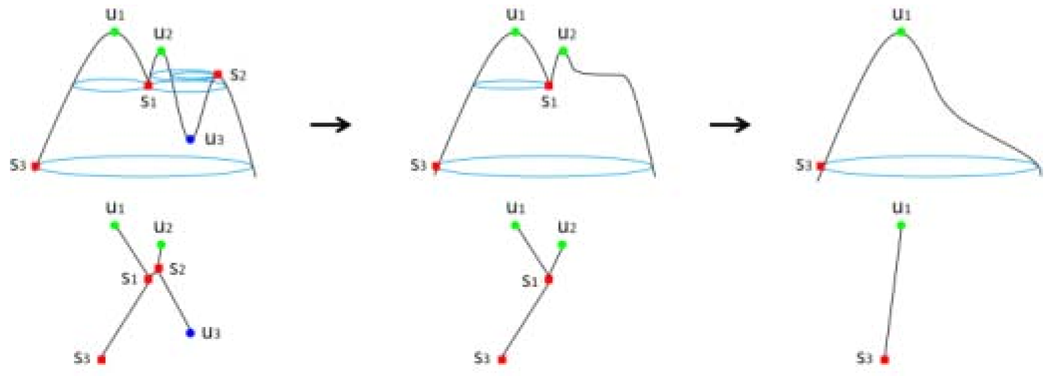


Figure 5.3 Pair cancelation order for a case involving strangulation. Left: a sub-tree with strangulation. Middle: removing pair (s_2, u_3) to solve the strangulation. Right: removing the once strangulated pair (s_2, u_2).

In a neighborhood containing strangulations, pairs cannot be removed in an order strictly according to the importance ranking. Figure 5.3 shows an exception, where the pair with the lowest persistence, pair (s_2, u_2), happens to be in a strangulation.

Persistence is used to measure importance in this example. The importance rankings among the pairs are pair (s_2, u_3)>pair (s_1, u_1)> pair (s_2, u_2). However, in order to solve strangulation, we remove pair (s_2, u_3) before we remove pair (s_2, u_2). Therefore, we first de-strangulate the pair (s_2, u_2) by removing its neighboring pair (s_2, u_3). In this example, the pair cancelation order is pair $(s_2, u_3) \rightarrow$ pair $(s_2, u_2) \rightarrow$ pair (s_1, u_1).

5.2.2 Top-down contour tree simplification

Section 5.2.2.1 gives a few definitions and lemmas. Section 5.2.2.2 gives a detailed implementation of the algorithm. Section 5.2.2.3 analyzes the algorithm.

5.2.2.1 Reversed simplification sequence and branches

Given a contour tree, with traditional bottom-up simplification [40], one simplifies it by cancelling saddle-extremum pairs with increasing persistence. After cancellations of a sequence S of pairs C_1, …, C_n, the simplified contour trees after each cancellation are CT_1, …, CT_n, respectively. The original contour tree is CT_0. The last cancelled pair C_n is a minimum and maximum pair. The key to constructing a top-down simplification is finding the reversed sequence S' for S.

We define a branch as a monotone path in a contour tree graph that starts from a given node and traverses a sequence of nodes with a non-decreasing (or non-increasing) value of f until it reaches the highest node (or lowest node) in the path. Pascucci et al. [39] and Weber et al. [34] showed that a contour tree can be decomposed into branches and rebuilt by assembling the branches afterward. Each branch rooted at an interior node, other than the two ends of the branch, is a *child branch.* The interior node is therefore called the *root* of the child branch. A branch which has child branches is called the *parent* branch *of its child branches.* A child branch is called *upward* if its root is lower than its opposite end or *downward* if its root is higher than its opposite end in value of f. The sub-tree which consists of all the edges and nodes along a branch and its descendants is called *the sub-tree of the branch*. Moreover, we define *the length of a branch* as the persistence of its two ends — the absolute difference of the two ends in iso-value. Figure 5.4 gives examples of these definitions. In this thesis, we make the term branch and pair interchangeable since for any pair $P_i(s,u)$, in the simplification sequence, there is a branch (s, u) in CT_{i-1}.

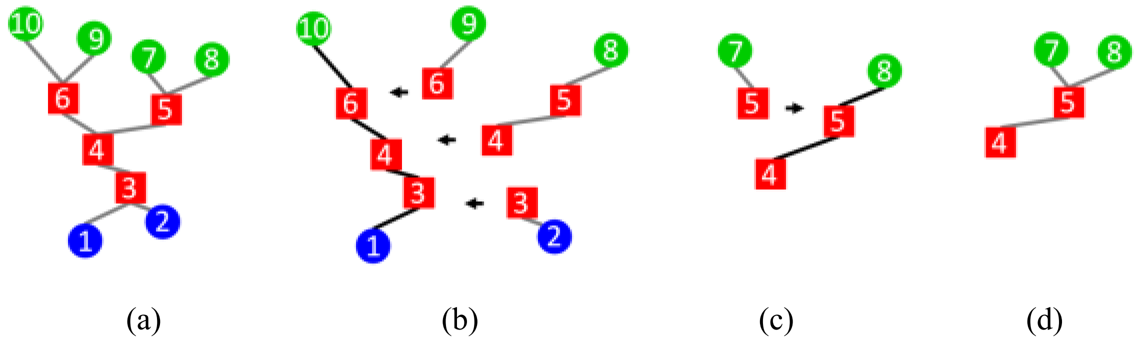


Figure 5.4 Definitions related to branches. (a) A contour tree. (b) A parent branch (1,10) with two upward child branches (6,9) and (4, 8) and one downward child branch (3,2). Branch (1,10) is longer than its child branches. (c) A parent branch (4,8) with one upward child branch (5, 7). Branch (4,8) is longer than branch (5, 7). (d) The sub-tree of branch (4,8). The nodes are numbered by their function values.

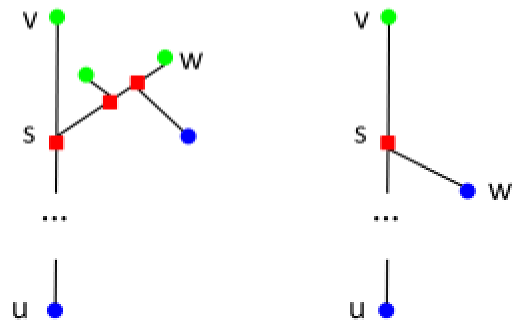
We introduce two lemmas for constructing the top-down simplification as follows.

Lemma 1 The last cancelled pair C_n in a bottom-up simplification is the pair of virtual minimum u and the farthest maximum v that is reached through a monotone ascending path of contour tree from u.

Proof. (1) The virtual minimum u has value $-\infty$; therefore, any pair connected to it has to be cancelled after other pairs.

(2) Proof that the maximum v has to be removed after cancellation of any pair C_i $(i = n-1,...,1)$: In any $CT_i(i = n-1,...,1)$, let the saddle connected to v be s_i. Let another branch connected to s be (s, w). There are two possible cases for w. First, w is a maximum (Figure 5. 5a). w cannot be higher than v since v is the highest node searchable from u . Even when the branch (s, w) contains strangulation child branches, pair (s, v) has to be removed after branch (s, w) since $w$the strangulation branch has to be canceled before branch (s, w). Second, w is a minimum (Figure 5. 5b). Pair (s, u) is therefore a strangulation pair and cannot be removed before branch (s, w).

Therefore, both u and v are left after all the other nodes are removed through simplification. The lemma follows. □



(a) (b)

Figure 5.5 Simplified contour tree CT_i. u is virtual minimum and v is highest maximum that is researched through a monotone path from u . (a) w is a maximum. (b) w is a minimum.

Lemma 2. Pair $C_i(i = n-1,...,1)$ is the pair with the highest persistence among all the child branches of current branches in a simplified contour tree, CT_{i+1}.

Proof. Let C_i be pair(s, w), without loss of generality, assume w is a maximum. A simplified contour tree, CT_{i+1}, consists of corresponding branches P_{i+1}, …, P_n of C_{i+1},

…, C_n. Let any other child branches in branches P_{i+1}, …, P_n be $B_j(j=1,...,m)$ $B_j(j=1,...,m)$. Let the sub-tree of branch C_i be ST and any branches of branch C_i be $D_k(k=1,...,q)$.

(1) B_j must be cancelled before C_i since it has lower persistence than C_i.

(2) Proof that D_k must be cancelled before C_i: Let the closest saddle to w in C_i be s_0. Assume that, at any simplification phase, another branch to s_0 is branch (s_0, v). If v is a maximum, it is lower than w since C_i is the one with higher persistence. So pair (s_0, v) has to be cancelled before C_i. If v is a minimum, due to the strangulation constraint, pair (s_0, v) has to be cancelled before C_i. s is cancelled after other saddles in branch (s, w) since pair (s, w) could be considered as a removable pair only after all the child branches of branch(s, w) are cancelled. Therefore, C_i remains after any D_k is cancelled. Therefore, the lemma follows. □

The following statement is concluded based on lemmas 1 and 2.

Remark 1. Starting from the last cancelled pair with the virtual minimum u and its highest reachable maximum v, a contour tree can be fully reconstructed from the child branches with highest persistence repeatedly found from the current contour tree.

This underlines the idea of our top-down contour tree simplification — growing a contour tree by repeatedly attaching the branches found in reversed simplification sequence.

5.2.2.2 Algorithm implementation

First, a function is needed to find pairs C_n, …, C_1 in the reversed simplification sequence S' and their persistence efficiently. This is achieved with a single pass traversal of the contour tree, beginning with the virtual minimum.

According to the contour tree properties which we discussed in Chapter 3, a node in a contour tree has at most three incident arcs. This means, a contour tree can be considered as a binary tree if we assume that an arbitrary leaf is its root. Figure 5.6 shows an example of viewing a contour tree as a binary tree. The virtual minimum is assigned the root of the tree.

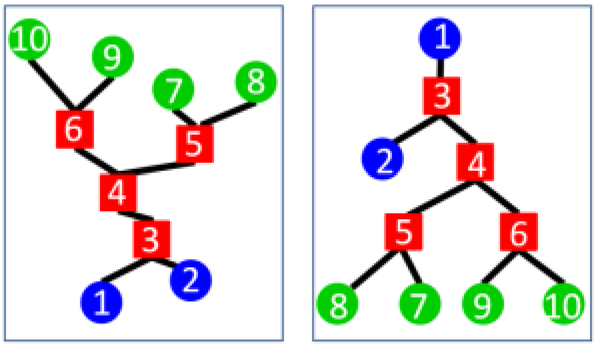


Figure 5.6 Transform a contour tree (left) into a binary tree (right).

Let the contour tree be *CT*. Let the virtual minimum be *CT* 's root. Given a node u, let the farthest reachable nodes through a monotone path from u in its two sub-trees be v_1 and v_2, respectively. Branch (u, v_1) and branch (u, v_2) are the longest branches with end point u in the two sub-trees, respectively. Always assume that branch (u, v_2) is longer than branch (u, v_1) (persistence (u, v_1)<persistence (u, v_2)). We perform a depth-first traversal of the tree and record the corresponding v_1 and v_2 for each node u and child branches of the branches (u, v_1) and (u, v_2).

```
Function Traversal (CT, $u$)
{
    If ($u$ is a leaf)
        $u.v_1 = u.v_2 = u$;
        branch ($u, u.v_1$).childbranches = branch ($u$, $u.v_2$).childbranches = $\varphi$;
    else if ($u$ is the virtual minimum)
        Traversal (CT, $u$.child);
        $u.v_2 = u$.child.$v_2$;
        branch($u, u$.child.$v_2$).childbranches=
        branch($u$.child, $u$.child.$v_2$).childbranches;
    else
        Let the left child of $u$ be $u_l$ and the right child of $u$ be $u_r$;
        Traversal (CT, $u_l$);
        Traversal (CT, $u_r$);
        If (persistence ($u$, $u_l.v_2$) < persistence ($u, u_r.v_2$))
            $u.v_1 = u_l.v_2$ and $u.v_2 = u_r.v_2$;
            branch($u, u.v_2$).childbranches =
            branch($u_r, u.v_2$).childbranches $\cup$ branch($u$, $u.v_1$);
        Else
            $u.v_1 = u_r.v_2$ and $u.v_2 = u_l.v_2$;
            branch($u, u.v_2$).childbranches =
            branch($u_l, u.v_2$).childbranches $\cup$ branch($u, u.v_1$);

}
```

Through this tree traversal, the farthest reachable node of the virtual minimum is found. According to Lemma 1, the pair of virtual minimum and its farthest reachable maximum form the last cancelation pair C_n. According to Lemma 2, pair C_{n-1} is found as the longest among the child branches of C_n. Recursively, C_i is found among the child branches of $C_n,...,C_{i+1}$. Along with the last pair C_n = pair (virtual minimum, virtual minimum . v_2), the child branches recorded cover all the branches in the cancellation sequence $C_n,...,C_1$. The order of the tree traversal and information stored for the contour tree in Figure 5.6 are shown as follows.

Table 5.1 The Order of the Tree Traversal and the Information Stored

N	N	N	Children of branch (u , v_2
2	2	2	--
8	8	8	--
7	7	7	--
9	9	9	--
10	1	1	--
5	7	8	(5,7)
6	9	1	(6,9)
4	8	1	(6,9) (4,8)
3	2	1	(6,9) (4,8) (3,2)
1	--	1	(6,9) (4,8) (3,2)

* u is the current visited node. * v_1 and v_2 are the farthest reachable nodes through a monotone path from u in its two sub-trees.

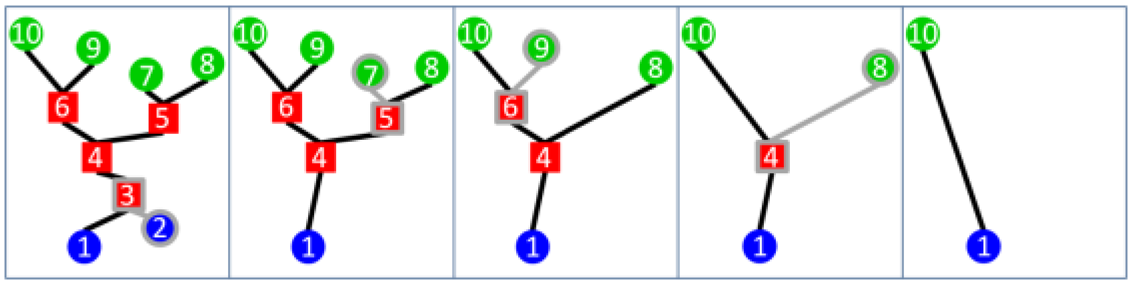
The following top-down simplification algorithm is constructed according to Remark 1. It gives the order of the recorded child branches in the cancellation sequence after the tree traversal. A priority queue storing the branches with decreasing lengths is employed.

```
*Algorithm* Top-down Simplification
Input: a contour tree $CT$ and an empty priority queue Q
Output: A sequence $S'$ of pairs $C_n$, …, $C_1$
    $u$ = virtual minimum;
    Traversal ($CT, u$);
    Push (Q, branch($u$ ,$u.v_2$));
    $n$ = the number of extrema in $CT$;
    *While* (Q is not empty)
    {
        branch($u$ ,$v$) = Pop(Q);
        *For* every branch $D_i$ in branch($u$ ,$v$).childbranches
        Push (Q, $D_i$) ;
        $C_n$=branch($u$ ,$v$);
        $n--$;
    }
```

The output of the above algorithm for the contour tree in figure 5.6 is a top-down simplification sequence S': C_5=pair (1,10), C_4=pair (4,8), C_3= pair(6,9), C_2= pair (5,7), and C_1= pair (3,2).

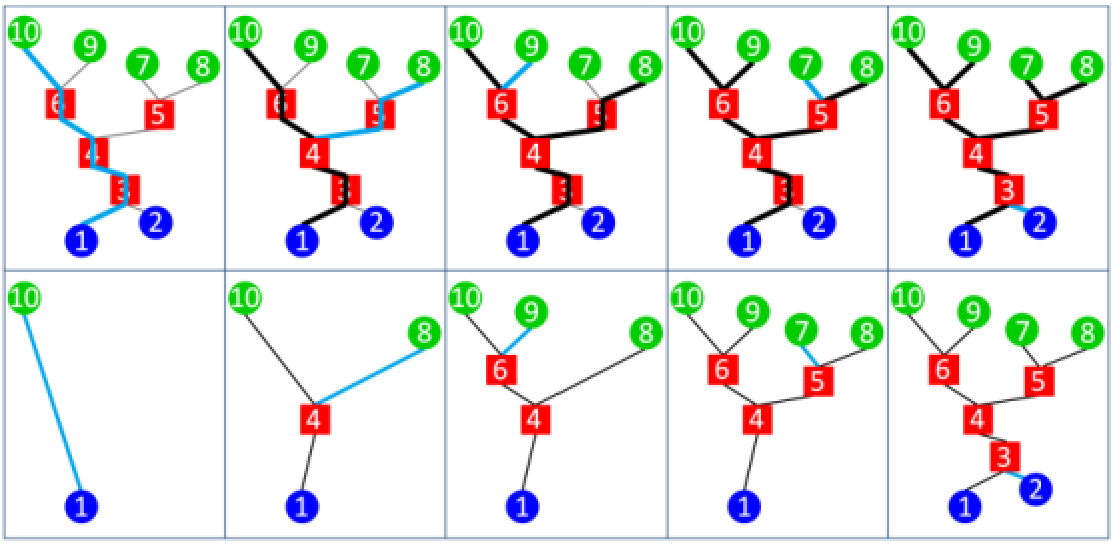
Figure 5.7 illustrates the relation between branch hierarchies and simplification sequences. Figure 5.7a shows a bottom-up simplification [40] by repeatedly pruning off the shortest branch on a current contour tree. As shown in Figure 5.7b, the top-down simplification exactly reverses the simplification sequence of the bottom-up simplification. Figure 5.7e shows the contour tree hierarchy built according to its simplification sequence. The black numbers 0, 1, ..., 4, indicate the order of assembling the branches. It corresponds to the reversed simplification sequence. As shown in this example, the reversed simplification sequence suggests a balanced hierarchy, the shorter branches are found at lower hierarchies and higher branches are located at higher hierarchies so that the more simplified contour tree always catches the more significant features.

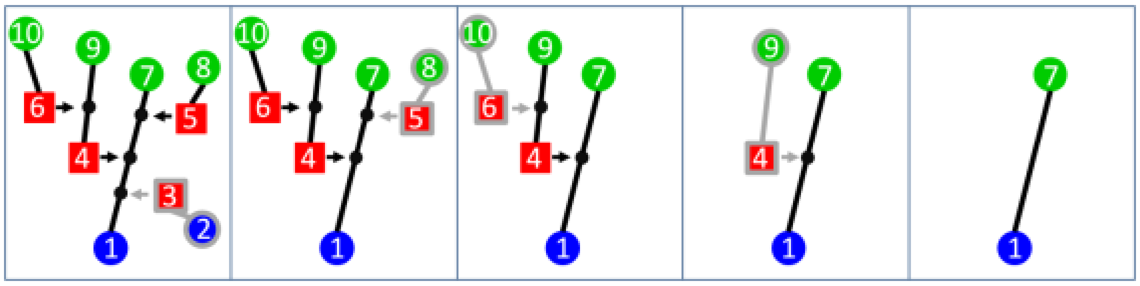
Figure 5.7c and d are reproduced from the example in [39]. It shows the hierarchy built according to the branch decomposition order. It provides a less balanced branch hierarchy: the longest branch (1,10) is missed, the shorter branch (1,7) is instead placed at the highest hierarchy. Therefore, we choose to conduct the top-down simplification instead of the branch decomposition [39] to build the contour tree hierarchy.



(a)

Figure 5.7 Top-down simplification compared with bottom-up simplification. (a) Bottom-up simplification by pruning off the shortest branch on the current contour trees. (b) Top-down simplification. (c) The simplification based on a branch decomposition result [39]. (d) The hierarchy built from the sequence produced by the simplification based on branch decomposition in (c). (e) The hierarchy built from the reversed simplification sequence in (b).


(b)


(c)

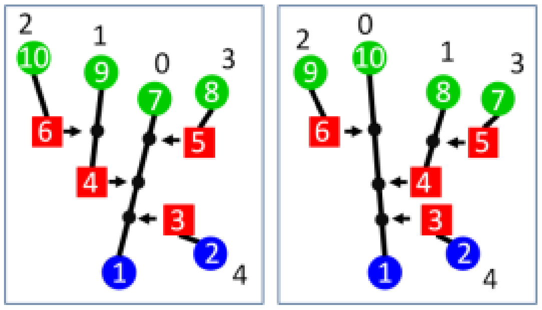

(d) (e)

Figure 5.7 (continued)

5.2.2.3 Algorithm analysis

The cost of the traversal of contour tree is $O(N)$, where N is the size of contour tree (the number of nodes on contour tree). The main cost of this algorithm is from running the priority queue Q, which depends on the dynamic size of Q. The number of branches n of a contour tree is about the number of its extremum, hence $n \approx N / 2$. The size of Q starts from 1 with pair C_n and grows as the child branches of $CT_i(i = n-1,...,a)$ increases, until, roughly, the child branches on $CT_i(i = a,...,1)$ reach a number less than the number of branches on CT_i. The size of Q has an upper bound n. In practice, its biggest size may be much lower than n.

The cost of running the priority queue maintained by persistence for a bottom-up simplification is analyzed for comparison purposes. The size of the priority queue starts from n and decreases as each pair is cancelled by one until the last pair is left. A brief analysis of the branch decomposition [39] method shows that their priority queue also starts with a size of about n and decreases by one with the "peeling off" each branch. The time complexity of maintaining such a priority queue is $O(n \log n)$.

Therefore, theoretically, the costs of top-down simplification, bottom-up simplification, and branch decomposition are almost the same. However, in practice, we often deal with large contour trees with thousands of branches. In this case, our method is much more efficient. For example, given a contour tree with $n = 1000$, a simplified contour tree with 100 nodes is desired. Our top-down simplification only has to work with a priority Qwhose maximum size is around100 while other methods have to work with a priority Q whose size starts from 1000 and drops by number one after each pair cancellation until the simplified tree with 100 nodes is obtained.

5.3 Contour tree layout and tree view graph design

We visualize uncertainty information through contour trees. The layout of a contour tree becomes an issue when the size and complexity of the contour tree increase

[43]. It is most intuitive to use the *y*-axis to encode the scalar value of a function in a planar display [6]. Heine et al. [43] suggested minimizing edge crossings as one of their planar contour tree drawing criteria. However, as pointed out in previous literatures [39], [5], self-intersections and visual saturation are unavoidable in such a display. Radial tree graph display [65] is a possible alternative, but it is hard to encode the height information meaningfully. A 3D tree layout is likely to be less interactive than a planar layout. To explore data with uncertainty efficiently, a preferred contour tree layout is the one that is two-dimensional, shows hierarchy and height information intuitively, suppresses branch self-intersections, allows for a fast navigation through different levels of simplification, and allows displaying uncertainty information. To meet these requirements, we design a rectangular contour tree layout which is integrated with the tree view graph interaction.

The property of critical points within a sub-tree on a branch is first investigated in section 5.3.1; then, the detailed implementation of the contour tree layout and tree view graph design is discussed in section 5.3.2 and 5.3.3.

5.3.1 Removable sub-trees

Lemma 3. The number of the critical points in each sub-trees of a saddle-extremum branch satisfies #{maxima}-#{1-saddle}+#{ minima}=0 for 2D case and #(maxima) - #(2-saddle) + #(1-saddle) - # (minima) =0 for 3D case.

Proof. (by induction). Let the root of the sub-tree be *s* and the edge incident to *s* be *e*. Let an arbitrary contour crossing the edge *e* be *c*. Without loss of generality, assume the sub-tree *ST* is higher than *s* (see an example in Figure 5.8).

(1) For a 2D case, the contour c surrounds all the critical point of the sub-tree except for *s*. When considering the sphere with the virtual minimum connected to the

boundary of the domain, the total number of critical points satisfies: #{maxima}-#{1-saddle}+#{minima + virtual minimum}=2. Therefore, the number of critical points within the contour is #{maxima}-#{1-saddle}+#{ minima}-# (virtual minimum) =1. Plus *s*, as a whole, the sub-tree satisfies the Eular Formula=0.

(2) For a 3D case, without loss of generality, assume *s* is 2-saddle, and the sub-tree *R*' is higher than *s*. When considering the 3D sphere with the virtual minimum added to the boundary of the domain, the Eular Formula = 0; therefore, the number of critical points within the 3D contour is #(maxima) - #(2-saddle) + #(1-saddle) - # (minima)= -1. Plus *s*, the Eular Formula = 0 for the critical points on the sub-tree. □

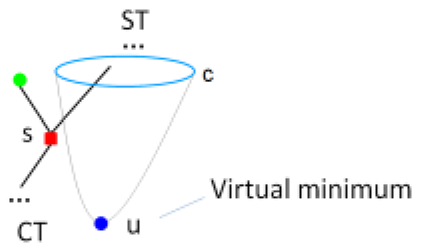


Figure 5.8 A sub-tree with a virtual minimum

We include the following Remark according to lemma 3.

Remark 2. Any sub-tree along a saddle-extremum branch could be removed from a contour tree with the resulted Euler equation unchanged for the rest of contour tree since the Eular Formula for the sub-tree has a result of 0.

5.3.2 2D contour tree layout

Our planar contour tree layout is proposed to show intuitive branch hierarchy and vertex height information with minimized branch crossings. Vertices are positioned on the *y*-axis according to their values to reflect the height information. It can be integrated with the tree view graph interaction to facilitate a fast navigation through different levels of simplification (discussed in section 5.3.3). Branches can be attached with uncertainty glyphs (discussed in section 5.4). We discuss our strategies to emphasize branch hierarchy and to control self-intersections as follows.

The key to prevent unnecessary self-intersections is to hierarchically locate sub-trees in nested regions in which each of the smaller sub-trees shares a smaller bounded region. An example is shown in Figure 5.9d. The rectangular layout is created by recursively assigning a vertical slot to a branch so that its child branches are contained entirely within the slot. To be more specific, a branch B is assigned a vertical slot R, and each child branch b_i of B is assigned a disjoint portion of R — a smaller vertical slot R_i within R. Each slot is positioned according to the heights of its two ends.

To emphasize the hierarchy, a branch is first rendered in the middle, and its child branches are spread out to the both left and right sides of it. The long branch which spans the last cancelation pair P_n is drawn as a vertical straight line segment in the middle of the display. All the other branches have L shapes that connect extremum to their paired saddles to prevent branches from crossing the slots of their siblings.

We further make a few special arrangements to avoid space conflicts between different branches. The slots assigned for upward child branches and downward child branches need to be carefully differentiated. Otherwise, the upward branches and downward branches may unnecessarily run into each other (Figure 5.9a). An easy way to avoid such intersections is to put all the upward branches on the one side and put all the downward branches on the other side of their parent branch. In addition, a higher upward branch is assigned a slot closer to its parent branch than the lower upward branches so that they do not intersect with each others. Symmetrically, a lower downward branch is assigned a slot closer to its parent branch than the higher downward branches. As shown in Figure 5.9b, this strategy effectively prevents all the intersection between the child branches. However, it is not a space-saving solution. An improved solution is given in Figure 5.9c which takes less horizontal space than the tree layout in Figure 5.9b. For a given parent branch, we separated it into three parts vertically: from high to low, upward branch zone where all the child branches are upward; mixed branch zone where child branches are either upward or downward; downward branch zone where all the child branches are downward. In the mixed branch zone, the upward branches take one side while the downward branch takes the other. In the upward branch or downward branch zone, the child branches stretch outward from the parent branch on both left and right sides without overlapping each other. In order to prevent a child branch from intersecting with its parent branch, one needs to decide which side of the parent branch for it to take. Without loss of generality, we assume that the parent branch is an upward branch located at the right side of its parent as shown in Figure 5.9. In this case, only downward child branches may intersect with the parent when it is on the left side of the parent and the lower end of it is lower than that of the parent, as the green branch in Figure 5.9. If this happens, we switch such child branch to the right side. If the child branch is located in mixed zoon, we place all the other downward child branches on the right side of the parent branch while the upward child branches on the left. As shown in Figure 5.9c, the red and green branches are placed on the right side so that they do not intersect with the parent branch. This strategy prevents all the intersections between the child branches and its parent.

This new contour tree layout shows the height and hierarchy information effectively and prevents all the self-intersections among siblings and between parent and child branches. Some self-intersections are unavoidable due to the strangulation cases where a downward branch appears as a child of an upward branch, or vice versa. As shown in Figure 5.9d, a strangulation branch (indicated in the red box) is long and reaches the branches outside its parent branch (indicated in the green box). This happens infrequently. As shown in Figure 5.9e, there is always at least one self-intersection in the contour tree. However, in Figure 5.9f, the rectangular display of the same tree reduces the number of crossings from three to one since our layout design rules out the case where a child branch intersects with its parent or siblings.

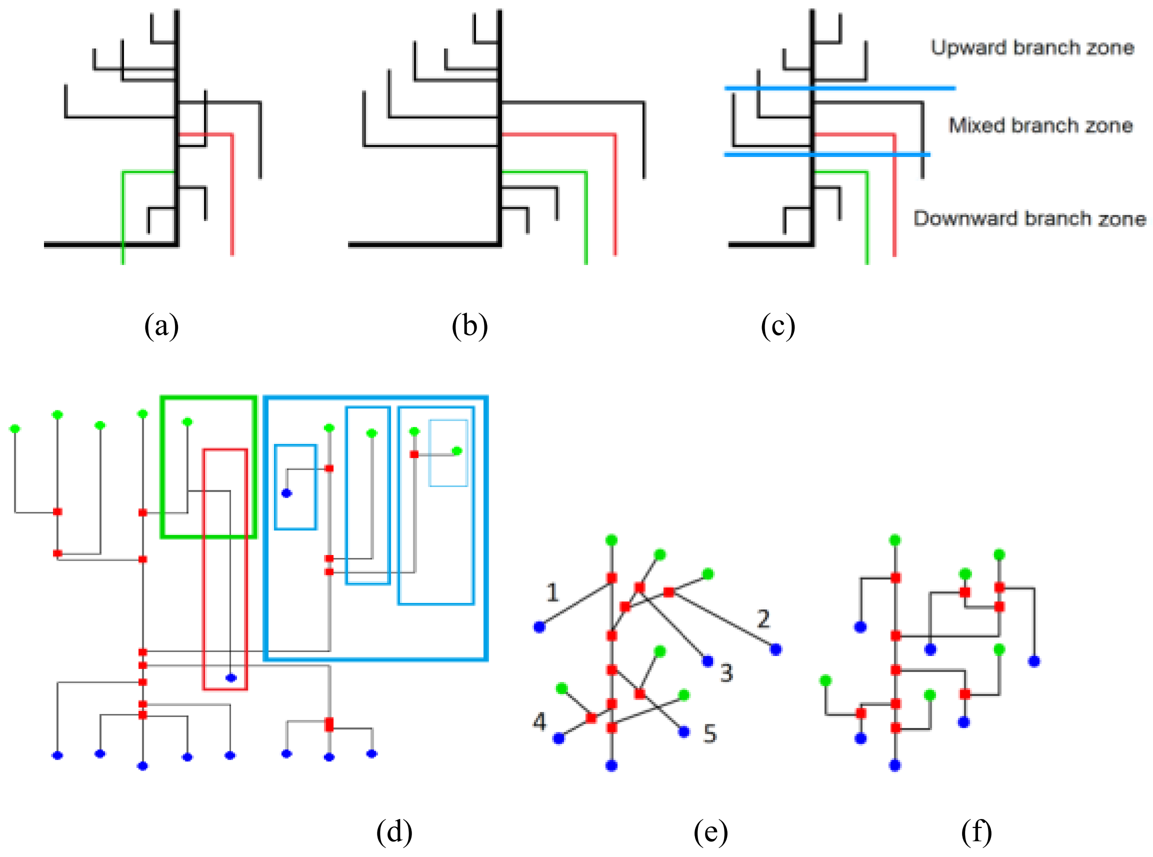


Figure 5.9 Strategies to reduce self-intersections. (a) An unguided placement of child branches with multiple branch crossings. (b) Placing upward child branches and downward child branches on the different sides. (c) Placing upward child branches and downward child branches on the different sides only in the mixed branch zone. (d) The rectangular 2D layout of a contour tree with each branch assigned in nested vertical slots. Boxes indicate vertical slots assigned to some branches. An unavoidable self-intersection is shown in the red box. (e) A 2D layout of a contour tree with 5 strangulations. (f) The rectangular display of the contour tree in e.

We notice that our layout appears to be similar to the orthogonal layout recently developed by Heine et al. [43] which shows L shape edges with values mapped to heights and branches grouped hierarchically. They adopted a delicate but time consuming four-phrase procedure to obtain a layout: grouping branches, minimizing crossings between branch groups, ordering, and positioning branches. The second phase which uses a random walk combined with simulated annealing to minimize the crossing costs is nondeterministic and takes most of the runtime. As stated in [43], computing the layout for a contour tree with 2,000 critical points could take up to 10 minutes with a certain number of crossings. Our method utilizes the contour tree hierarchy to avoid the time consuming minimization process. It only takes linear time to travel through each branch of the contour tree hierarchically but prevents all the branch intersections except for the rare cases where strangulation branches are too long to be contained in the vertical slots of their parents. Our method usually takes less than half a second for a contour tree with around 2,000 critical points. For example, for the weather simulation dataset with 1724 critical points in Figure 5.18, it takes 312 microseconds and produces no branch crossing; for the brain dataset with 2142 critical points in Figure 5.19, 343 microseconds, no branch crossing.

5.3.3 Tree view graph interaction design

A typical tree view graph displays a hierarchy of items by indenting child items beneath their parents [66]. In tree view representations, the interactions are directly embedded: the user can collapse (or expand) a particular sub-tree to hide (or show) the items in that sub-tree.

Our planar contour tree layouts based on the nested sub-trees can be directly integrated with a tree view interaction. As stated in Remark 2, any sub-tree of a saddle-extremum branch could be collapsed with the resulted Euler equation unchanged and the topological integrity of the contour tree unaffected.

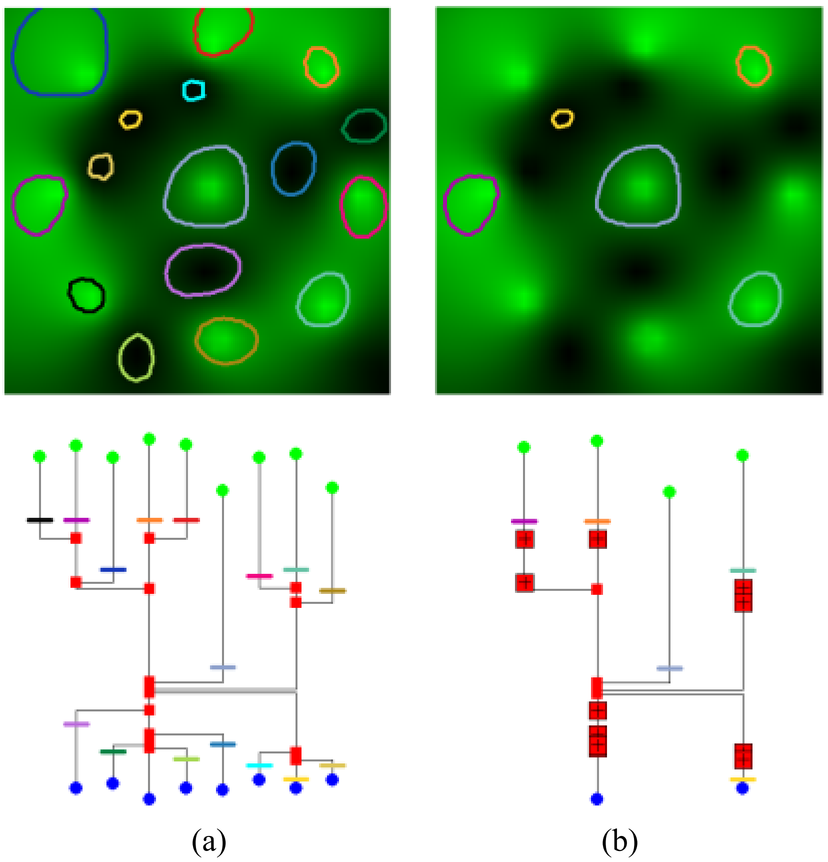


Figure 5.10 Tree view interaction.(a) An original contour tree and its corresponding 2D scalar field with contours selected for every branch of it. (b) An interactively simplified contour tree after a user has clicked on several nodes and contours have been selected for every branch of it.

An example of data exploration through tree view intersection is given in Figure 5.10. The data is a vorticity magnitude field of a simulated flow with vortices. A click on an inner node of the original contour tree (Figure 5.10a bottom) results in hiding or showing the sub-tree rooted at the node. The persistence, indicated as the vertical length of a branch, serves as an importance indicator for users to select contours. An interactively simplified contour tree is shown in Figure 5.10b. The roots of the collapsed sub-trees are marked with “plus mark” icons. To show how the simplification of a contour tree links to the data visualization, for each branch $P(s,u)$ of a contour tree, a single contour with iso-value $f_{iso} = tf(s) + (1-t)f(u), (0 \le t \le 1)$ is extracted. The horizontal line segments shown on the contour trees indicate the one-to-one mappings between the nodes in a contour tree and the contours in the scalar field. The same color is used for a contour and its corresponding location in the tree. The selected contours are the representative contours that give an overview of the whole scalar field. The more a contour tree is simplified, the higher level overview is obtained.

5.4 Contour tree-based uncertainty visualization

This section discusses three uncertainty metrics and their visualization representations. The uncertainty of the data is explored through data-level uncertainty, contour-level uncertainty, and topology-level uncertainty. The uncertainty information is attached to the new planar contour tree display to give a high-level overview of uncertainties and to allow a quick and accurate selection of contours with different levels of uncertainty.

We call the scalar field obtained by averaging the values from all the ensemble members at each grid point the ensemble mean and the contour tree of the ensemble mean the mean contour tree. In this thesis, we show the data level uncertainty by displaying the difference between each ensemble member and the ensemble mean at each grid point or along a contour. For contour-level uncertainty, given a contour in the ensemble mean, we compute the mean and variance of the differences between this contour in the mean field and its corresponding contours in all the ensemble members. For topology-level uncertainty, we map the contour trees of all the ensemble members to the mean contour tree and use their discrepancy to indicate uncertainty.

Our method is not limited by the uncertainty definitions we use in this thesis. Different uncertainty measurements can be shown through the contour tree via the use of different uncertainty metrics and corresponding glyph designs.

5.4.1 Data-level uncertainty

The data-level uncertainty measures how uncertain the numerical values of the data are. Uncertainty measures, such as standard deviation, inter-quartile range, and the confidence intervals, fall into this category. Sanyal et al. [13] visualized data-level uncertainties via circular or ribbon-like glyphs over a color-mapped image of the data. Due to the overlap between the data and uncertainty glyphs, the number of the glyphs has to be limited and information loss for both the data and uncertainty is unavoidable. Other techniques which overlay or embed uncertainty representation in the underlining data visualization face similar issues. We propose an alternative visualization that attaches uncertainty glyphs to a contour tree rather than integrating them with data visualization directly.

We adapted Sanyal et al.'s graduated glyphs to visualize data-level uncertainty. They proposed circular graduated glyph and ribbon-like graduated glyph. A circular glyph encodes the deviation of all ensemble members from the ensemble mean at a grid point. Let the differences for ensemble members be $d_1, d_2, \dots, d_k$. These difference values are then scaled according to the glyph-spacing and sorted in decreasing order to generate a new array $D_1, D_2, \dots, D_k$. Starting with D_k, we render successively smaller glyphs with decreasing sizes, $D_1, D_2, \dots, D_k$ and increasing saturation levels, $1/k, 2/k, \dots, 1$. A graduated ribbon is constructed by interpolating between circular glyphs placed along an iso-line in the data image.

We use the circular graduated glyph to show uncertainty at each grid point. Figure 5.11 shows a 2D 9×9 uncertain scalar dataset which is a down-sampled sub-region of the data in Figure 5.10. In Figure 5.11a, the average field of eight members is color-mapped and overlaid with uncertainty glyphs at each grid point. Likewise, the glyph of a grid point is attached to its corresponding vertex in the fully augmented contour tree as shown in Figure 5.11c.

Optionally, graduated ribbons are attached to branches to show the average data-level uncertainty along individual contours. We resample the varying data-level uncertainty along each branch using a fixed step size and use linear interpolation to produce a ribbon-like uncertainty glyph along the branch. For a given location v of a branch, let its corresponding contour be C, the evenly spaced samples along C be p_j ($j = 0,...,l$). Let the difference between ensemble member i and the ensemble mean at p_j be d_{ij}. The average difference from ensemble member i to the ensemble mean along C be $w_i = \sum_{j=1}^{l} d_{ij} / l$. To order to produce a ribbon along the branch, $w_1, w_2, ..., w_3$ are scaled according to the branch spacing and sorted in increasing order to generate a new array D_1, D_2, … , D_k which give the radii of ribbons $1, 2, ..., k$ at location v. A graduated ribbon is produced by overlapping ribbons $1, 2, ..., k$ with increasing saturation levels, $1/k, 2/k, ..., 1$. Figure 5.11d illustrates a segment of a graduated ribbon along a branch.

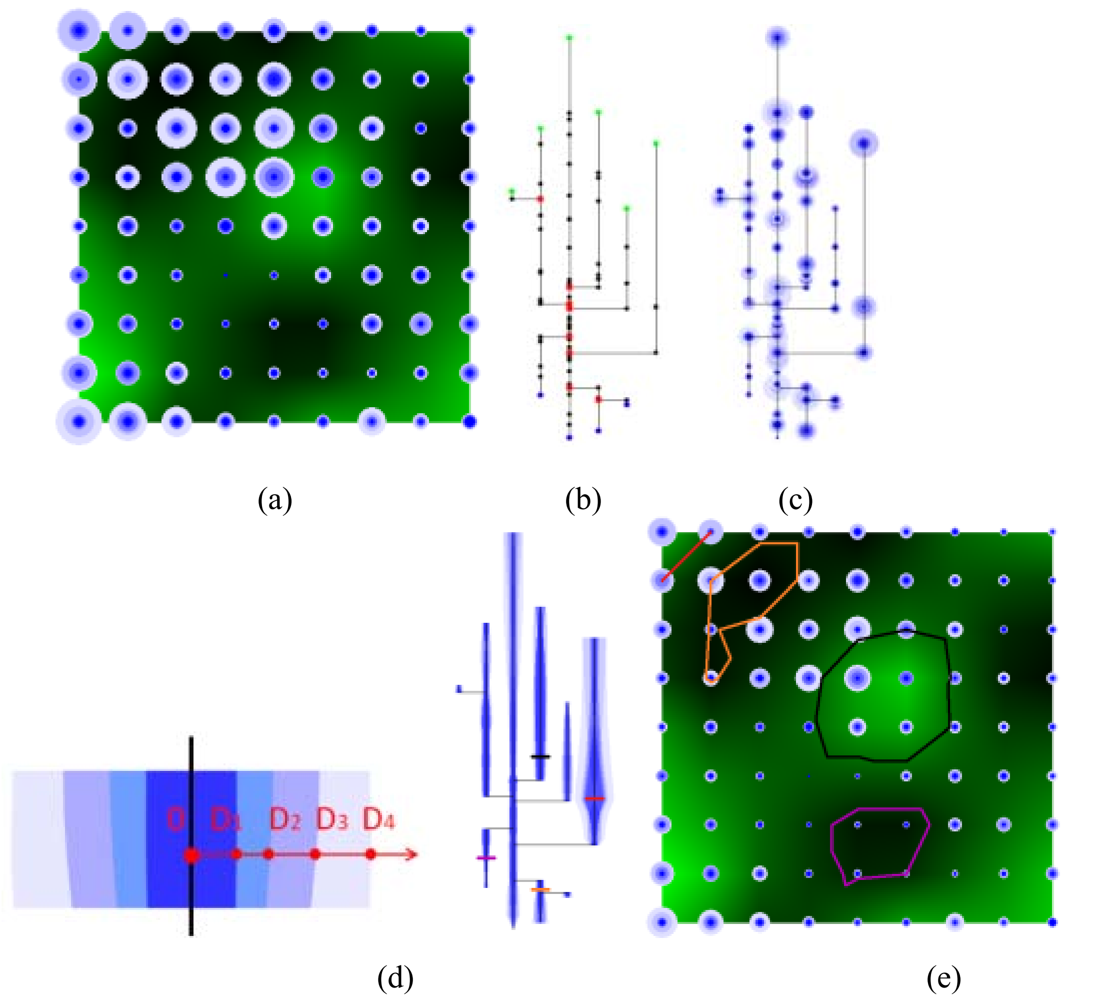


Figure 5.11 Data-level uncertainty representation based on contour tree. (a) Graduated circular uncertainty glyphs on an uncertain scalar field. (b) A fully augmented contour tree. The red, green, and blue dots indicate critical points while the black dots indicate regular vertices. (c) A contour tree with circular graduated uncertainty glyphs. (d) A segment of a graduated ribbon on a branch (black) constructed by overlaying thinner ribbons successively. (e) Contour tree attached with average data-level uncertainty along each contour and four sets of corresponding contours are shown after clicking on four locations (indicated by colored line segments) in the contour tree.

As shown in Figure 5.11 c and e, while a graduated circle illustrates the data-level uncertainty at a grid point, the graduated ribbon provides continuous uncertainty representation along individual contours with less clutter. Therefore, we use ribbon-like glyphs for representing data-level uncertainty along contours in our implementation. It provides a straightforward way to visualize data-level uncertainty along contours in the contour tree. A ribbon that has a dense and saturated core with a faint periphery indicates that ensemble members mostly agree and have a few outliers while a mostly dark glyph indicates large differences among individual members.

5.4.2 Contour-level uncertainty

Contour-level uncertainty measures the uncertainty in the position of a contour. In a spaghetti plot [13], the most unstable contours and the places where the contours are extremely diverse among individual ensemble members are interesting to users. However, the users' estimates tend to be inaccurate due to the randomness of the contour size, shape and length. Additionally, it is hard to look into such uncertainty in a large data set. Therefore, precise and automatic contour–level uncertainty measurement is needed to assist the exploration of the large uncertain data. A contour-level uncertainty which quantifies how uncertain a contour is within multiple ensemble members is developed to address this need.

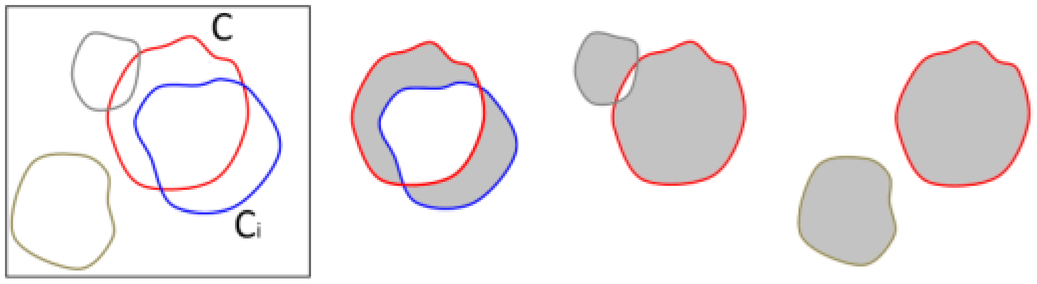
To measure the uncertainty of a contour, we first identify corresponding contours in different ensemble members. Then, we calculate the differences between them and use the mean and variance of the differences to represent the contour-level uncertainty. Our method is different from previous methods of studying uncertain contours which visualize probability field through color-mapping or volume rendering around a contour [19], [21], [22] since it is visualized through contour trees and measures the uncertainty as contour deviations in different ensemble members directly.

5.4.2.1 Contour correspondence and difference

For a contour in one ensemble member, there may be more than one contour in another ensemble member with the same iso-value with it, or it may be missing in some ensemble members. Spatial overlap [41], [42] is frequently used as a similarity measure to match features in different data sets. For instance, the correspondence between two contours can be measured by the degree of contour overlap as discussed by Sohn and Bajajin [42] when they computed correspondence information of contours in time-varying scalar fields. Given a contour C in the ensemble mean, we search in ensemble member i for the contour C_i $(i = 1, ..., k)$ who shares the same iso-value with C and has the best correspondence with C . The best matched contours, if found, are considered the same contours which appear in different ensemble members.

Figure 5.12a shows three contours (in blue, gray, and brown) in ensemble i with different correspondence degrees with contour C (in red). With the largest overlap degree with C , the blue contour is identified as the corresponding contour of C in ensemble member i. It is possible that no contour in C_i fulfils the correspondence criteria discussed in [42]. For example, in ensemble member i, no contour with the same iso-value of C exists or all the contours of the iso-value are apart from C . In these cases, we consider no matched contour found for C in ensemble member i.

Accordingly, the non-overlapped area $A(C, C_i)$ between the two corresponding contours C and C_i decides the difference between the two contours. Figure 5.12 b, c and d illustrate the non-overlapping area in different cases. The non-overlapping area is computed by using B-spline function [6], [42] for each simplice of a scalar field defined on a simplicial mesh. To reduce bias towards long or short contours, larger or smaller iso-surfaces, we normalize the non-overlapping area with the contour length (or iso-surface area in 3D case) of C : $difference(C, C_i) = A(C, C_i) / size(C)$, where $size(C)$ is the contour length (or iso-surface area in 3D case) of C.



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

Figure 5.12 Contour difference measured by non-overlapped areas. (a) Three contours (in blue, gray, and brown) in ensemble member i share a domain with contour C (in red) in the ensemble mean. (b), (c), and (d) Non-overlapped areas (filled with gray) of different contours.

5.4.2.2 Contour-level uncertainty metrics and visualization

To help a user select contours according to the quantified contour-level uncertainty information, we calculate the mean and variance of the differences to the mean. Given a contour C in the ensemble mean, let its corresponding contours of k individual ensemble members be C_i ($i = 1, ..., k$). C_i is the contour with the same iso-value that is matched with C . The average difference among the corresponding contours is: $mean = \sum difference(C, C_i) / k$. The variance of difference among contours is: $va = \frac{\sum (difference(C, C_i) - mean)^2}{k-1}$. Each ensemble member is supposed to have its contribution to the above equations. If an ensemble member does not have a matched contour for C , its contribution is set to a large value, the maximum non-overlapping area found between C and all the matched contours C_i.

The contour-level uncertainty along a contour can be shown at the corresponding location on the contour tree. Figure 5.13 illustrates the glyph design to encode the contour-level uncertainty. Before visualization, both uncertainty statistics are normalized to a range between 0 and a unit width. Since ribbon-like glyph is preferred over circular glyph to prevent visual cluttering as discussed in section 5.4.1, we resample the varying contour-level uncertainty along each branch and use linear interpolation to produce a ribbon-like uncertainty representation along each branch. For each branch, two ribbons are attached. The blue one is for the mean difference, while the green one is for the variance. The varying width of each ribbon indicates the varying magnitude of each uncertainty measurement for the contours along the branch.

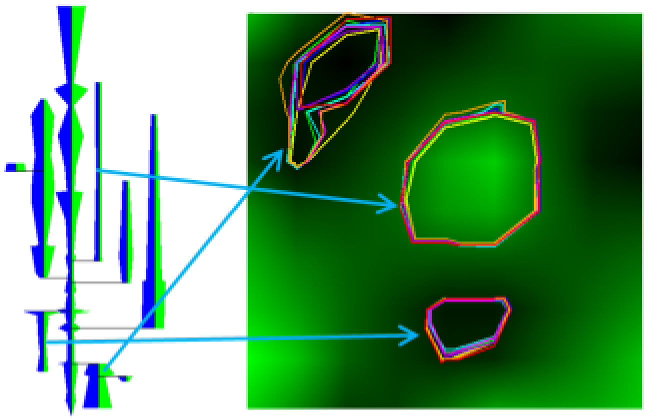


Figure 5.13 Contour-level uncertainty visualization based on contour tree. The uncertainty ribbons attached to the contour tree (left) indicate the uncertainty of corresponding contours in the data (right). Three sets of corresponding contours are shown after clicking on three locations (indicated by arrows) in the contour tree.

One can click on a location in the contour tree to display a bunch of corresponding contours (spaghetti plots [13]) with certain uncertainty levels in a data image. This saves users time and effort searching the whole image for a contour with a specified level of uncertainty. An example is shown in Figure 5.13. Uncertainty ribbons are attached to a contour tree. Three sets of contours with different levels of uncertainty are shown after clicking on three locations in the tree.

5.4.3 Topology-level uncertainty

The uncertainty within the data impact not only the values or the contour positions locally, but also the global pattern of the data which is described by the topology of the data. Visualizing the uncertainty concerning the topology among the ensemble members provides new perspective on the global impact of uncertainty. We propose a topology-level uncertainty visualization based on contour trees.

In this thesis, the topology-level uncertainty is defined as the variation in the height and number of branches in the contour tree of an uncertain scalar field. The idea is to map the branches between the contour tree of different ensemble members and the contour tree of the ensemble mean and to overlay the mapped branches. A set of matched branches are assigned with a same x-axis value but keep their original y-axis values so that an overlap of the branches on x-axis indicates their correspondence while the disagreements between the branches on y-axis indicate their discrepancy in iso-value. A branch of mean contour tree may not find a matched branch in the some ensemble members. The number of matched branches is encoded with the width of the branch. A thicker branch is more certain than a thinner one.

5.4.3.1 Contour region of a branch

We define a *contour region of a branch* as the region covered by all the contours within the sub-tree of the branch. Figure 5.14 shows two ensemble members of an uncertain scalar field. A comparison of their contour trees reveals that their iso-value ranges in different contour regions are different since their branches have different lengths and heights on y-axis. The contour regions for the purple branches are the regions inside the purple contours. In Figure 5.14 a and b, the iso-value ranges of the contour regions are different. The latter is higher and larger. This is clearly reflected in the different y-axis locations and lengths of their corresponding branches. An extra branch is found in the lower right corner of Figure 5.14 b. It indicates an uncertain contour region (inside the orange contour) which does not exist in the ensemble member in Figure 5.14 a.

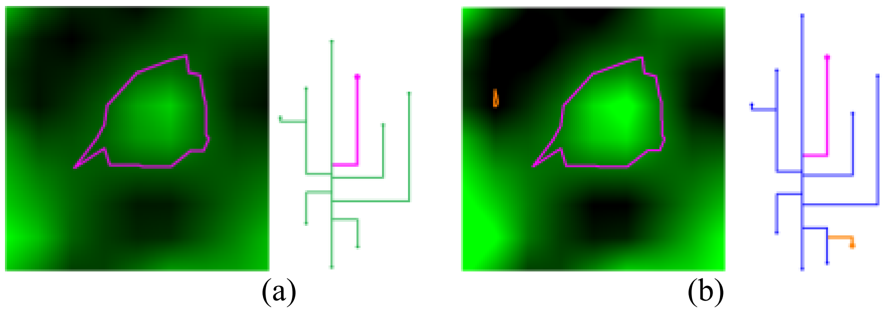


Figure 5.14 A side-by-side display of two ensemble members. (a) and (b) show two scalar fields with their contour trees. The lengths and vertical positions of the thick branches represent the iso-value range of the region inside the purple contour. The short branch in the lower right corner of (b) is an uncertain branch whose corresponding contours are missing in (a).

The contour region of a child branch is included in the contour region of its parent branch. Figure 5.15a shows tree branches and their corresponding contour regions. The nodes and contours are numbered by their iso-values. The contour region of a branch is bounded by the contour of its root and with all the nodes of its sub-tree inside. For instance, the contour region of branch (5, 7) is the area inside the contour 5. The contour region of branch (4, 8) is the region inside the contour 4 (including the region of branch (5, 7)).

5.4.3.2 Branch correspondence

Spatial overlap is the most common similarity measure between features [41], [42]. We measure the correspondence degree between two branches as the spatial overlap between their contour regions.

Given a branch B in the mean contour tree, we search in ensemble member i for its best matched branch. The best matched branches, if found, are considered the same branches which appear in different ensemble members. We do not need to search all the branches of the contour tree in ensemble member i for the branch with largest overlap with B. The contour tree hierarchy and branch orientation help us limit the number of branches to compare. (1) The matched branch must be found among the child branches of the matched branch of its parent due to the nesting relationship between child and parent branches. (2) A downward branch does not match an upward branch, or vice versa. An upward branch is related to a contour region enclosing its upper end (a maximum) while the downward branch indicates a region enclosing its lower end (a minimum). They are two topological features which need to be differentiated.

Figure 5.15 illustrates the branch correspondences detected according to the spatial overlaps of contour regions. Figure 5.15a and b show contours and contour trees of two scalar fields sharing a same spatial domain. In Figure 5.15a, the sub-tree of the middle branch (1, 10) is the whole contour tree. The sub-tree of branch (0.5, 10.5) is the whole contour tree in Figure 5.15b. Both branches share the whole data domain and hence are the best matched branches of each other. For the rest branches, the correspondence indicated by overlaps between their contour regions: branch (2, 3) → branch (1.8, 3.3), branch (6, 9) → branch (6.5, 9.6), branch (4, 8) → branch(4.2, 8.3), and branch(5, 7) → branch(5.4, 7.1). For example, branch (4, 8) matches branch (4.2, 8.3) since they have the largest overlaps. Branch (5.4, 7.1) matches branch (5, 7) even when branch (4, 8) has larger overlap with it since branch (4, 8) is mapped to its parent branch (4.2, 8.3). Figure 5.15 c shows the matched branch in the same x-axis location.

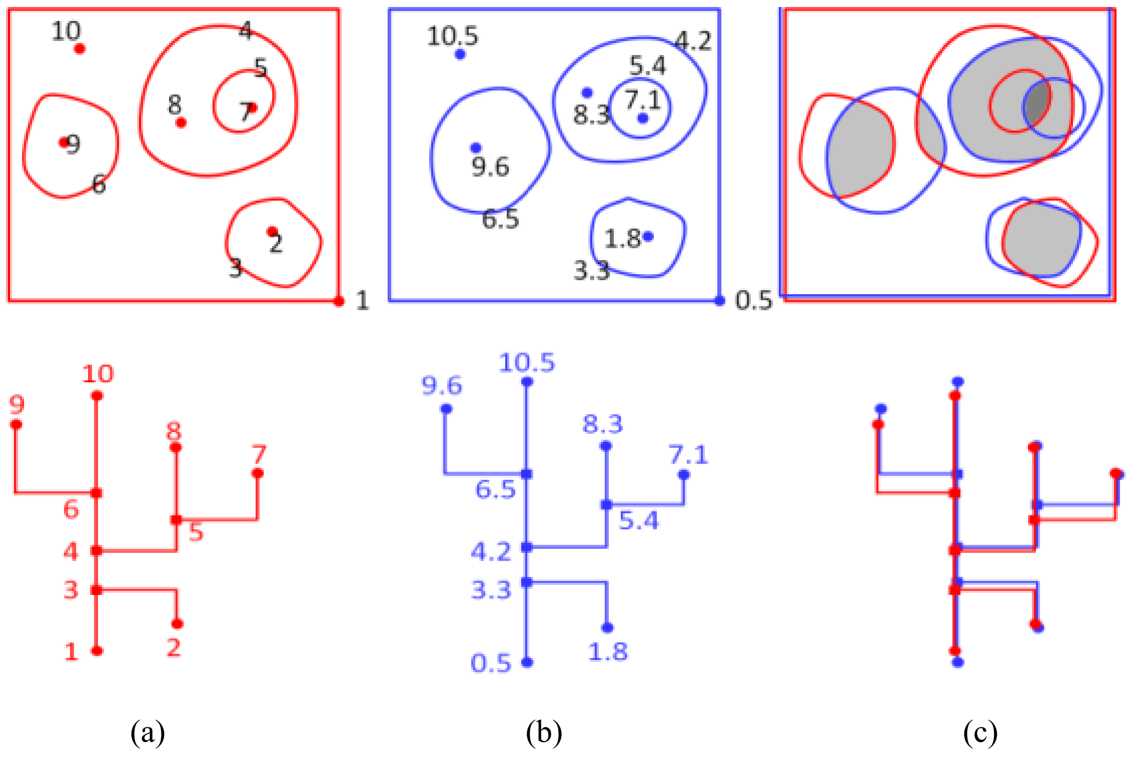


Figure 5.15 Branch correspondence indicated by contour region overlaps. The nodes and contours are numbered by their iso-values. (a) and (c) show contour trees of two data fields and the corresponding contour regions of the tree branches.

5.4.3.3 Topology-level uncertainty visualization via contour tree mapping

We map the contour tree of each ensemble member to the mean contour tree hierarchically. Let CT be the contour tree of the ensemble mean, CT_i be a contour tree of ensemble member i. Let the branch number in CT be n and the branch number in CT_i be m. Starting from the last branch in the reversed bottom-up simplification sequence $\{C_{i,j}, j = m,...,1\}$ of CT_i, we search the best matched branch $C'_{i,j}$ in CT for branch $C_{i,j}$.

First, the last branch $C_{i,m}$ in the reversed simplification sequence of CT_i is matched with the last branch of the mean contour tree, C_n, since they share the same data domain. For each branch $C_{i,j}$ in CT_i, we first check whether its parent branch has a matched branch CT. If its parent has no matched branch, neither does it. Mark $C_{i,j}$ as an unmatched branch. Otherwise, let its parent branch be $PC_{i,j}$, and the best matched branch of $PC_{i,j}$ in CT be $PC'_{i,j}$. Assume without loss of generality that $C_{i,j}$ is an upward branch. Check all the upward child branches of $PC'_{i,j}$ which are unmatched currently. The best matched branch $C'_{i,j}$ for $C_{i,j}$ is the one with the largest overlapping region. Mark $C_{i,j}$ and $C'_{i,j}$ as matched branches. If all child branches of $PC'_{i,j}$ are matched already, mark $C_{i,j}$ as an unmatched branch.

The algorithm outline of the matching process for CT and CT_i is given below.

```
Mark $C_{i,m}$ and $C_n$ matched branches;
For $j$ in $(m-1,...,1)$
    Find $C_{i,j}$'s parent branch $PC_{i,j}$
    If ( $PC_{i,j}$ has its best matched branch $PC'_{i,j}$ in $CT$ )
        (Assume $C_{i,j}$ is an upward branch)
      If ( no unmatched upward child branch overlapping $C_{i,j}$
           is found on $PC'_{i,j}$ )
            Mark $C_{i,j}$ as an unmatched branch
      else Mark the child branch of $PC'_{i,j}$ with the largest
            overlap with $C_{i,j}$ as its matched branch $C'_{i,j}$.
```

In the resulting visualization, the contour trees of different ensemble members are rendered beneath the mean contour tree in different colors. The x-axis locations of the matched branches are aligned. The number of unmatched branches for a branch in the mean contour tree is encoded as the width of the branch.

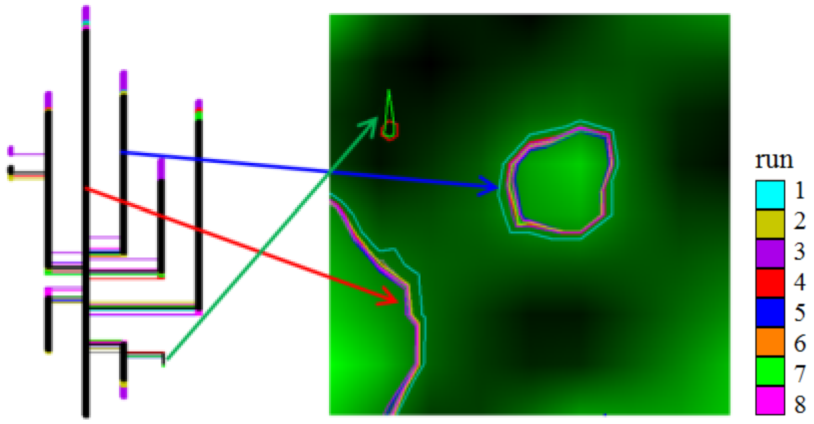


Figure 5.16 Topology-level uncertainty visualization. Three sets of contours are shown after clicking on three locations (indicated by arrows) in the contour tree.

As shown in Figure 5.16, the mapped contour trees indicate how uncertain the iso-value ranges of individual branches are and how uncertain the number of branches is. The places where branches of the contour trees disagree with each other indicate the uncertain iso-value ranges of different contour regions segmented by branches. The overall blurring or clearness of the display indicates the overall high or low uncertainty. The thickness of a branch indicates the number of matched branches found in the ensemble members. For example, a thin branch (indicated by the green arrow) is found in the lower right corner of the tree. It represents an uncertain contour region (or minimum) which only appear in a few ensemble members. As shown in Figure 5.16, only two contours (in blue and green) from ensemble members 4 and 7 are shown after clicking on the branch. On the contrary, the two sets of contours selected from the middle locations of two thick branches (indicated by red and blue arrows) include corresponding contours from all the ensemble members. Accordingly, the topology-level uncertainty provides users a quick and high level overview of how much uncertainty there is in the topology of contours.

5.5 User interface design

We use an open source version of Qt 4.5 to build our interface and VTK 5.8 to implement iso-surfacing and volume rendering of the data.

5.5.1 Interface components and layouts

Figure 5.17 shows the interface designed to enable efficient browsing, manipulation, and quantitative analysis of uncertain scalar data fields. It consists of five display areas. The top-left area shows a 2D or 3D visualization of a data set, including iso-lines and color-mapped image for 2D data, or iso-surfaces and volume rendering for 3D data. It provides interactions such as rotation and zooming. The top-right area shows a 2D display of the contour tree of the average data field. It allows interactive contour tree simplification, contour selection. The three bottom areas show the three levels of uncertainty of the data. They allow similar interactions as the top-right contour tree display area. All five areas can be hidden, resized, or switched with other areas.

Once a data set is imported with the pre-computed uncertainty information, the tool shows the color-mapped data (volume rendered for 3D data) in the data display area. The contours (iso-lines or iso-surfaces) are also rendered in the data display area and updated accordingly as a user clicks on one contour tree or changes the iso-value by sliding on the vertical bar in the top-right contour tree display area.

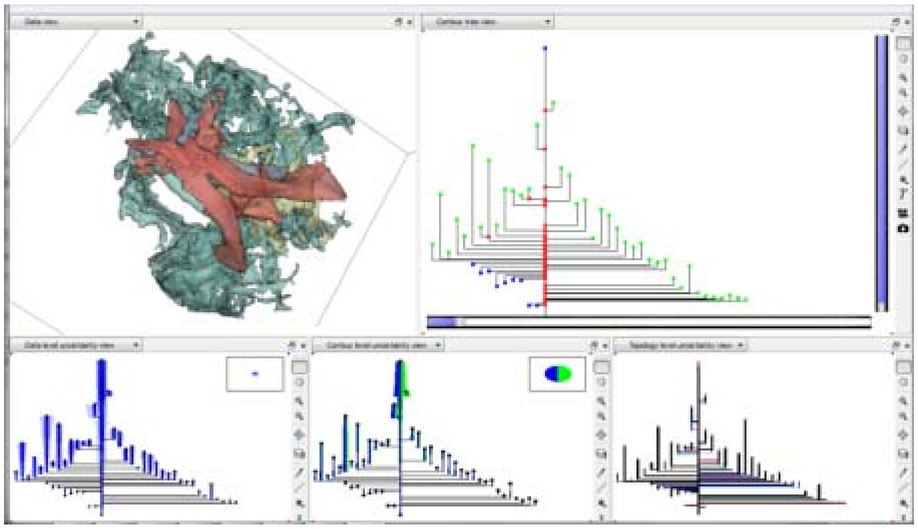


Figure 5.17 The user interface.

5.5.2 Contour tree simplification

Users may simplify the contour tree in two ways. (1) Use a horizontal bar under the contour tree display. It controls the current number of branches in the contour tree. As a user drags the slider from right to left, branches are removed from the contour tree according to the order shored in the pre-computed simplification sequence. (2) Directly right-click on the inner nodes of the contour tree to prune or extend sub-trees. In practice, a user may first slide through the horizontal bar to get a certain simplification level; then, work through the tree view graph to show or hide sub-trees according to the uncertainty information attached.

5.5.3 Contour selection

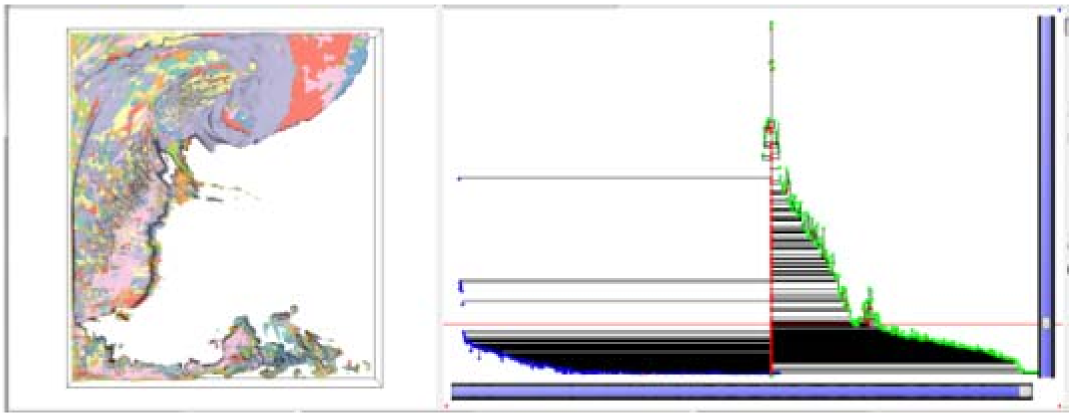
Users may select contours to display in two ways. (1) Display a set of corresponding contours (spaghetti plots) with the same iso-value in different ensemble members in the data display area by clicking on a location in the contour tree. For 3D data, users may choose to show overlapped iso-surfaces with or without transparency. The corresponding data-level and contour-level uncertainties for the selected contour are shown with circular uncertainty glyphs after each click. (2) Display a set of corresponding contours with the same iso-value in different ensemble members by sliding on the vertical bar on the right of the contour tree display. The value of the slider corresponds to the selected iso-value. The contours are chosen at the vertices on the current contour tree branches with the selected iso-value.

5.6 Applications and results

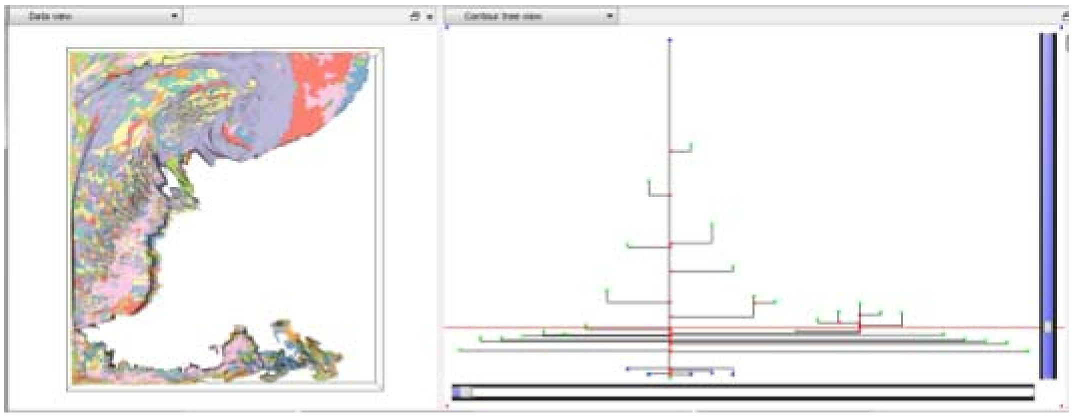
In this section, we demonstrate the effectiveness of the new uncertainty visualization by applying it to a WRF weather data set (135×135×30) and a medical volumetric data set (128×128×71). The contour trees, branch hierarchies, and uncertainty information are pre-computed before an interaction starts.

The first experiment demonstrates an effective application of the new uncertainty visualization on the simulated data from Weather Research and Forecast Environmental Modeling System (WRF) ensemble runs. The members of numerical weather prediction ensembles are individual simulations with either slightly perturbed initial conditions or different model parameterizations. Scientists use the average ensemble output as a forecast and utilize spaghetti plots to analyze the spread of the ensemble members. In our application, eight simulation runs of the 1993 “Super storm” are used. Figure 5.18 a shows the original contour tree (1734 critical points and 867 branches) and all the contours with the iso-value indicated by the red horizontal line connecting the slider of the vertical bar. Figure 5.18 b shows a manually simplified contour tree (66 critical points and 33 branches) and a few contours which corresponds to the vertices on the current branches crossed by the red line. Three levels of uncertainty are shown in the simplified contour tree in Figure 5.18c. The right contour tree in Figure 5.18c shows the topology-level uncertainty based on the contour tree. A few branches with high disagreement on heights appear in the upper region of the contour tree indicating the contour regions with higher value variation. A few thin branches appear at its bottom region indicating the uncertain contour region (or minima) in the data. Figure 5.18d shows the integrated visualization of both data and uncertainty glyphs. Figure 5.18e shows a set of corresponding contours with high data-level uncertainty while Figure 5.18f shows a set of corresponding contours with high contour-level uncertainty. Circular uncertainty glyphs indicating the magnitude of their data-level and contour-level uncertainties are shown at the bottom. The gray rectangle indicates the maximum size for each glyph. The color bar on the right indicates the correspondence between the iso-surfaces and simulation runs.

The second application is to visualize a non-weighted diffusion imaging dataset of human brains. We apply our method to study the difference between five brain images aligned after affine registration. The results are shown in Figure 5.19. The between-subject variation in brain anatomy is a type of uncertainty that is important to doctors [67]. The average data field contains 2143 critical points and 1071 branches in the original contour tree. Figure 5.19a shows the volume rendering of the average data with circular uncertainty glyphs whose sizes vary according to the level of data-level uncertainty. Three levels of uncertainty are shown in the simplified contour tree in Figure 5.19c. A set of corresponding contours (Figure 5.19b) in different brain data were generated after a user clicks on a node at the bottom part of the middle branch of the right contour tree in Figure 5.19 c. The high diversities among the selected contours reflect the high variations in iso-value of the branch. The contours of different ensemble members are shown in different colors.

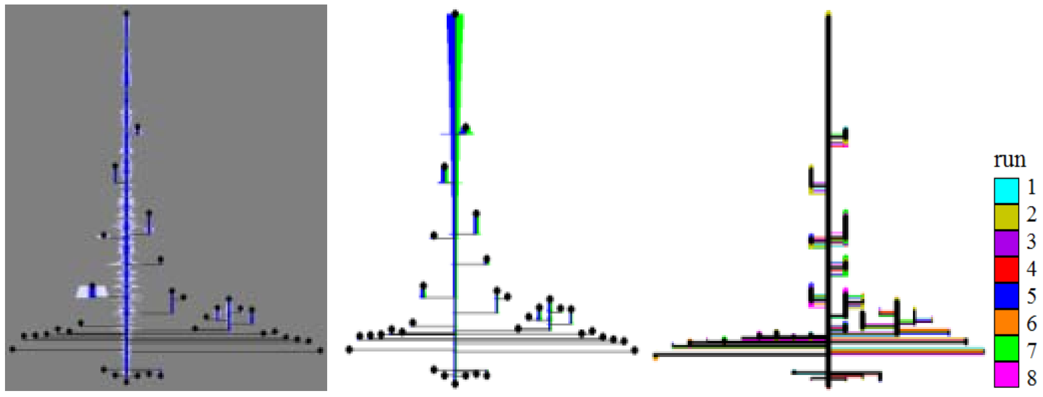


(a)

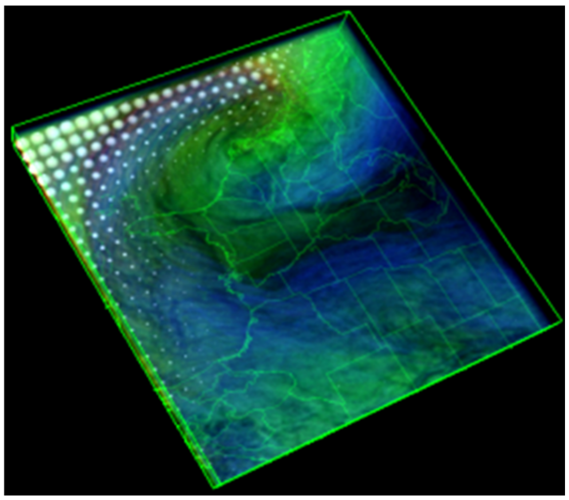


(b)

Figure 5.18 Weather data application. (a) Original contour tree. (b) Manually simplified contour tree with tree-view glyph interaction. (c) Left to right: data-level uncertainty, contour-level uncertainty, and topology-level uncertainty shown in the simplified contour tree. (d) Volume rendering with uncertainty glyphs. (e) A set of contours with high data-level uncertainty indicated by the large graduated circle. (f) A set of contours with high contour-level uncertainty indicated by the large glyph for contour-level uncertainty.

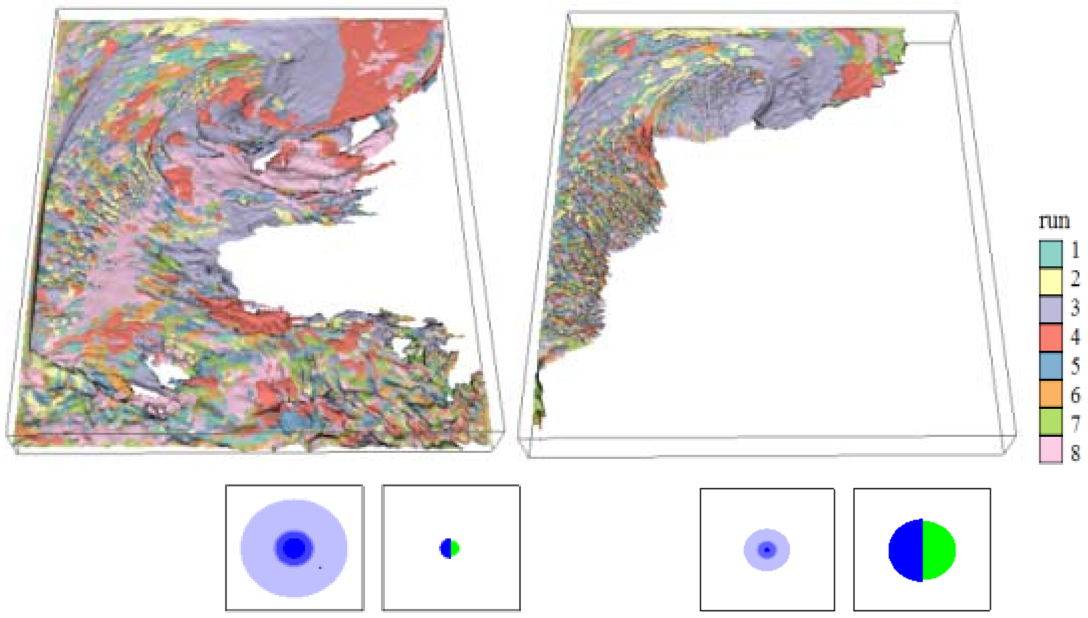


(c)



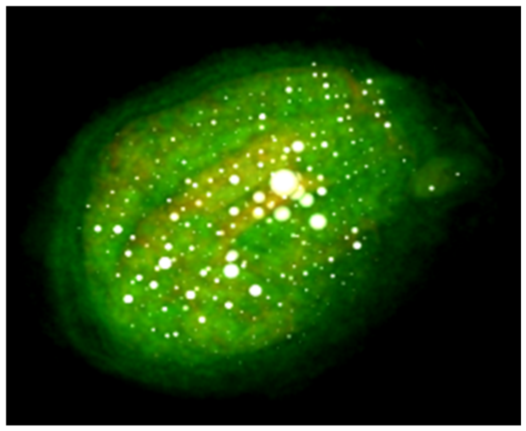
(d)

Figure 5.18 (continued)

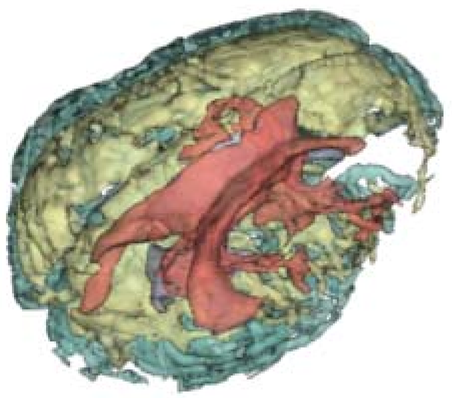


(e) (f)

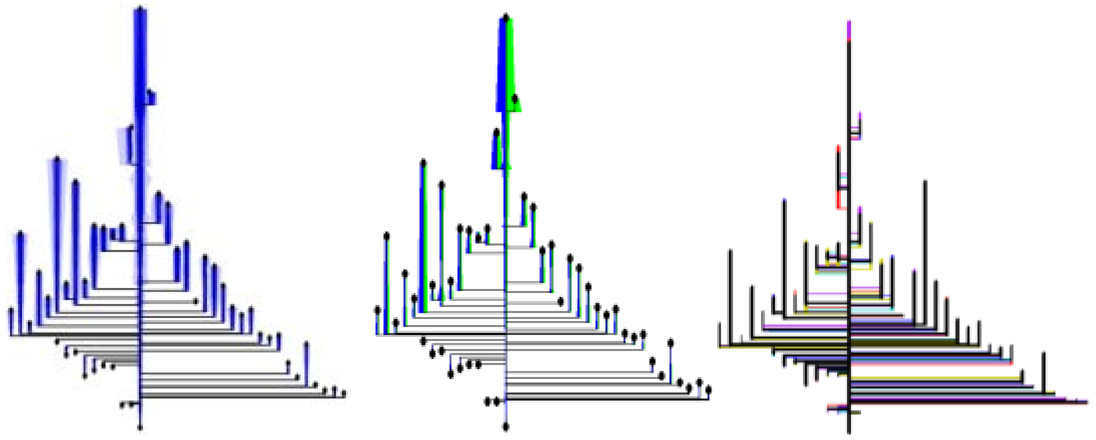
Figure 5.18 (continued)



(a)



(b)



(c)

Figure 5.19 Brain data with uncertainty. (a) Volume rendering with circular uncertainty glyphs. (b) Corresponding contours of the same iso-value in different brain data. (c) Left to right: data-level uncertainty, contour-level uncertainty, and topology-level uncertainty shown in a simplified contour tree.

As shown in the above applications, the new contour tree based visualization provides a combined exploration of both the data and the uncertainty. Users are allowed to look into three levels of uncertainty in the data. With the contour tree displays, the workload in viewing and analyzing 3D data or complicated 2D data with uncertainty is significantly reduced. In addition, explicitly showing the three levels of uncertainty in a planar contour tree layout helps users investigate the contours with high or low uncertainty more precisely. Volume rendering with circular uncertainty glyphs (Figure Figure 5.18d and 5.19a) are provided for a comparison purpose. With 3D glyphs placed within the volumetric data, the details of the data are noticeably blocked by the glyphs while the glyphs are overlapped with each other or appear to be blurred or buried in a 3D scene. Accordingly, our new visualization method provides an alternative which shows both data and uncertainty clearly.

5.7 Conclusion

We present a contour-tree-based visualization for exploring data with uncertainty. One contribution of the chapter is the novel usage of a topology tool — the contour tree. Free of occlusions, the new visualization provides a promising alternative to the standard spatial layout of both data and uncertainty. Another unique contribution of this chapter is a full exploration of uncertainties in scientific data with the concepts of the data-level uncertainty, contour-level uncertainty, and topology-level uncertainty based on the contour tree. This information provides new insight into how the uncertainty exists with and affects the underlying data. With quantified uncertainty information attached to the contour trees, users can precisely select contour with specific uncertainty to display. The interaction design based on the contour tree is applicable to other applications. The experimental results demonstrate its effective applications in exploring uncertainties in weather forecasting and medical imaging. In addition, the rectangular contour tree layout with tree view interaction is another contribution which assists the implementation of our interactive uncertainty visualization

As for future plans, we would like to investigate more metrics to measure each level of uncertainty and apply our methods to address different types of uncertainty models. In contour tree simplification methods [39], [40], [64], [56], error is introduced in the simplified version of a contour tree. The persistence which is used as the importance measure can be used as the topological error norm [56]. However, this error norm does not reflect the uncertainty information related to the simplified branches. One solution is to use the integral of the uncertainty magnitude over the contour region of a branch as a new error norm to guide a simplification. The contour tree is an abstract description of a scalar field, so the use of our visualization tools may require some specific training. Our interactive visualization tool may be further improved based on user feedback.

CHAPTER VI

CONCLUSIONS AND POSSIBLE FUTURE DIRECTIONS

This thesis has conducted an in-depth investigation of feature-level uncertainties and developed effective uncertainty visualizations by exploiting topology tools.

First, a framework of feature-level uncertainty visualization is presented to study the uncertainty of the topological features in Chapter 4. This framework works for both scalar and vector data in both 2D and 3D. It quantified feature-level uncertainties as feature deviations, solved the mapping between features by adopting tracking method based on Feature Flow Fields, and visualized the uncertainties with intuitive graduated elliptical glyphs. The 2D and 3D elliptical glyphs reveal the uncertain numbers and locations of the features effectively. As a result, we offered a feasible solution to visualizing feature-level uncertainties.

In Chapter 5, based on contour trees, we tackled the interaction and perception issue of uncertainties in 3D and large 2D scalar data with a new interactive. The experimental results demonstrate its potential for effectively exploring uncertainties in weather forecasting and medical imaging. The new techniques could significantly reduce a user's workload in viewing and analyzing data with uncertainty information and helps show a quick and accurate selection of prominent contours with high uncertainty. Attaching the uncertainty glyphs on contour trees solve the inherent perception issues, such as occlusion and clutter, of the integrated visualization with both data and uncertainty. Further, we conducted a full exploration of uncertainties in scientific data with the concepts of the data-level uncertainty and two feature-level uncertainties — contour-level uncertainty and topology-level uncertainty — based on contour trees. Additionally, the rectangular contour tree layout with tree view interaction provides a flexible interaction tool that assists users to explore their data. With quantified uncertainty information attached to the branches of the contour tree layout, users can precisely select contours with specific uncertainty to display. Therefore, the new visualization provides a promising alternative to the standard spatial layout of both data and uncertainty.

In conclusion, this thesis innovatively explores the feature-level uncertainties and suggested promising directions of future uncertainty studies in exploiting topology tools. The presented feature-based techniques alleviate the inherent perception issues such as clutter and occlusion in uncertainty visualizations in 3D or large 2D scenes. The incorporation of the feature-level uncertainties into visualization provides fundamental insights into the reliability of the extracted features which otherwise would remain unknown with the visualization of only data-level uncertainty. Particularly, the novel use of contour trees provides an effective solution for interacting with 3D or large 2D data sets with uncertainty.

There are many possible directions in which one could extend this work: (1) Apply the developed framework of feature-level uncertainty to study uncertain data of various fields. (2) Test and improve the interactive uncertainty visualization through a domain expert evaluation. (3) The possibilities inherent in topology tools are not exhausted yet. For example, the possibility of using Morse-Small complex for uncertainty study has not been investigated yet. Therefore, we will continue our current work in utilizing topology tools to study uncertainty.

REFERENCES

[1] A. Pang, C. Wittenbrink, and S. Lodha, "Approaches to Uncertainty Visualization," *The Visual Computer,* vol. 13, pp. 370–390, 1997.

[2] C. Wittenbrink, A. Pang, and S. Lodha, "Glyphs for visualizing uncertainty in vector fields," *IEEE Transactions on Visualization and Computer Graphics,* vol. 2, pp. 266–279, 1996.

[3] C. Garth and X. Tricoche, "Topology- and Feature-based Flow Visualization: Methods and Applications " in *SIAM Conference on Geometric Design and Computing*, 2005.

[4] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Comput. Geom. Theory Appl.,* vol. 24, pp. 75-94, 2003.

[5] H. Carr and J. Snoeyink, "Path seeds and flexible isosurfaces using topology for exploratory visualization," presented at the Proceedings of the symposium on Data visualisation 2003, Grenoble, France, 2003.

[6] C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The contour spectrum," in *Visualization '97., Proceedings*, 1997, pp. 167-173.

[7] A. M. MacEachren, "Visualizing uncertain information," *Cartographic Perspective,* vol. 13, pp. 10–19, 1992.

[8] C. R. Ehlschlaeger, A. M. Shortridge, and M. F. Goodchild, "Visualizing spatial data uncertainty using animation," *Computers & Geosciences,* vol. 23, pp. 387 – 395, 1997.

[9] T. Hengl and N. Toomanian, "Maps are not what they seem: representing uncertainty in soil-property maps," in *7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, 2006, pp. 805–813.

[10] T. J. Davis and C. P. Keller, "Modeling and visualizing multiple spatial uncertainties," *Computers & Geosciences,* vol. 23, pp. 397-408, 1997.

[11] S. Djurcilova, K. Kima, P. Lermusiauxb, and A. Pang, "Visualizing scalar volumetric data with uncertainty," *Computers and Graphics,* vol. 26, pp. 239-248, 2002.

[12] J. Sanyal, S. Zhang, G. Bhattacharya, P. Amburn, and R. Moorhead, "A User Study to Compare Four Uncertainty Visualization Methods for 1D and 2D Datasets," *IEEE Transactions on Visualization and Computer Graphics,* vol. 15, pp. 1209-1218, 2009.

[13] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. J. Moorhead, "Noodles: A Tool for Visualization of Numerical Weather Model Ensemble Uncertainty," *IEEE Transactions on Visualization and Computer Graphics,* vol. 16, pp. 1421-1430, 2010.

[14] A. Cedilnik and P. Rheingans, "Procedural annotation of uncertain information," in *Visualization, 2004. IEEE*, 2000, pp. 77–84.

[15] G. Grigoryan and P. Rheingans, "Point-based probabilistic surfaces to show surface uncertainty," *IEEE Transactions on Visualization and Computer Graphics,* vol. 10, pp. 564-573, 2004.

[16] G. S. Schmidt, S.-L. Chen, A. N. Bryden, M. A. Livingston, B. R. Osborn, and L. J. Rosenblum, "Multidimensional Visual Representations for Underwater Environmental Uncertainty," *IEEE Computer Graphics and Applications,* vol. 24, pp. 56-65, 2004.

[17] R. Kosara, S. Miksch, H. Hauser, J. Schrammel, V. Giller, and M. Tscheligi, "Useful properties of Semantic Depth of Field for better F+C visualization," presented at the Proceedings of the symposium on Data Visualisation 2002, Barcelona, Spain, 2002.

[18] P. J. Rhodes, R. S. Laramee, R. D. Bergeron, and T. M. Sparr, "Uncertainty Visualization Methods in Isosurface Rendering," presented at the Eurographics 2003, 2003.

[19] T. Pfaffelmoser, M. Reitinger, and R. Westermann, "Visualizing the Positional and Geometrical Variability of Isosurfaces in Uncertain Scalar Fields," presented at the Eurographics / IEEE Symposium on Visualization 2011 (EuroVis 2011), 2011.

[20] M. Pauly, N. J. Mitra, and L. Guibas, "Uncertainty and Variability in Point Cloud Surface Data," presented at the Eurographics Symp. Point-Based Graphics, 2004.

[21] K. Pöthkow and H.-C. Hege, "Positional Uncertainty of Isocontours: Condition Analysis and Probabilistic Measures," *IEEE Transactions on Visualization and Computer Graphics,* vol. 17, pp. 1393-1406, 2011.

[22] K. Pöthkow, B. Weber, and H.-C. Hege, "Probabilistic Marching Cubes," *Computer Graphics Forum,* vol. 30, pp. 931-940, 2011.

[23] M. Otto, T. Germer, H.-C. Hege, and H. Theisel, "Uncertain 2D Vector Field Topology," *Computer Graphics Forum,* vol. 2, pp. 347--356, 2010.

[24] I. A. Sadarjoen and F. H. Post, "Detection, quantification, and tracking of vortices using streamline geometry," *Computers & Graphics,* vol. 24, pp. 333-341, 2000.

[25] P. Majer, "A Statistical Approach to Feature Detection and Scale Selection in Images," PhD, Georg-August Universitt G¨ottingen, 2000.

[26] C. Garth and X. Tricoche, "Topology- and Feature-based Flow Visualization: Methods and Applications," presented at the SIAM Conference on Geometric Design and Computing, 2005.

[27] J. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *IEEE Computer Graphics and Applications,* vol. 22, pp. 27-36, 1989.

[28] X. Tricoche, "Vector and Tensor Field Topology Simplification, Tracking and Visualization," PhD, Universität Kaiserslautern, Germany 2002.

[29] R. F. W. Bader, T. T. Nguyen-Dang, and Y. Tal, "Quantum topology of molecular charge distributions ii. Molecular structure and its charge," *Journal of Chemical Physics,* vol. 70, pp. 4316–4329, 1979.

[30] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A topological hierarchy for functions on triangulated surfaces," *IEEE Transactions on Visualization and Computer Graphics,* vol. 10, pp. 385-396, 2004.

[31] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci, "Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees," *IEEE Transactions on Visualization and Computer Graphics,* vol. 15, pp. 1177-1184, 2009.

[32] S. Smale, "On gradient dynamical systems," *Annals of Mathematics,* vol. 71, pp. 199-206, 1961 1961.

[33] A. Gyulassy, V. Natarajan, V. Pascucci, P. T. Bremer, and B. Hamann, "Topology-based simplification for feature extraction from 3D scalar fields," in *Visualization, 2005. VIS 05. IEEE*, 2005, pp. 535-542.

[34] G. H. Weber, P. T. Bremer, and V. Pascucci, "Topological Landscapes: A Terrain Metaphor for Scientific Data," *Visualization and Computer Graphics, IEEE Transactions on,* vol. 13, pp. 1416-1423, 2007.

[35] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," presented at the Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987.

[36] M. v. Kreveld, R. v. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore, "Contour trees and small seed sets for isosurface traversal," presented at the Proceedings of the thirteenth annual symposium on Computational geometry, Nice, France, 1997.

[37] I. Fujishiro, T. Azumay, and Y. Takeshimay, "Automating transfer function design for comprehensible volume rendering based on 3D field topology analysis," in *Visualization '99. Proceedings*, 1999, pp. 467-563.

[38] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote, "Simple and optimal output-sensitive construction of contour trees using monotone paths," *Comput. Geom. Theory Appl.,* vol. 30, pp. 165-195, 2005.

[39] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli, "Multi–resolution computation and presentation of contour tree," in *the IASTED conference on Visualization, Imaging, and Image*, 2004, pp. 452–290.

[40] H. Carr, J. Snoeyink, and M. v. d. Panne, "Simplifying flexible isosurfaces using local geometric measures," in *Visualization, 2004. IEEE*, 2004, pp. 497-504.

[41] D. Schneider, A. Wiebel, H. Carr, M. Hlawitschka, and G. Scheuermann, "Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization," *Visualization and Computer Graphics, IEEE Transactions on,* vol. 14, pp. 1475-1482, 2008.

[42] B. S. Sohn and B. Chandrajit, "Time-varying contour topology," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 14-25, 2006.

[43] C. Heine, D. Schneider, H. Carr, and G. Scheuermann, "Drawing Contour Trees in the Plane," *IEEE Transactions on Visualization and Computer Graphics,* vol. 17, pp. 1599-1611, 2011.

[44] H. Theisel and H.-P. Seidel, "Feature flow fields," in *Data Visualization 2003. Proc. VisSym 03*, 2003, pp. 141–148.

[45] C. Garth, X. Tricoche, and G. Scheuermann, "Tracking of vector field critical points in unstructured 3D time-dependent datasets," in *Proc. IEEE Visualization 2004*, 2004, pp. 329–336.

[46] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Topological methods for 2D time-dependent vector fields based on stream lines and path lines," *IEEE Transactions on Visualization and Computer Graphics,* vol. 11, pp. 383–394, 2005.

[47] J. Sahner, T. Weinkauf, and H.-C. Hege, "Galilean invariant extraction and iconic representation of vortex core lines," in *Proc. EuroVis 2005*, 2005, pp. 151–160.

[48] H. Theisel, C. Rössl, and H.-P. Seidel, " Combining topological simplification and topology preserving compression for 2d vector fields," in *Proc. Pacific Graphics 2003*, 2003, pp. 419-423.

[49] H. Theisel, C. Rössl, and H.-P. Seidel, "Using feature flow fields for topological comparison of vector fields," in *Proc. Vision, Modeling and Visualization 2003*, 2003, pp. 521–528.

[50] J. Bertin, *Semiology of Graphics*. WI: The University of Wisconsin Press, 1983.

[51] E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, CT: Graphics Press, 2001.

[52] C. Ware, *Information Visualization: Perception for Design*, 2nd ed.: Morgan Kaufmann Publishers, 2004.

[53] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 1053-1060, 2006.

[54] A. Gyulassy, N. Vijay, V. Pascucci, P. T. Bremer, and B. Hamann, "A topological approach to simplification of three-dimensional scalar functions," *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, pp. 474-484, 2006.

[55] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Morse-smale complexes for piecewise linear 3-manifolds," presented at the Proceedings of the nineteenth annual symposium on Computational geometry, San Diego, California, USA, 2003.

[56] P.-T. Bremer, "Topology–based Multi–resolution Hierarchies," PhD, University of California at Davis, Davis, CA, USA, 2004.

[57] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda, "Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data," *Computer Graphics Forum,* vol. 14, pp. 181–192, 1995.

[58] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Saddle Connectors - An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields," in *IEEE Visualization 2003*, 2003, pp. 225-232

[59] G. M. Nielson, "Modeling and Visualizing Volumetric and Surface-on-Surface Data," presented at the Focus on Scientific Visualization, 1993.

[60] R. Franke and G. Nielson, "Smooth and interpolation of large sets of scattered data," *InternationalJournal for Numerical Methods in Engineering,* p. 1691~1704, 1980.

[61] Y. Rubner, C. Tomasi, and L. J. Guibas, "A Metric for Distributions with Applications to Image Databases," in *IEEE International Conference on Computer Vision*, Bombay, India, 1998, pp. 59-66.

[62] T. v. Walsum, F. H. Post, D. Silver, and F. J. Post, "Feature Extraction and Iconic Visualization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 2, pp. 111-119, 1996.

[63] P. Diggle, P. Heagerty, K.-Y. Liang, and S. Zeger, *Analysis of Longitudinal Data*: Oxford University Press, USA, 2002.

[64] S. Takahashi, Y. Takeshima, and I. Fujishiro, "Topological volume skeletonization and its application to transfer function design," *Graph. Models,* vol. 66, pp. 24-49, 2004.

[65] G. d. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*: Prentice-Hall, 1999.

[66] Microsoft. (2011). *Tree Views*. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa511496.aspx

[67] S. B. Eickhoff, A. R. Laird, C. Grefkes, L. E. Wang, K. Zilles, and P. T. Fox, "Coordinate-Based Activation Likelihood Estimation Meta-Analysis of Neuroimaging Data: A Random-Effects Approach Based on Empirical Estimates of Spatial Uncertainty," *Human Brain Mapping,* vol. 30, pp. 2907-2926, 2009.