International Journal for Uncertainty Quantification, 3 (3): 203–223 (2013)

A CONTOUR TREE BASED VISUALIZATION FOR EXPLORING DATA WITH UNCERTAINTY

Keqin Wu & Song Zhang*

The GeoSystems Research Institute, Mississippi State University, Starkville, Mississippi 39762, USA

Original Manuscript Submitted: 09/14/2011; Final Draft Received: 08/03/2012

Uncertainty is a common and crucial issue in scientific data. The exploration and analysis of three-dimensional (3D) and large two-dimensional (2D) data with uncertainty information demand an effective visualization augmented with both user interaction and relevant context. The contour tree has been exploited as an efficient data structure to guide exploratory visualization. This paper proposes an interactive visualization tool for exploring data with quantitative uncertainty representations. First, we introduce a balanced planar hierarchical contour tree layout integrated with tree view interaction, allowing users to quickly navigate between levels of detail for contours of large data. Further, uncertainty information is attached to a planar contour tree layout to avoid the visual cluttering and occlusion in viewing uncertainty in 3D data or large 2D data. For the first time, the uncertainty information is explored as a combination of the data-level uncertainty which represents the uncertainty concerning the numerical values of the data, the contour variability which quantifies the positional variation of contours, and the topology variability which reveals the topological variation of contour trees. This information provides a new insight into how the uncertainty exists with and relates to the features of the data. The experimental results show that this new visualization facilitates a quick and accurate selection of prominent contours with high or low uncertainty and variability.

KEY WORDS: *Uncertainty visualization, contour tree, topology simplification, weather ensemble, volumetric data*

1. INTRODUCTION

Visualizing uncertainty facilitates data analysis and decision making in various areas. However, exploring three-dimensional (3D) or large two-dimensional (2D) data with uncertainty is challenging due to the increased complexity of the data. Current uncertainty visualizations generally 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 or information overload due to the large amount of information and the interference between the data and its uncertainty. We believe that one promising direction to cope with this challenge is to allow users to explore data hierarchically and to provide informative clues about where to look.

The question of how to quantify uncertainty appropriately is challenging. 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 [2, 3]. While these techniques achieve decent visualization results, they do not provide an insight into how much uncertainty affects the features in the data. Contours, isolines or isosurfaces, are features frequently investigated for exploring data with uncertainty [4, 5]. For instance, uncertainty in climate modeling is often represented by ensembles which contain multiple results for the simulated quantities. Rendering contours from all ensembles in a single image, known as spaghetti plots [6], is a conventional technique used by meteorologists for observing uncertainty in their simulations. Users are often interested in the variability of the contours, e.g., how different are the contours of different ensemble members? This paper focuses on new methods for

*Correspond to Song Zhang, E-mail: szhang@cse.msstate.edu

2152–5080/13/$35.00 © 2013 by Begell House, Inc.

visualizing the uncertainties in scalar fields. Though many visualization techniques have been developed for uncertain scalar fields, the variability related to topological features, e.g., the topological structure of the scalar data, is barely studied. Uncertainty has different sources and types. The uncertainty introduced as data-level uncertainty propagates in visualization processes. We therefore analyze uncertainty-related information on three levels: on data level, we study the uncertainty of the data; on contour level, we quantify the positional variation of the contours; and on topology level, we reveal the variability of the contour trees.

A contour tree is an important topology tool which stores the nesting relationships of the contours in a scalar field. It is frequently used to visualize data hierarchically, to accelerate contour extraction, and to serve as a user interface for exploring data [7–10]. 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 longstanding perception issues such as clutter and occlusion in many uncertainty visualizations in 3D or large 2D scenes. Second, a contour tree stores the information related to the geometry of individual contours, which can be utilized to compute the positional variation of contours. Third, investigating the variable structure of the contour tree reveals the topology variability 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 or variability. 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 layout and interaction of contour trees are highly active research topics as well [7–9, 11]. A major challenge in designing the layout for a large contour tree is the large number of branches with self-intersections. Contour tree simplification alleviates the size issue, but an efficient navigation through different simplification levels is required. A 3D orrery-like layout [7] was presented, but it is less maneuverable than a 2D layout and occlusion occurs due to edge crossings in projection. Therefore, we propose a planar and interactive contour tree display in this paper. Though bearing a resemblance to Heine et al.'s orthogonal layout [12], our layout algorithm does not require an optimization step and is thus more time efficient. The new contour tree layout innovatively adopts the form of tree view interaction so that users can easily interact with a contour tree to obtain any desired simplification level.

The core of this paper is the use of contour trees as a tool to represent uncertainty and select contours accordingly. 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 and (2) a new paradigm of investigating and visualizing uncertainty based on a contour tree that integrates the uncertainty or variability representations on the data level, the contour level, and the topology level.

2. RELATED WORK

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 psychovisual approaches.

Several efforts have been made to identify potential visual attributes for uncertainty visualization. For example, Davis and Keller [13] suggested value, color, and texture for representing uncertainty on static maps. Djurcilov et al. [14] used opacity deviations and noise effects to provide qualitative measures for the uncertainty in volume rendering. Sanyal et al. [15] 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 glyph and ribbon to encode uncertainty information of weather simulations [4].

Several methods have been developed to address the uncertainty or variability of the size, position, and shape of contours [5, 16, 17]. 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. [18] quantified and visualized the uncertainty introduced in the reconstructions of surfaces from point cloud data. Pfaffelmoser et al. [17] presented a method for visualizing the positional variability

around a mean isosurface using direct volume rendering. A method to compute and visualize the positional uncertainty of contours in uncertain input data has been suggested by Pthkow and Hege [19]. Assuming certain probability density function, they modeled a discretely sampled uncertain scalar field by a discrete random field. Pthkow et al. [20] extended their model to correlated random fields.

Contour tree is a powerful visualization tool for abstract data representations [11], contour extractions [21, 22], and transfer function design [23], etc. A number of algorithms have been used to compute contour trees [22, 24, 25]. Among them, we choose to implement Carr et al.'s join-split algorithm [25] to construct contour trees because it is a simple, robust, and fast algorithm that works on simplicial mesh of arbitrary dimension and requires no critical points to be precomputed. 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 [7, 8]. 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 [9, 26, 27]. The overlapped volume or area between contours is often used to compare or map contours of different scalar fields [26, 27].

Utilizing contour tree as an interface to explore data was first suggested by Bajaj et al. [9]. They introduced the "contour spectrum" to facilitate isovalue selection based on isosurface properties such as isoline length and isosurface area. Carr et al. [8, 10] proposed "path seeds" to facilitate a selection of distinct contours and "flexible isosurface" 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. [7] 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 layouts. In addition, a planar display allows for easier selection of an object and therefore is more desirable for an interface design. Heine et al. [12] 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.

In this paper, we propose to investigate data-level uncertainty, contour variability, and topology variability 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 representation can be adapted to different uncertainty models, in this paper, we measure the uncertainty and variability according to the differences between the data values, contours, or contour trees of different ensemble members or data sets.

3. CONTOUR TREE BACKGROUND

This section briefly reviews the theoretical background related to contour trees.

3.1 Level Set and Contours

Consider a smooth function $f : D \rightarrow R$ defined on a domain D. The level set of f for a given constant value c is the set $L(c) = \{x \in D | f(x) = c\}$. A point $v \in D$ is critical when the gradient of function $f(v)$ vanishes at this point.

One fundamental technique to reveal the behavior of f is to extract contours. A contour is a single connected component of the level set $L(c)$. Iso-lines are contours in 2D while isosurfaces are contours in 3D. As c increases, contours appear at minima, join or split at saddles, and disappear at local maxima of f.

3.2 Contour Tree

The contour tree is a loop-free Reebgraph that tracks the evolution of contours. Each leaf node is a minimum or maximum; each interior node is a saddle; each edge represents a set of adjacent contours with isovalues between the values of its two ends. There is a one-to-one mapping from a point in the contour tree (at a node or in an edge) to

a contour of the scalar field. Usually, in a contour tree based visualization, color is used to indicate the relationship between individual contours and the contour tree [8, 10]. A contour tree example is shown in Fig. 1.

For a contour tree whose critical points are nondegenerate, the contour tree has the following properties [28, 29]. (1) A maximum has only one edge whose opposite endpoint is a saddle or a minimum lower than it. (2) A minimum has only one edge whose opposite endpoint is a saddle or a 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. Note that to ensure these properties, one needs to resolve a multiple saddle where more than two contours merge or exist into simple ones [30]. Defining f as a linear interpolation over a simplicial mesh with unique data values at vertices ensures that f is a Morse function whose critical points are nondegenerate [25].

A fully augmented contour tree is a contour tree augmented by all the vertices in the simplicial mesh [31]. In this paper, the term contour tree refers to unaugmented contour tree whose nodes are merely critical points. We adopt Carr et al.'s join-split algorithm [25] for contour tree construction which sweeps through the data twice to compute a join tree and a split tree before merging them into a contour tree. This algorithm is capable of constructing both fully augmented and unaugmented contour trees.

3.3 Contour Tree Simplification

Simplification is introduced for the contour trees that are too large or complicated to be studied or displayed directly [7, 8]. 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). Topology integrity [15] and importance ranking [8] are two important criteria for the contour tree simplification.

Topology integrity ensures that the simplified topology is consistent with the original one. Takahashi et. al. [28] stated that the critical points must maintain topological integrity by satisfying the Euler formula. They suggested connecting all the boundary vertices of the data to a virtual minimum with value $-\infty$ so that a function becomes a topological sphere [28]. The Euler formula states that the critical point number of a 2D sphere satisfies $\#\{maxima\} - \#\{saddles\} + \#\{minima\} = 2$ and the critical point number of a topological 3D sphere [29] satisfies $\#\{maxima\} - \#\{2 - saddles\} + \#\{1 - saddles\} - \#\{minima\} = 0$. For a contour tree that satisfies the Euler equation, a simplified contour tree through cancellations of the aforementioned pair candidates satisfies the Euler equation as well.

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 [8].

4. CONTOUR TREE LAYOUT AND TREE VIEW GRAPH DESIGN

In this paper, we visualize uncertainty and variability information through contour trees. The layout of a contour tree becomes an issue when the size and complexity of the contour tree increase [12]. It is most intuitive to use the *y* axis to

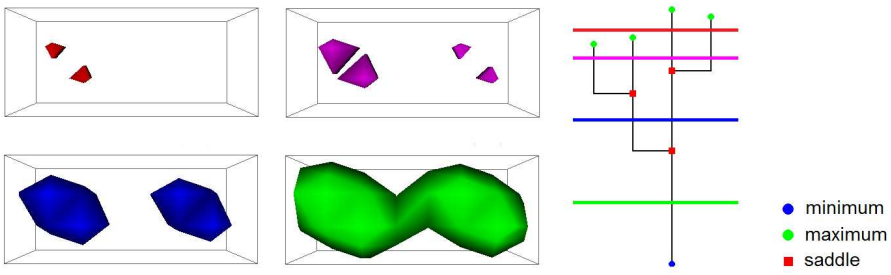


FIG. 1: Contour tree and contours of a 3D scalar field. Each horizontal line cuts exactly an edge of the tree for every contour at the corresponding isovalue. Color is used to indicate the correspondence between a line and its corresponding contours.

encode the scalar value of a function in a planar display [9]. Heine et al. [12] suggested minimizing edge crossings as one of the planar contour tree drawing criteria. However, as pointed out in previous literature [7, 10], self-intersections and visual saturation are unavoidable in such a display. It is hard for a radial tree graph display [32] 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 displays uncertainty information. To meet these requirements, we propose a 2D contour tree layout which is integrated with the tree view graph interaction.

In this section, a few definitions are given first. A detailed discussion of the contour tree layout and tree view graph design follows.

4.1 Simplification Sequence and Branch Hierarchy

With traditional bottom-up simplification [8], one simplifies a contour tree by canceling saddle-extremum pairs with increasing persistence. Given an original contour tree CT_0, after cancellations of a sequence of pairs $P_1, ..., P_n$, the simplified contour tree is CT_n. The last canceled pair P_n is a minimum and maximum pair.

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 nondecreasing (or nonincreasing) value of f until it reaches the highest (or lowest) node in the path. Pascucci et al. [7] and Weber et al. [11] 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 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 in value of f or *downward* if its root is higher than its opposite end. The subtree which consists of all the edges and nodes along a branch and its descendants is called the *subtree* of the branch. Further, we define the *length* of a branch as the persistence of its two ends—the absolute difference of their isovalues. Figure 2 gives examples of these definitions.

In this paper, we make the terms 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 3 illustrates the relation between branch hierarchies and simplification sequences. Figure 3(a) shows a bottom-up simplification [8] by repeatedly pruning off the shortest branch on a current contour tree. Figure 3(b) 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: 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 3(c) is reproduced from a example in [7] and 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,

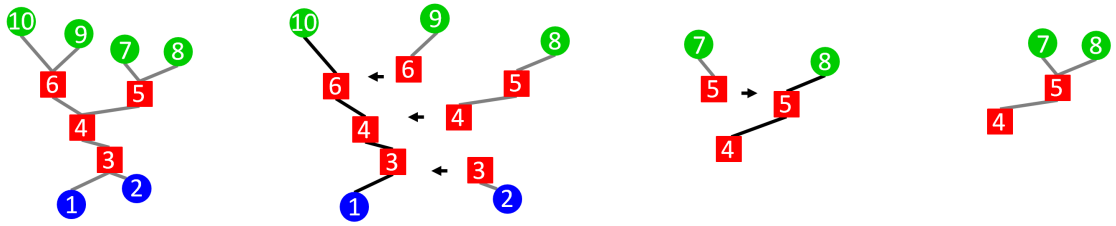
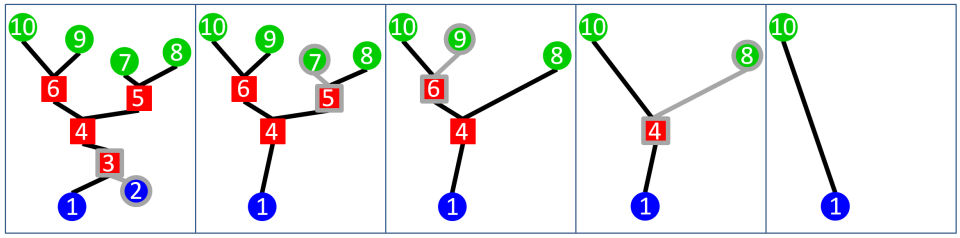
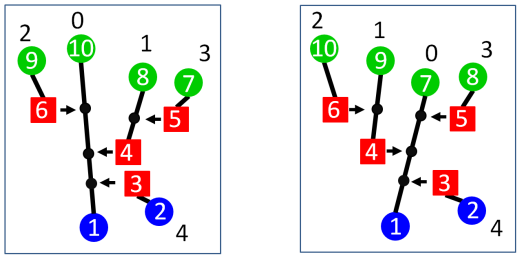


FIG. 2: Definitions related to branches. (a) Contour tree. (b) Parent branch (1, 10) with two upward child branches (6, 9) and (4, 8) and one downward child branch (3, 2). Branch (1, 10) is longer than its child branches. (c) Parent branch (4, 8) with one upward child branch (5, 7). Branch (4, 8) is longer than branch (5, 7). (d) Subtree of branch (4, 8). Nodes are numbered by their function values.



(a)



(b) (c)

FIG. 3: Branch hierarchies built according to reversed simplification sequences. (a) Bottom-up simplification [8] by pruning off the shortest branch on the current contour trees. (b) Hierarchy built from the reversed simplification sequence of (a). (c) Hierarchy built according to the branch decomposition [7]. Nodes are numbered by their values.

we choose to conduct the bottom-up simplification instead of the branch decomposition [7] to build a contour tree hierarchy.

4.2 2D Contour Tree Layouts

Our planar contour tree layout is proposed to show intuitive branch hierarchy and height information with minimized branch crossings. Nodes are positioned on the y axis according to their function values. It can be integrated with the tree view graph interaction to facilitate a fast navigation through different levels of simplification (discussed in Section 4.3). Branches can be attached with uncertainty glyphs (discussed in Section 5). We describe our strategies to emphasize branch hierarchy and to control self-intersections as follows.

The key to prevent unnecessary self-intersections is to recursively assign a vertical slot to a branch so that its child branches are contained entirely within the slot. An example is shown in Fig. 4(d). 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.

To emphasize the hierarchy, a branch is rendered in the middle, and its child branches are spread out to its left and right sides. The long branch spanning the last cancellation pair P_n is drawn as a vertical line segment in the middle of the display. All the other branches have L shapes that connect extremum to their paired saddles to prevent them 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 [Fig. 4(a)]. An easy way to avoid such intersections is to put upward branches and downward branches on different sides 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 other. Symmetrically, a lower downward branch is assigned a slot closer to its parent branch than higher downward branches. As shown in Fig. 4(b), this strategy effectively prevents all the intersections between the child branches. However, it is not a space-saving solution. An improved solution is given in Fig. 4(c) which takes less horizontal space than the tree layout in Fig. 4(b). For a given parent branch, we separated it into three parts vertically:

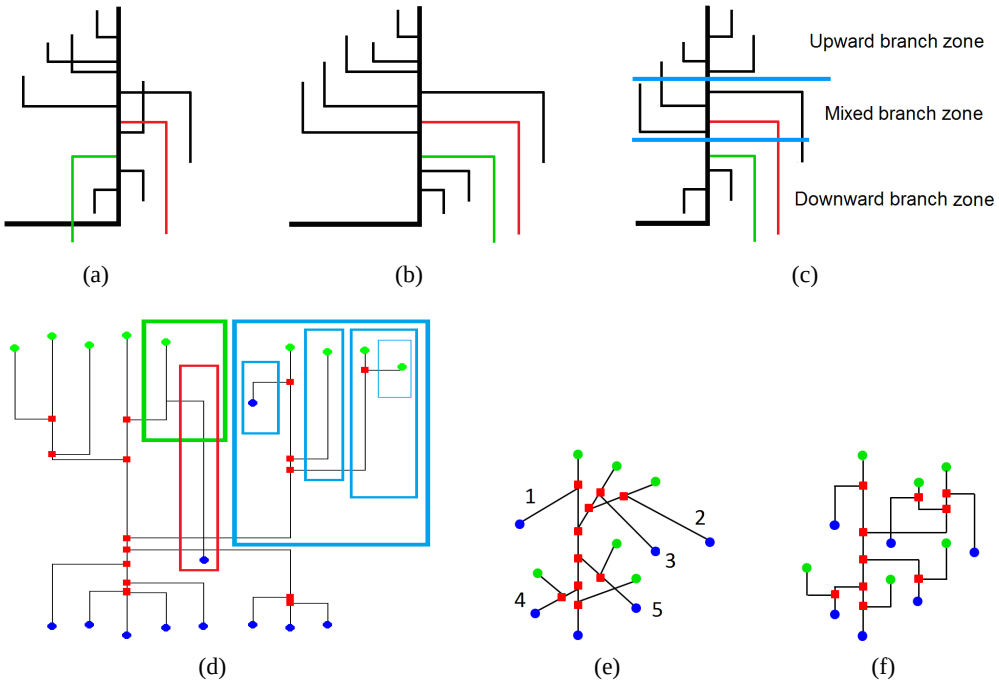


FIG. 4: Strategies to reduce self-intersections. (a) Unguided placement of child branches with multiple branch crossings. (b) Placing upward child branches and downward child branches on the different sides. (c) Placing upward child branches and downward child branches on the different sides only in the mixed branch zone. (d) 2D layout of a contour tree with each branch assigned in nested vertical slots. Boxes indicate vertical slots assigned to some branches. Unavoidable self-intersection is shown in the red box. (e) 2D layout of a contour tree with five strangulations. (f) New display of the contour tree in (e).

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, and downward branch zone where all the child branches are downward. In the mixed branch zone, the upward branches take one side while the downward branches take the other. In the upward branch or downward branch zone, the child branches stretch outward from the parent branch on both the 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 Fig. 4. 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 Fig. 4. If this happens, we switch such child branch to the right side. If the child branch is located in a mixed zone, 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 Fig. 4(c), 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. However, 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 Fig. 4(d), a strangulation branch (indicated in the red box) is so long that it reaches the branches outside its parent branch (indicated in the green box). This happens infrequently. Figure 4(e) shows a contour tree example that always exhibits at least one self-intersection. In Fig. 4(f), our layout reduces the number of crossings from three to one.

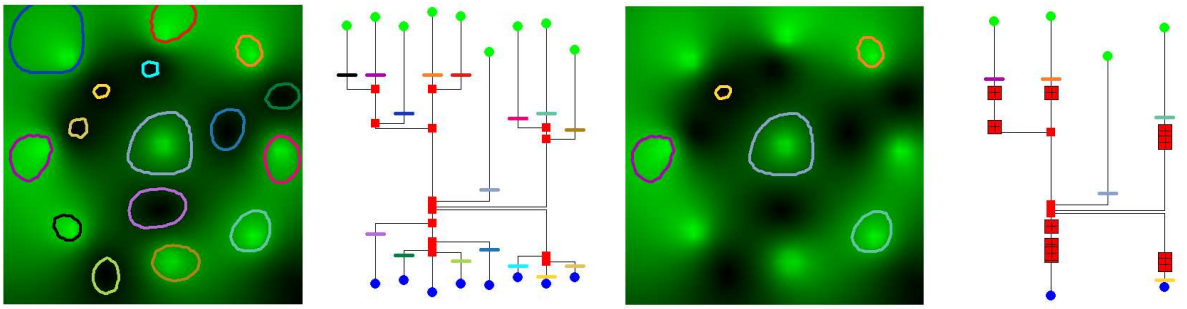
We notice that our layout appears to be similar to the orthogonal layout recently developed by Heine et al. [12] which shows L-shape edges with values mapped to heights and branches grouped hierarchically. They adopted a delicate but time consuming four-phrase procedure: grouping branches, minimizing crossings between branch groups, ordering, and positioning branches. The second phase, which uses a random walk combined with simulated annealing to minimize the crossing is nondeterministic and takes most of the runtime. As stated in [12], computing the layout for a contour tree with 2000 critical points could take up to 10 min 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 tree branches 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 2000 critical points. For example, for the brain data set with 2142 critical points in Fig. 13, it takes 343 milliseconds and produces no branch crossing; for the weather simulation data set with 1724 critical points in Fig. 14, 312 milliseconds, no branch crossing.

4.3 Tree View Interaction Design

A typical tree view graph displays a hierarchy of items by indenting child items beneath their parents [33]. In tree view representations, the interactions are directly embedded: the user can collapse (or expand) a particular subtree to hide (or show) the items in it.

Our contour tree layouts based on the nested subtrees can be directly integrated with a tree view interaction. It is not hard to prove that the number of the critical points in a subtree of a saddle-extremum branch satisfies $\#\{maxima\} - \#\{saddles\} + \#\{minima\} = 0$ for a 2D case and $\#\{maxima\} - \#\{2 - saddles\} + \#\{1 - saddles\} - \#\{minima\} = 0$ for a 3D case. Therefore, any subtree of a saddle-extremum branch could be collapsed or expanded with the Euler equation unchanged and the topological integrity of the contour tree unaffected.

An example of data exploration through tree view intersection is given in Fig. 5. The data are a vorticity magnitude field of a simulated flow with vortices. A click on an inner node of the original contour tree [Fig. 5(a) right] results in hiding or showing the subtree 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 Fig. 5(b). The roots of the collapsed subtrees 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 isovalue $f_{iso} = tf(s) + (1 - t)f(u), (0 \leq t \leq 1)$ is extracted. A contour in the scalar field is represented by a point in the tree (indicated by a horizontal line segment in the same color as the contour). The selected contours are representative contours that give an overview of the whole scalar field. The more a contour tree is simplified, the higher level overview is obtained.



(a) (b)

FIG. 5: Tree view interaction. (a) Contour tree and a 2D scalar field from which this tree was constructed with contours selected for every branch of it. (b) Interactively simplified contour tree after clicking on several nodes and contours selected for every branch of it.

5. CONTOUR TREE BASED UNCERTAINTY VISUALIZATION

This section discusses uncertainty metrics and their visualization representations. The uncertainty of the data is explored by considering data-level uncertainty and by showing how much variability is introduced by this uncertainty to features like contours and contour trees. The uncertainty or variability information is attached to the new planar contour tree display to give a high-level overview of uncertainty and to allow a quick and accurate selection of contours with different levels of uncertainty or variability.

We call the scalar field obtained by averaging the values from all the ensemble members at each data point the ensemble mean and the contour tree of the ensemble mean the mean contour tree. In this paper, we show the data-level uncertainty by displaying the difference between each ensemble member and the ensemble mean at each data point or along a contour. For contour variability, 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 variability, we map the contour trees of all the ensemble members to the mean contour tree and show their discrepancy to indicate variability.

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

5.1 Data-Level Uncertainty

The data-level uncertainty measures how uncertain the numerical values of the data are. Uncertainty measures, such as standard deviation, interquartile range, and the confidence intervals, fall into this category. Sanyal et al. [4] visualized data-level uncertainties via circular or ribbonlike 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 ribbonlike graduated glyph. A circular glyph encodes the deviation of all ensemble members from the ensemble mean at a data point. Let the differences for ensemble members be d_1, d_2, ... , d_k. These difference values are then scaled according to the glyph spacing and sorted in increasing order to generate a new array D_1, D_2, ... , D_k. Starting with D_k, we render successively smaller glyphs with decreasing sizes, D_k, D_{k-1}, ... , D_1 and increasing saturation levels of blue, $1\backslash k$, $2\backslash k$, ... , 1. A graduated ribbon is constructed by interpolating between circular glyphs placed along an isoline in the data image.

We use the circular graduated glyph to show uncertainty at each data point. Figure 6 shows a 9×9 2D uncertain scalar data set which is a downsampled subregion of the data in Fig. 5. Figure 6(b) shows the fully augmented contour tree with all the data points. In Fig. 6(a), the mean field of eight members is color-mapped and overlaid with uncertainty glyphs at each data point. Likewise, as shown in Fig. 6(c), the glyph of a data point is attached to its corresponding location in the fully augmented contour tree.

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 ribbonlike 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}. Let the average difference from ensemble member i to the ensemble mean along C be $w_i = \sum_{j=1}^{l} d_{ij} \backslash l$. In order to produce a ribbon along the branch, w_1, w_2, ... , w_k 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 k, $k-1$, ... , 1 with increasing saturation levels, $1\backslash k$, $2\backslash k$, ... , 1. Figure 6(d) illustrates a segment of a graduated ribbon along a branch. As shown in Figs. 6(c) and 6(e), while a graduated circle illustrates the data-level uncertainty at a data point, the graduated ribbon provides continuous uncertainty representation along individual contours with less clutter. Therefore, we use ribbonlike glyphs for representing data-level uncertainty along contours in our implementation. It

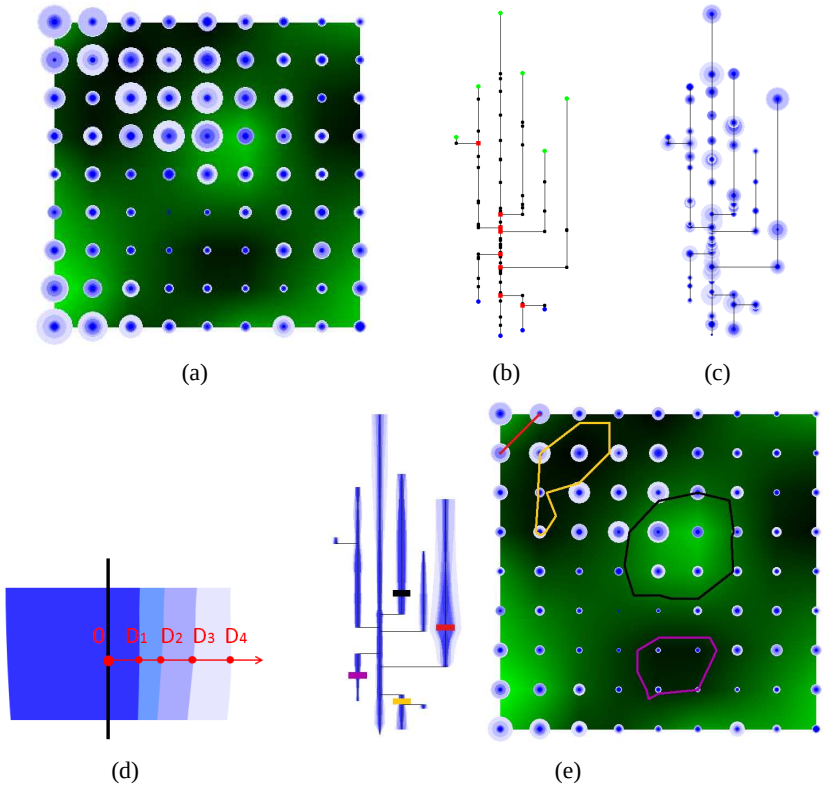


FIG. 6: Data-level uncertainty representation based on the contour tree. (a) Graduated circular uncertainty glyphs on an uncertain scalar field. (b) Fully augmented contour tree. Red, green, and blue dots indicate critical points while black dots indicate regular points. (c) Contour tree with circular graduated uncertainty glyphs. (d) 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.

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.

5.2 Contour Variability

In this paper, contour variability measures the variation in the position of a contour. In a spaghetti plot [4], 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 variability measurement is needed to assist the exploration of the large uncertain data. The contour variability which quantifies how diverse a contour is within multiple ensemble members is proposed to address this need.

To measure the variability 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 positional variation among contours. Our method is different from previous methods of studying uncertain contours which visualize probability field through color-mapping or volume rendering around a contour [17, 19, 20] since it

is visualized through contour trees and measures the variability as contour deviations in different ensemble members directly.

5.2.1 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 isovalue with it, or it may be missing in some ensemble members. Spatial overlap [26, 27] is frequently used as a similarity measure to match features in different data sets under an assumption that these features only have small spatially deviation. For instance, the correspondence between two contours can be measured by the degree of contour overlap as discussed by Sohn and Bajaj in [27] 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 $C_i(i = 1, ..., k)$ that shares the same isovalue 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.

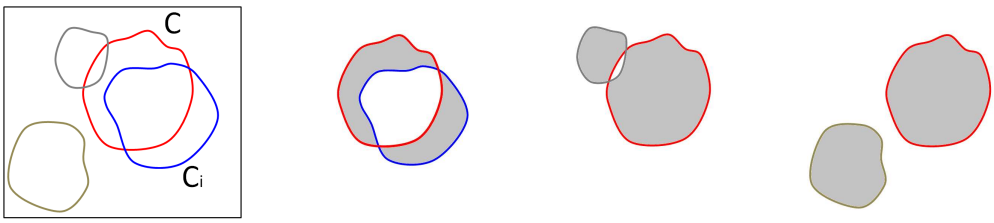
Figure 7(a) shows three contours (in blue, gray, and brown) in ensemble i with different correspondence degrees with contour C (in red). With the largest overlap degree with C, the blue contour is identified as the corresponding contour of C in ensemble member i. It is possible that no contour in C_i fulfills the correspondence criteria discussed in [27]. For example, in ensemble member i, no contour with the same isovalue of C exists or all the contours of the isovalue are apart from C. In these cases, we consider no matched contour found for C in ensemble member i.

Accordingly, the nonoverlapped area $A(C, C_i)$ between the two corresponding contours C and C_i decides the difference between the two contours. Figures 7(b), 7(c), and 7(d) illustrate the nonoverlapping area in different cases. The nonoverlapping area is computed by using B-spline function [9, 27] for each simplice of a scalar field defined on a simplicial mesh. To reduce bias toward long or short contours, larger or smaller isosurfaces, we normalize the nonoverlapping area with the contour length (or isosurface area in 3D case) of C: $difference(C, C_i) = A(C, C_i) \backslash size(C)$, where $size(C)$ is the contour length (or isosurface area in 3D case) of C.

5.2.2 Metrics and Visualization

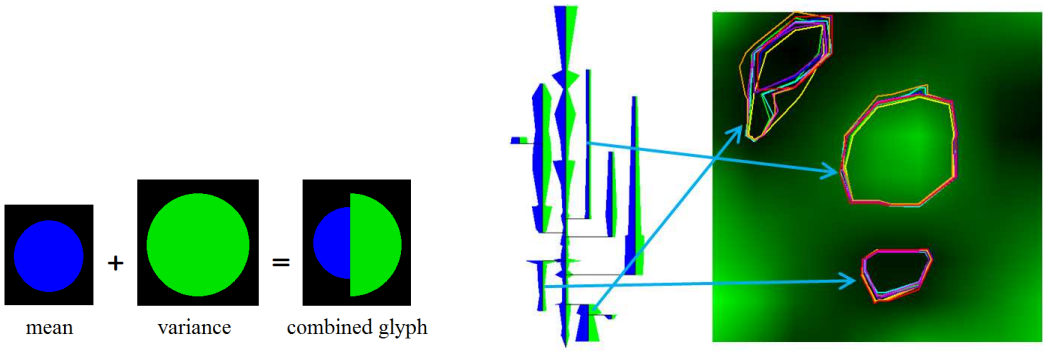
To help a user select contours according to the quantified contour variability information, we calculate the mean and variance of the differences to the mean. Given a contour C in the ensemble mean, let its corresponding contours of k individual ensemble members be $C_i(i = 1, ...k)$. C_i is the contour with the same isovalue that is matched with C. The average difference among the corresponding contours is $mean = \sum difference(C, C_j) \backslash k$. The variance of difference among contours is $va = \sum (difference(C, C_j) - mean)^2/(k - 1)$. 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 nonoverlapping area found between C and all the matched contours C_i.

The contour variability along a contour can be shown at the corresponding location in the contour tree. Figure 8 illustrates the glyph design of encoding the contour variability. Before visualization, both variability statistics are normalized to a range between 0 and a unit width. Since a ribbonlike glyph is preferred over a circular glyph to prevent



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

FIG. 7: Contour differences measured by nonoverlapped 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) Nonoverlapped areas (filled with gray) of different contours.



(a) (b)

FIG. 8: Visualization of contour variability based on contour tree. (a) Circular glyph for contour variability is produced by combining two circular glyphs for the mean and variance of differences between corresponding contours in different ensemble members and the ensemble mean. (b) Ribbons attached to the contour tree (left) indicate the variability 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.

visual cluttering, as discussed in Section 5.1, we resample the varying contour variability along each branch and use linear interpolation to produce a ribbonlike glyph 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 variability measurement for the contours along the branch.

One can click on a location in the contour tree to display a bunch of corresponding contours (spaghetti plots [4]) with certain variability levels in a data image. This saves users time and effort searching the whole image for a contour with a specified level of variability. An example is shown in Fig. 8(b). Ribbons are attached to a contour tree. Three sets of contours with different levels of variability are shown after clicking on three locations in the tree.

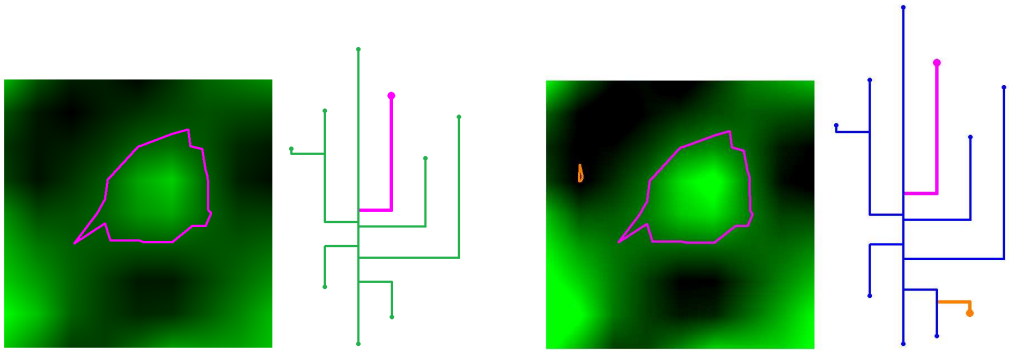
5.3 Topology Variability

The uncertainty within the data impacts 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 topological variation among the ensemble members provides new perspective on the global impact of uncertainty. We propose a visualization of topology variability based on contour trees.

In this paper, the topology variability is expressed 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 the x axis indicates their correspondence, while the disagreements between the branches on the y axis indicate their discrepancy in isovalue. A branch of the 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.

5.3.1 Contour Region of a Branch

We define the *contour region* of a branch as the region covered by all the contours within the subtree of the branch. Figure 9 shows two ensemble members of an uncertain scalar field. A comparison of their contour trees reveals that their isovalue ranges in different contour regions are different since their branches have different lengths and heights on the y axis. The contour regions for the purple branches are the regions inside the purple contours. In Figs. 9(a) and 9(b), the isovalue ranges of the contour regions are different. The latter is higher and larger. This is clearly reflected



(a) (b)

FIG. 9: 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 isovalue range of the region inside the purple contour. The small orange branch in the lower right corner of (b) is an uncertain branch whose corresponding contours are missing in (a).

in the different y axis locations and lengths of their corresponding branches. An extra branch is found in the lower right corner of Fig. 9(b). It indicates an uncertain contour region (inside the orange contour) which does not exist in the ensemble member in Fig. 9(a).

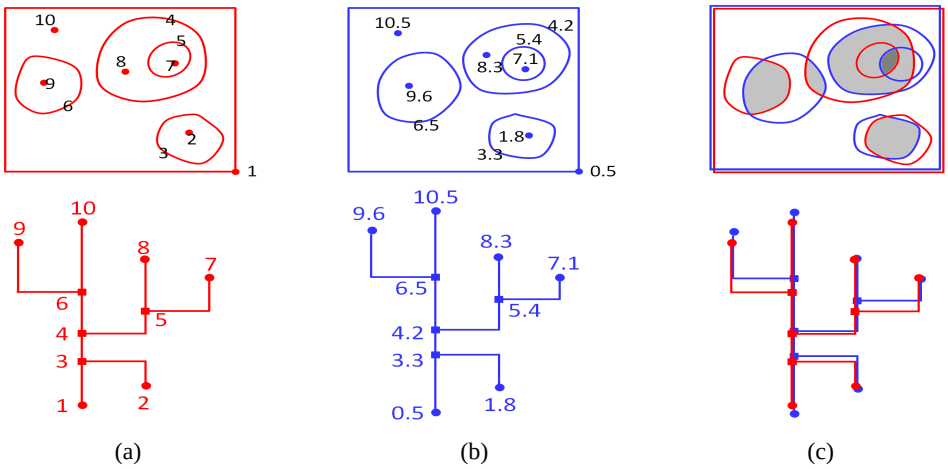
The contour region of a child branch is included in the contour region of its parent branch. Fig. 10(a) shows tree branches and their corresponding contour regions. The nodes and contours are numbered by their isovalues. The contour region of a branch is bounded by the contour of its root and with all the nodes of its subtree 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)].

5.3.2 Branch Correspondence

Spatial overlap is the most common similarity measure between features [26, 27]. 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 topologically different and need to be differentiated.

Figure 10 illustrates the branch correspondences detected according to the spatial overlaps of contour regions. Figures 10(a) and 10(b) show contours and contour trees of two scalar fields sharing the same spatial domain. In Fig. 10(a), the subtree of the middle branch (1, 10) is the whole contour tree. The subtree of branch (0.8, 10.5) is the whole contour tree in Fig. 10(b). Both branches share the whole data domain and hence are the best matched branches of each other. For the rest of the branches, the correspondence indicated by overlaps between their contour regions: branch (2, 3) $\rightarrow$ branch (1.8, 3.3), branch (6, 9) $\rightarrow$ branch (6.5, 9.6), branch (4, 8) $\rightarrow$ branch (4.2, 8.3), and branch (5, 7) $\rightarrow$ 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 a larger overlap with it since branch (4, 8) is mapped to its parent branch (4.2, 8.3). Figure 10(c) shows the matched branch in the same *x*-axis location.



(a) (b) (c)

FIG. 10: Branch correspondence indicated by contour region overlaps. The nodes and contours are numbered by their isovalues. (a) and (c) Contour trees of two data fields. (c) Corresponding contour regions of the tree branches.

5.3.3 Topology Variability 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 is a contour tree of ensemble member i. Let the branch number of CT be n and let the branch number of 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 in 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.

As shown in Fig. 11, the mapped contour trees indicate how uncertain the isovalue 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 isovalue ranges of different contour regions segmented by branches. The overall blurring

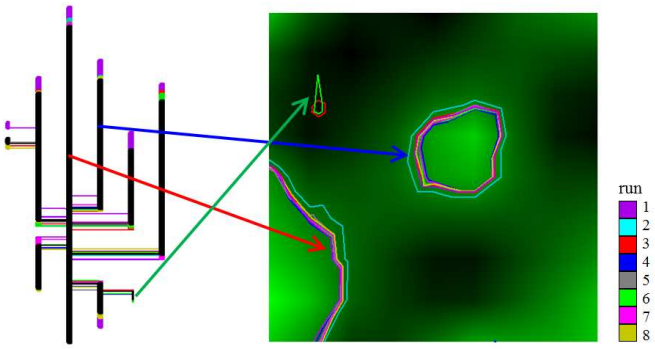


FIG. 11: Visualization of topology variability. Three sets of contours are shown after clicking on three locations (indicated by arrows) in the contour tree.

or clearness of the display indicates the overall high or low variability. 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 Fig. 11, only two contours (in red and green) from ensemble members 3 and 6 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 variability provides users a quick and high-level overview of how much variability there is in the topology of contours.

6. USER INTERFACE DESIGN

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

6.1 Interface Components and Layouts

Figure 12 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 isolines and color-mapped image for 2D data, or isosurfaces 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 mean data field. It allows interactive contour tree simplification and contour selection. The three bottom areas show the data-level uncertainty, contour variability, and topology variability of the data based on contour trees. They allow similar interactions of 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 precomputed uncertainty and variability information, the tool shows the color-mapped data (volume rendered for 3D data) in the data display area. The contours (isolines or isosurfaces) are also rendered in the data display area and updated accordingly as a user clicks on one contour tree or changes the isovalue by sliding on the vertical bar in the top-right contour tree display area.

6.2 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 stored in the precomputed simplification sequence. (2) Directly right click on the inner nodes of the contour tree to prune or extend subtrees. In practice, a user may first slide through the

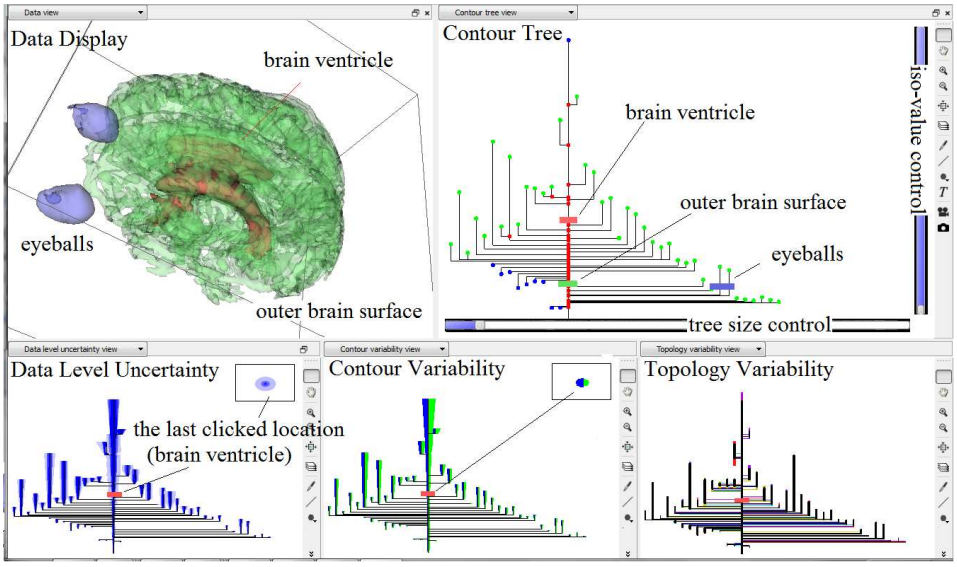


FIG. 12: User interface. Top-left: data display area shows isosurfaces chosen from the simplified mean contour tree (of the mean field of five brains) on the top right. Top-right: contour tree display area shows a simplified contour tree. Bottom: display areas for data-level uncertainty, contour variability, and topology variability shown on simplified contour trees.

horizontal bar to get a certain simplification level; then, work through the tree view graph to show or hide subtrees according to the uncertainty information attached.

6.3 Contour Selection

Users may select contours to display in three ways. (1) Display contours of the mean field with selected isovalues by clicking on the contour tree of the top-right area. A selected contour is shown in the same distinct color with a point placed at the clicked location (see Fig. 12). (2) Display a set of corresponding contours with the same isovalue in different ensemble members by sliding on the vertical bar in the top-right display area. The value of the slider corresponds to the selected isovalue. The contours are chosen at the points on the current contour tree branches with the selected isovalue. (3) Display a set of corresponding contours with the same isovalue in different ensemble members in the data display area by double clicking on a location in one of the three contour trees in the bottom. Contours from different ensemble members are shown in different colors.

For 3D data, users may choose to show isosurfaces with or without transparency. Both data-level uncertainty and contour variability for the selected contour are shown with circular glyphs after each click on a contour tree in any of the four display areas (the top-right display area or the three display areas in the bottom).

7. APPLICATIONS AND RESULTS

In this section, we demonstrate the effectiveness of the new uncertainty visualization by applying it to a simulated weather data set ($135 \times 135 \times 30$) and a medical volumetric data set ($128 \times 128 \times 71$). The contour trees, branch hierarchies, uncertainty,and variability information are pre-computed.

The first application is to visualize nonweighted diffusion images. The results are shown in Fig. 13. In this application, we apply our method to study the difference between the brain images from five subjects after affine registration to one particular data set chosen at random using FLIRT [34]. Contour trees had been introduced to explore brain data in previous literature [10]. The between-subject variation in brain anatomy is closely related to the uncertainty study of brain imaging [35]. The mean field of the brains contains 2143 critical points and 1071 branches in the original

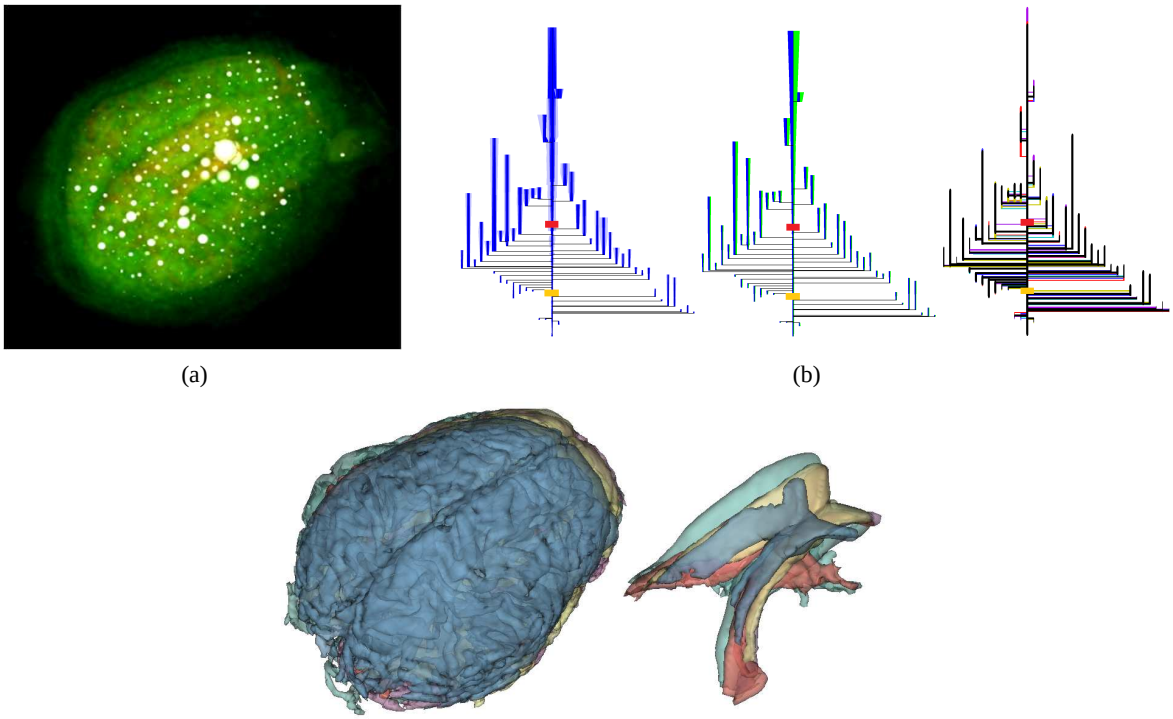


FIG. 13: Exploration of five brain data sets with contour tree based visualization. (a) Volume rendering of the mean field of the five brains with circular glyphs for data-level uncertainty. (b) Left to right: data-level uncertainty, contour variability, and topology variability shown in a simplified contour tree of the mean brain data field. (c) Five overlapped brain outer surfaces corresponding to the yellow points in the contour trees and five overlapped ventricle surfaces corresponding to the red points in the contour trees.

contour tree. Figure 13(a) shows the volume rendering of the average data with circular uncertainty glyphs whose sizes vary according to the level of data-level uncertainty. Uncertainty and variability information is shown in the simplified contour tree in Fig. 13(b). Brain outer surfaces and ventricle surfaces are selected from two points (indicated in red and yellow) in the contour trees. As shown in Fig. 13(a), the inner part of the brain area exhibits higher data value (indicated by more yellow color) along with higher data-level uncertainty (indicated by bigger circular glyphs). This is reflected in the generally thicker graduated ribbons observed in the upper region of the left tree in Fig. 13(b). Meanwhile, contours corresponding to the upper region of the trees have higher contour variability and higher topology variability, indicating the impact of the data-level uncertainty on the contours and contour trees. The inner features such as brain ventricles exhibit higher variability than the brain outer surfaces based on the quantified uncertainty and variability information shown on the corresponding locations in the contour trees. Our interactive visualization supports the exploration of uncertainty and variability information that is hidden in the conventional view for selected anatomical structures.

The second experiment demonstrates an effective application of the new uncertainty visualization on the simulated data from Weather Research and Forecasting (WRF) model runs. The members of numerical weather prediction ensembles are individual simulations with either slightly perturbed initial conditions or different model parametrizations. Scientists use the average ensemble output as a forecast and utilize spaghetti plots to analyze the spread of the ensemble members. In our application, water-vapor mixing ratio data from eight simulation runs of the 1993 “Super storm” are used. Figure 14(a) shows the original contour tree (1734 critical points and 867 branches) and all the contours with the isovalue indicated by the red horizontal line connecting the slider of the vertical bar. Figure 14(b) shows

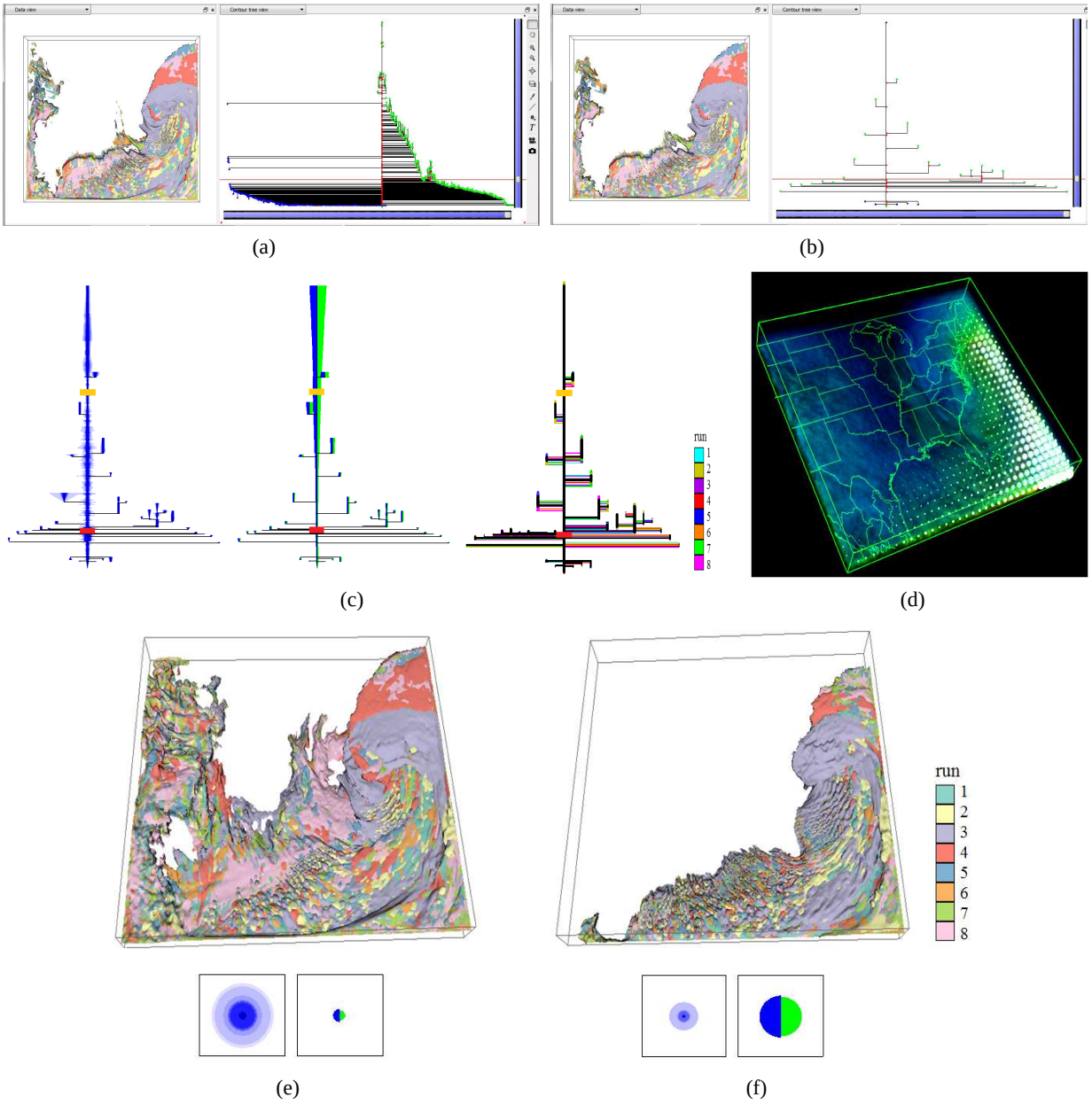


FIG. 14: Exploration of weather ensemble simulation with contour tree based visualization. (a) Original contour tree. (b) Manually simplified contour tree with tree-view glyph interaction. (c) Left to right: data-level uncertainty, contour variability, and topology variability shown in the simplified contour tree. (d) Volume rendering with uncertainty glyphs. (e) Set of contours (corresponding to the red points in the contour trees) with high data-level uncertainty indicated by the large graduated circle. (f) Set of contours (corresponding to the yellow points in the contour trees) with high contour variability indicated by the large glyph for contour variability.

a manually simplified contour tree (66 critical points and 33 branches) and a few contours which correspond to the points on the current branches crossed by the red line. Uncertainty and variability information is shown in the simplified contour tree in Fig. 14(c). The right contour tree in Fig. 14(c) shows the topology variability 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 14(d) shows the integrated visualization of both data and uncertainty glyphs. Figure 14(e) shows a set of corresponding contours with high data-level uncertainty while Fig. 14(f) shows a set of corresponding contours with high contour variability. Circular glyphs indicating the magnitude of the data-level uncertainty and contour variability are shown at the bottom. The color bar on the right indicates the correspondence between the isosurfaces and simulation runs. Our interactive visualization allows users to explore data hierarchically with quantified uncertainty and variability information as the clue to where to look.

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 different uncertainty or variability information 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 uncertainty or variability in a planar contour tree layout helps users investigate the contours with certain uncertainty or variability more precisely. Volume renderings with circular uncertainty glyphs [Fig. 13(a) and Fig. 14(d)] 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.

8. CONCLUSION AND FUTURE WORK

We present a contour tree based visualization for exploring data with uncertainty. One contribution of the paper 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 paper is a full exploration of uncertainties in scientific data with the data-level uncertainty, contour variability, and topology variability based on the contour tree. 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 new 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 uncertainty and variability and apply our methods to address different types of uncertainty models. In contour tree simplification methods [7, 8, 29, 36], 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 [36]. 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 training. Our interactive visualization tool may be further improved based on user feedback.

ACKNOWLEDGMENTS

This work has been supported in part by NSF1117871. The authors wish to thank Ruiyi Wu for implementing the brain registration and Jibonananda Sanyal for providing the simulated weather data and helping us to design the interface with Qt.

REFERENCES

1. Pang, A., Wittenbrink, C., and Lodha, S., Approaches to uncertainty visualization, *Visual Comput.*, 13(8):370–390, 1997.
2. Wittenbrink, C., Pang, A., and Lodha, S., Glyphs for visualizing uncertainty in vector fields, *IEEE Trans. Visualization Comput. Graphics*, 2(3):226–279, 1996.

3. Djurcilov, S., Kim, K., Lermusiaux, P. F. J., and Pang, A., Volume rendering data with uncertainty information, In *Proceedings of the Joint EUROGRAPHICS—IEEE TCVG Symposium on Visualization*, Springer-Verlag, Switzerland, 2001.
4. Sanya, J., Zhang, S., Dyer, J., Mercer, A., Amburn, P., and Moorhead, R. J., Noodles: A tool for visualization of numerical weather model ensemble uncertainty, *IEEE Trans. Visualization Comput. Graphics*, 16(6):1421–1430, 2010.
5. Grigoryan, G. and Rheingans, P., Point-based probabilistic surfaces to show surface uncertainty, *IEEE Trans. Visualization Comput. Graphics*, 10(5):564–573, 2004.
6. Diggle, P., Heagerty, P., Liang, K.-Y., and Zeger, S., *Analysis of Longitudinal Data*, Oxford, Oxford University Press, 2002.
7. Pascucci, V., Cole-McLaughlin, K., and Scorzell, G., Multi–resolution computation and presentation of contour tree, In *the IASTED Conference on Visualization, Imaging, and Image*, ACTA, Marbella, Spain, pp. 452–290, 2004.
8. Carr, H., Snoeyink, J., and van de Panne, M., Simplifying flexible isosurfaces using local geometric measures, In *Visualization, IEEE*, pp. 497–504, 2004.
9. Bajaj, C. L., Pascucci, V., and Schikore, D. R., The contour spectrum, In *Visualization, Proceedings*, pp. 167–173, 1997.
10. Carr, H. and Snoeyink, J., Path seeds and flexible isosurfaces using topology for exploratory visualization, In *Proceedings of the Symposium on Data Visualisation*, Eurographics Association Aire-la-Ville, Switzerland, 2003.
11. Weber, G., Bremer, P.-T., and Pascucci, V., Topological landscapes: A terrain metaphor for scientific data, *IEEE Trans. Visualization Comput. Graphics*, 13(6):1416–1423, 2007.
12. Heine, C., Schneider, D., Carr, H., and Scheuermann, G., Drawing contour trees in the plane, *IEEE Trans. Visualization Comput. Graphics*, 17(11):1599–1611, 2011.
13. Davis, T. J. and Keller, P., Modeling and visualizing multiple spatial uncertainties, *Comput. Geosci.*, 23(4):397–408, 1997.
14. Djurcilova, S., Kima, K., Lermusiauxb, P., and Pang, A., Visualizing scalar volumetric data with uncertainty, *Comput. Graphics*, 26(2):239–248, 2002.
15. Sanyal, J., Zhang, S., Bhattacharya, G., Amburn, P., and Moorhead, R., A user study to compare four uncertainty visualization methods for 1d and 2d datasets, *IEEE Trans. Visualization Comput. Graphics*, 15(6):1209–1218, 2009.
16. Rhodes, P. J., Laramee, R. S., Bergeron, D., and Sparr, T. M., Uncertainty visualization methods in isosurface rendering, In *Eurographics*, 2003.
17. Pfaffelmoser, T., Reitinger, M., and Westermann, R., Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields, In *Eurographics / IEEE Symposium on Visualization*, 2011.
18. Pauly, M., Mitra, N. J., and Guibas, L., Uncertainty and variability in point cloud surface data, In *Eurographics Symposium Point-Based Graphics*, 2004.
19. Pthkow, K. and Hege, H.-C., Positional uncertainty of isocontours: Condition analysis and probabilistic measures, *IEEE Trans. Visualization Comput. Graphics*, 17(10):1393–1406, 2011.
20. Pthkow, K., Weber, B., and Hege, H.-C., Probabilistic marching cubes, *Comput. Graphics Forum*, 30(3):931–940, 2011.
21. Lorensen, W. E. and E. Cline, H., Marching cubes: A high resolution 3D surface construction algorithm, In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, 1987.
22. van Kreveld, M., van Oostrum, R., Bajaj, C., Pascucci, V., and Schikore, D., Contour trees and small seed sets for isosurface traversal, *Proceedings of the 13th Annual Symposium on Computational Geometry*, 1997.
23. Fujishiro, I., Azumay, T., and Takeshimay, Y., Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis, In *Visualization, Proceedings*, pp. 467–563, 1999.
24. Chiang, Y.-J., Lenz, T., Lu, X., and Rote, G., Simple and optimal output-sensitive construction of contour trees using monotone paths, *Comput. Geom. Theory Appl.*, 30(2):165–195, 2005.
25. Carr, H., Snoeyink, J., and Axen, U., Computing contour trees in all dimensions, *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.
26. Schneider, D., Wiebel, A., Carr, H., Hlawitschka, M., and Scheuermann, G., Interactive comparison of scalar fields based on largest contours with applications to flow visualization, *IEEE Trans. Visualization Comput. Graphics*, 14(6):1475–1482, 2008.
27. Sohn, B.-S. and Bajaj, C., Time-varying contour topology, *IEEE Trans. Visualization Comput. Graphics*, 12(1):14–25, 2006.
28. Takahashi, S., Ikeda, T., Shinagawa, Y., Kunii, T. L., and Ueda, M., Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data, *Comput. Graphics Forum*, 14(3):181–192, 1995.

29. Takahashi, S., Takeshima, Y., and Fujishiro, I., Topological volume skeletonization and its application to transfer function design, *Graph. Models*, 66(1):24–49, 2004.

30. Takahash, S., Takeshima, Y., Nielson, G. M., and Fujishiro, I., Topological volume skeletonization using adaptive tetrahedralization, In *Geometric Modeling and Processing, Proceedings*, pp. 227–236, 2004.

31. Carr, H., *Topological Manipulation of Isosurfaces*, PhD Thesis, 2004.

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

33. Microsoft, Tree views, http://msdn.microsoft.com/en-us/library/windows/desktop/aa511496.aspxe/, 2011.

34. Jenkinson, M. and Smith, S., A global optimisation method for robust affine registration of brain images, *Med. Image Anal.*, 5(2):143–156, 2001.

35. Eickhoff, S. B., Laird, A. R., Grefkes, C., Wang, L. E., Zilles, K., and Fox, P. T., Coordinate-based activation likelihood estimation meta-analysis of neuroimaging data: A random-effects approach based on empirical estimates of spatial uncertainty, *Hum. Brain Map.*, 30(9):2907–2926, 2009.

36. Bremer, P.-T., *Topology–Based Multi–Resolution Hierarchies*, PhD Thesis, 2004.