Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization

Dominic Schneider, Alexander Wiebel, Hamish Carr, *Member, IEEE*,
Mario Hlawitschka and Gerik Scheuermann, *Member, IEEE CS*

Abstract— Understanding fluid flow data, especially vortices, is still a challenging task. Sophisticated visualization tools help to gain insight. In this paper, we present a novel approach for the interactive comparison of scalar fields using isosurfaces, and its application to fluid flow datasets. Features in two scalar fields are defined by largest contour segmentation after topological simplification. These features are matched using a volumetric similarity measure based on spatial overlap of individual features. The relationships defined by this similarity measure are ranked and presented in a thumbnail gallery of feature pairs and a graph representation showing all relationships between individual contours. Additionally, linked views of the contour trees are provided to ease navigation. The main render view shows the selected features overlapping each other. Thus, by displaying individual features and their relationships in a structured fashion, we enable exploratory visualization of correlations between similar structures in two scalar fields. We demonstrate the utility of our approach by applying it to a number of complex fluid flow datasets, where the emphasis is put on the comparison of vortex related scalar quantities.

Index Terms—Scalar topology, comparative visualization, contour tree, largest contours, flow visualization.

1 Introduction

A predominant technique for segmenting and analysing three-dimensional scalar fields are isosurfaces. Their position and shape provide clues to the underlying structure of the scalar field. In the application area considered in this paper, namely the analysis and visualization of flow data, isosurfaces are a common tool for visualizing flow features such as vortices or shock fronts amongst other features. The surfaces are extracted for physical quantities like pressure, density and temperature, as well as for derived quantities like vorticity, divergence, scalar variables resulting from feature detection criteria like λ_2 [15], or measures of flow variation over time [33]. The comparison of the isosurfaces and thus of the different variables can confirm and yield insights into the interaction and relation of different flow related quantities. For this purpose we implemented an interface for scalar field comparison.

Of special interest for flow visualization is the reliable detection and visualization of vortical structures. The method considered to be state of the art is the λ_2 algorithm. The resulting scalar field is usually visualized using isosurfaces (see Figure 1 (a)-(c)). The isosurface extracted for zero isovalue encloses all swirling motion (see Figure 1(a)). Unfortunately, this segmentation makes analysis relatively difficult as vortices are not automatically separated. As we will demonstrate throughout this paper, contour trees are helpful to solve this problem.

In scalar field visualization, the contour tree concept and data structure plays a central role. It captures the topological evolution of isosurfaces for varying isovalue. Thus, the contour tree naturally provides a segmentation of a scalar field. In this work we want to make use of this segmentation property of the contour tree to detect and compare features in fluid flow related scalar fields. To the best of our knowledge contour trees have not been used in flow visualization so far. We use them to generate a largest contour segmentation, to find spatially overlapping features in the segmentations of two fields and for linked contour tree views. In Section 4.2, we introduce the *similarity browser* as part of an interface to easily explore (spatial) conjointly appearing features. This is achieved by providing preview renderings of related feature pairs which are displayed in descending order of similarity to display the strongest relationships first. The relationship itself is established by means of spatial overlap. Additionally, a graph view is provided in the interface which illustrates all identified relationships using information visualization techniques.

In Section 4.3 we describe the interface for interactive user-directed comparison of the scalar fields based on selecting nodes in linked contour tree views. Selecting a certain node in a view causes the connected subtree to be highlighted. All affected largest contours in the second contour tree are automatically extracted and highlighted.

We dicuss a wide range and variety of application datasets from the fluid flow domain in Section 6. Special emphasis is put on the comparison and analysis of vortices and vortex related quantities

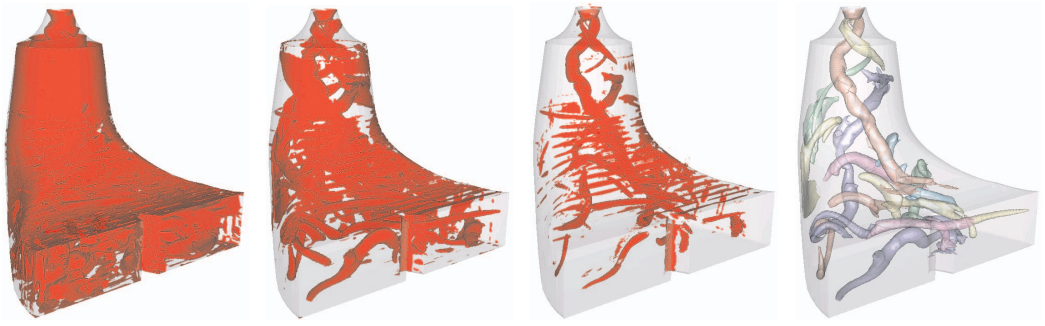
- *Dominic Schneider, Alexander Wiebel, Mario Hlawitschka and Gerik Scheuermann are with the University of Leipzig, E-mail: {schneider, wiebel, hlawitschka, scheuermann}@informatik.uni-leipzig.de.*
- *Hamish Carr is with University College Dublin, E-mail: hamish.carr@ucd.ie.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Problem Statement

As already mentioned, the detection of different vortices is of great interest for flow analysis but the natural segmentation provided by the popular λ_2 criterion does not always separate different vortices. From Figure 1(a) it can be seen that isosurface visualizations using the standard isovalue zero suffer from visual clutter and occlusion rendering analysis or even comparison practically infeasible. A separation of different vortices is possible by using negative isovalues with larger absolute value. Still, standard isosurface techniques fail to segment all different vortices because they may merge at different isovalues. In the following we consider four datasets to show the practical relevance of the problem stated above.

The first dataset models the flow in a part of a Francis turbine. The considered part is the draft tube where the water coming from the rotor leaves the turbine. The data is obtained by an incompressible computational fluid dynamics (CFD) simulation. The water runs through the tube from the upper part of the images in Figure 1 and leaves this part of the tube at a divider slightly visible in the lower part of the images. Images 1(a) to 1(c) show complete isosurfaces of λ_2 for different isovalues. Image 1(a) illustrates the problem stated above as it shows the isosurface for zero isovalue yielding a completely useless image because of strong occlusion and clutter. While clutter and occlusion are reduced in images 1(b) and 1(c) by decreasing the isovalue, some of the features disappear. The largest contour segmentation in image 1(d) reveals *all* relevant structures while being less cluttered and easier to understand than the other images.



(a) λ_2 isosurface for 0.0 (b) λ_2 isosurface for -1000 (c) λ_2 isosurface for -5000 (d) λ_2 with largest contours

Fig. 1. (a)-(c) λ_2-isosurfaces for different isovalues. Lower isovalues reduce clutter and isolate different features but eliminate certain features. (d) Largest contour path seed approach reveals all features.

The cuboid dataset, our second example, is a direct numerical simulation of unsteady incompressible flow around a cuboid. The simulation was carried out with the NaSt3DGP[1] flow solver. The velocity and pressure data is stored on a $100 \times 100 \times 100$ rectilinear grid. We use one out of 1355 available time steps. The sharp edges of the cuboid induce vortices which separate from the body. The vortices are stretched into hairpin shape as they travel downstream. Now it is known that λ_2 and pressure exhibit small values in regions of vortical behaviour. In order to analyse such a relationship in detail, single vortices must be extracted along with the according pressure minimum or minima, respectively.

Our third dataset stems from a CFD simulation of the flow around a delta wing configuration. The most prominent features of the flow are two large vortices above the wing which strongly contribute to its lift. The angle of the flow approaching the wing increases over time. The inclination of the two vortices changes with the changing on-flow. At some point the angle of the flow becomes critical and causes the vortices to *burst* and develop so called vortex breakdown bubbles. It has been shown that helicity and stagnation points play an important role in the creation of the bubbles [8]. We pick a time step with fully developed bubbles for investigation and analysis. Now, if the bubbles and their creation are to be analysed further, their location and shape need to be extracted. It is understood that more than one quantity is involved in the creation of the bubbles, therefore at least two quantities and their relationship need to be studied at once.

NASA's well known blunt fin dataset [12] is our fourth example. The image in Figure 7 shows only the left part of the fin because of the symmetry assumed for the simulation. The flow approaches the fin from the right side of the image and exhibits several vortices and a shock branching in front of the fin. From the fluid mechanics literature it is known that a shock can be described by a high gradient in density and pressure [27]. Here, this relationship can easily be verified by using standard isosurface techniques but an automatic extraction would be more convenient.

3 Related Work

Relevant prior work to this paper includes work on visualizing two scalar fields simultaneously [35], visualizing correlations between multiple fields [21], information visualization of similarities between multiple fields [1], feature identification and tracking [22, 16], topological analysis and display of data [6], and topological tracking [23].

The simplest method for multi-variate visualization is to juxtapose visualizations of individual properties, relying on linked views[34] to impose sufficient visual structure for the human user to detect important relationships between the properties. More advanced methods superimpose two visualizations, define visual properties using derived properties [35], map individual properties to different visual channels [17] or even different techniques [18], or allow visual queries of individual properties [9].

Such methods, however, tend to bog down because of occlusion and visual clutter caused by the sheer complexity of engineering simulations in particular, and more recent work has looked at global correlations between fields [21] to extract significant relationships. However, not all relationships are global - many are instead small-scale correlations. Other researchers [14] have looked at local relationships using well-established notions of neighbourhoods, but these neighbourhoods tend to be of fixed size and shape.

3.1 Vortex Detection

We will discuss our comparison approach on some vortex related quantities in Section 6. There is a large number of such quantities. This may result from the lack of a precise mathematical definition of a vortex [19] although vortices are important features and thus some kind of description is needed. The simplest and most important quantity related to vortices is vorticity, i.e. the curl of the velocity field. As vorticity describes the rotational behaviour of a fluid, vortices usually correlate with regions of high vorticity. However, exceptions like shear flow exist. A second indicator of vortices is locally low pressure. Some deficiencies of low pressure as vortex criterion led to a kind of modified pressure criterion known as the λ_2-criterion introduced by Jeong an Hussain [15]. There, negativity of two eigenvalues of a matrix related to the pressure Hessian indicates the existence of vortices. The matrix is obtained from the velocity gradient tensor and the negativity of the eigenvalues is usually ensured by searching for negative values of the middle eigenvalue, i.e. λ_2. All these values indicate vortex regions in contrast to vortex core lines, the centre lines of vortices, as extracted by Sujudi and Haimes [25] and subsequent work.

Most similar to the approach described in this work is the exploratory tool to study vortical flows described by Stegmaier et al. [24]. It uses the Banks-Singer predictor-corrector algorithm in combination with the λ_2 field to detect the core line of a vortex using λ_2 minima which are smaller than zero as seed points. Vortex boundaries are detected by radially sending out rays in the plane perpendicular to the vortex core line in every step and sampling the λ_2 field along the rays until the user-defined λ_2 isovalue is exceeded. Using the resulting points, a surface enclosing the core of the vortex is constructed. In contrast, the approach described in this work is more general since it considers the whole volume by means of the contour tree. Features are defined as largest contours, consequently vortices are minima in the λ_2 field, which are separated automatically without manual interaction or thresholds. Since the optimal isovalue for every vortex is determined separately, this is a more natural separation criterion than artifically designated thresholds. Contour tree simplification accounts for vortices which may consist of smaller connected minima regions.

[1] NaSt3DGP was developed by the research group in the Division of Scientific Computing and Numerical Simulation at the University of Bonn. It is essentially based on the code described in a book by Griebel *et al.* [10]. A version of the NaSt3DGP code, as well as related information and documentation is available for download at `http://wissrech.iam.uni-bonn.de/research/projects/NaSt3DGP/index.htm`.

3.2 Contour Trees

Contour trees are abstractions representing the nesting relationship of all possible isovalued contours in a field [3], and have been used as abstract representations of the data [2], to store or extract individual contours [28, 13, 6], to simplify the topology of the field [7], and for direct comparison of fields by graph matching [11, 36].

Conceptually, contour trees result from shrinking all of the contours in a function to single points, leaving only a topological skeleton of the function which has a 1-1 correspondence between points in the tree and entire contours in the function. Formally, the vertices or *supernodes* of the contour tree represent contours passing through *critical points* of the function at which the connectivity of level sets change, while the edges or *superarcs* of the contour tree represent continuously deforming sets of contours with unchanging connectivity at varying isovalues. Moreover, the local extrema of the function appear as leaf vertices of the contour tree, while saddle points appear as interior vertices providing that the connectivity, as opposed to the genus, of the contours, changes.

The *flexible isosurface* interface [6] exploits this 1-1 correspondence by showing that individual contours can be represented and manipulated as single points in the contour tree, and by allowing the manipulation of sets of contours which need not share a common isovalue. As a corollary, this also brought together prior work in feature tracking [22] and segmentation [20] by showing that the *largest contour* segmentation corresponded to contours placed near the interior (saddle) end of superarcs incident to extrema. Thus, the largest contour segmentation and feature tracking definitions of features are a special case of the flexible isosurface.

Subsequent work [7] refined the flexible isosurface by demonstrating that geometrically minor topological features can be removed by graph simplifications of the contour tree. An important property of this simplification method is that it is based on *local geometric measures* - geometric properties such as surface area or volume of individual contours. By suppressing topological features of small geometric volume, this permits direct interaction with larger scale topological features of the data. Correspondingly, this permits both largest contour segmentation and feature tracking approaches to ignore small scale features in favour of larger features, a property which we will exploit later. Additional work then extended the flexible isosurface approach to volume rendering by localising transfer functions to individual superarcs and the corresponding contours [26, 30].

In either case, the ability to render single features with varying isovalues helps address all of the problems noted above - locality, regularity of neighbourhoods, visual clutter and occlusion. Moreover, the ability to perform topological simplification allows the definition of feature to be varied to suppress noise and small features.

4 The Interface

As mentioned above, our new interface consists of three parts: the *similarity browser*, two linked contour tree views and a main render view. In the following subsections, we describe the first two parts of our interface and their connection to the render view. For the *similarity browser*, which consists of a thumbnail gallery showing feature pairs and a graph view illustrating their relationship we need to be able to extract the features. Therefore and for the comparison in general, we first give our definition of the terms feature and similarity of features.

4.1 Feature Definition and Feature Similarity

In general, we follow the argument of Silver et al. [22] that features can be defined by local extrema. A more specific definition was given by Manders et al. [20] who defined features through a largest contour segmentation obtained by region growing. In such a segmentation a feature is defined by the largest contour around an extremum which does not contain another critical point. Unfortunately, both approaches lack a well-founded topological algorithm for finding the threshold value of the feature boundary. Therefore, we define a feature as a leaf and its adjacent superarc in a simplified contour tree which naturally corresponds to a largest contour and its "interior".

We obtain the information about largest contours from the fully augmented contour tree which has been simplified using a local geometric measure [7]. Any geometric measure could be used for simplification, but since the similarity measure described later in this section is based on volumetric overlap we decided to use volume as simplification measure. For simplification two thresholds must be set manually, one for each scalar field. The threshold as such can be interpreted as a lower bound for the size of the extracted features. Since the fully augmented contour tree contains all vertices, a superarc between a local maximum or minimum and the respective saddle comprises of vertices belonging to the volume of the according largest contour. Since this set of vertices is associated to the largest contour, it is called *associated vertex set* in the following.

With the definition of feature we can now define the similarity of features of different scalar fields S_A and S_B. In order to compute the similarity between two given features F_A and F_B, let V_A be the associated vertex set of F_A and V_B the associated vertex set of F_B. Then the similarity measure s is defined as

$$s(V_A,V_B) = \frac{1}{2}\left(\frac{|V_A \cap V_B|}{|V_A|} + \frac{|V_A \cap V_B|}{|V_B|}\right) \quad (1)$$

where $V_A, V_B \subset R^3$ and $|\cdot|$ denotes the number of elements of a set. Note, that s is symmetric. This is necessary to ensure that the order of comparison does not matter. Moreover, s is a volumetric measure meaning that the more overlapping volume relative to their size two features have, the more similar they are. The reason for similarity being computed relative to the size of a feature is to obtain a high rank not only for features with high overlap but also for pairs of small features with similar size. During similarity computation a user defined threshold filter is applied to suppress insignificant similarity relationships.

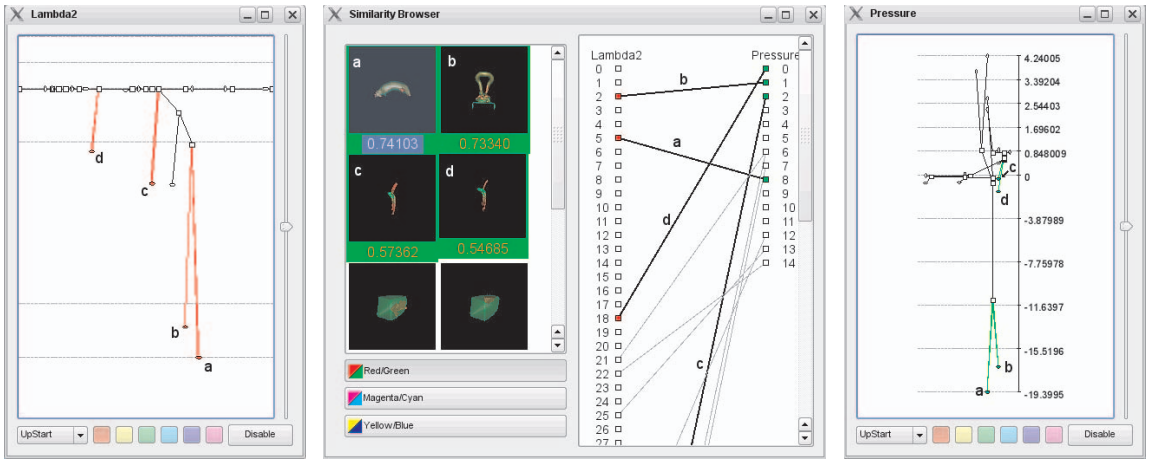
For the descriptions of the interaction with the contour trees in Section 4.3, we need to generalise the concept of a largest contour to include more than one critical point. Therefore, we define the *generalised largest contour* to be the largest contour bounding a region containing the critical points of a connected subtree of the contour tree but no other critical points. This corresponds to the largest contour for a simplified version of the contour tree where the subtree is simplified to a leaf edge.

In order to avoid any of the perceptional issues with open contours mentioned in Section 1 we close open contours via the boundary This clearly indicates which side of the drawn contour is meant by exploiting human perception of objects [29]. By closing an open contour, we are visually creating a three-dimensional object instead of an open surface. In the case of an arbitrary isovalue an additional parameter we call *direction* is needed to decide whether the upstart or downstart region of a contour is meant. Upstart and downstart have been defined by Carr et al. [7] since terms like *above* and *below* do not apply to a contour in general. Nor do *inside* and *outside* for open contours. However, an upstart region is a region only reachable from the contour by paths that initially ascend from the contour and never return to it. Once the *direction* has been fixed, terminology like inside and outside have meaning even for contours intersecting the boundary, i.e. open contours. In the special case of largest contours the choice of the direction is clear. If isosurfaces are used the parameter need to be defined manually. The advantage is immediately obvious as the created closed objects represent the parts of the data where the value is higher or lower than the specified isovalue, respectively.

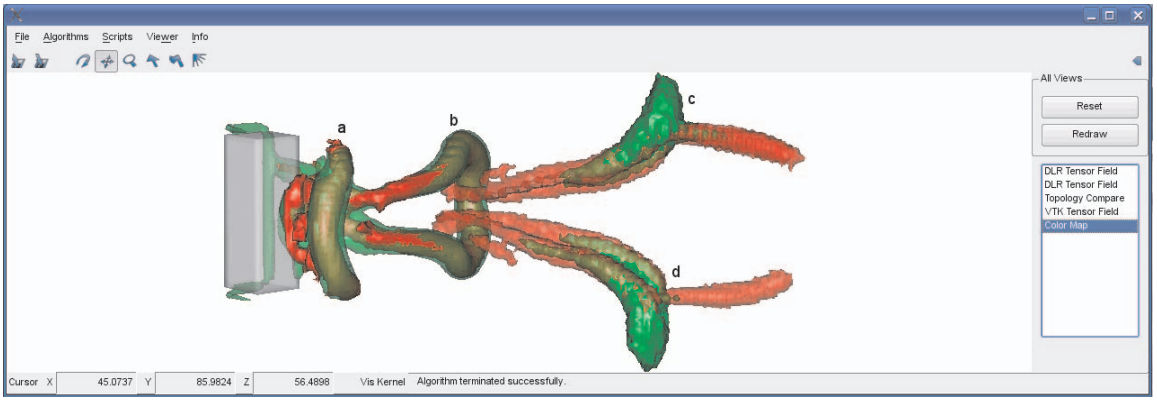
A contour is a connected component of an isosurface, but in combination with *direction* it is a bit more subtle: Due to the *direction* we have a definition of what is inside and outside. Now, with the definition of inside, contours represent individual objects not defined just by the isovalue rather than the contour tree because they may include regions above *and* below the isovalue of the contour. This is useful in applications where contours actually encode spacious objects.

4.2 Similarity Browser Window

The thumbnail gallery (see e.g. Figure 2(b) left part) shows preview images rendered for each feature pair exhibiting a significant similar-



(a) λ_2 Contour Tree (b) Similarity Browser (c) Pressure Contour Tree



(d) Main Render View

Fig. 2. Interface overview **Up:** Figure (b) shows the selected feature pairs in the *Similarity Browser* window. Figure (a) and (c) show the respective contour tree views with the according largest contours marked in each tree as highlighted in red and green. **Bottom:** Rendering of matched contours as selected in the *Similarity Browser*. The red surfaces represent contours in λ_2 field, whereas the green surfaces belong to contours in the pressure field.

ity, to allow for a quick overview of the shapes of the matched features. In the graph view (see e.g. Figure 2(b) right part) on the other hand we employed information visualization techniques to show all relationships between features. From the two disjoint sets of features a bipartite graph is constructed. Each set is sorted by volume in descending order and displayed in a column where boxes represent the individual features. Now, for each feature pair a thin, light grey edge connecting the respective features is inserted into the graph. Three different colour pairs are provided by buttons in the lower left corner of the *similarity browser* for cases where a differentiation between matches is necessary. The provided colour pairs have been chosen to be complementary colours to allow for high contrast [29]. If a thumbnail is selected, the two corresponding features are assigned colours respective to the chosen colour pair. The according edge in the bipartite graph is marked, drawing it in thick black, and the boxes representing the corresponding features in the graph are filled with the assigned colour. Since features are defined through largest contours, the two contour tree views (see Section 4.3) are updated as well, drawing the node of the extremum and the according edge in the respective colour. Moreover, features are drawn in the main render view using a modified path seed approach [6]. Since we are only extracting largest contours, the grid edge joining a connected component and the saddle vertex, as part of the according path seed, identified by the union-find data structure will suffice. The larger of the two overlapping features, i.e. the one with the higher vertex count, is drawn translucently to make features visible which might be fully or partly enclosed by the contour of the larger feature (see e.g. Figs. 7 and 2(d)).

4.3 Interactive Contour Tree Views

In addition to the powerful automatic comparison provided by the *similarity browser* described in the previous section, our interface allows the user to interactively explore relationships between features in two scalar fields through two linked contour tree views.

4.3.1 Contour Tree Interaction and Interface Elements

Depending on the *direction*, which can be chosen from a drop down box in the lower left corner (see e.g. Fig. 2(a)), clicking on a node of a contour tree causes a certain subtree to be extracted and highlighted. The according generalised largest contour is visualized in the main render view. All features (largest contours) of the second tree intersecting the contour selected in the first tree are highlighted (see Fig. 3 right) and drawn in the main render view.

Both, highlighting and drawing the contours, are carried out with a colour which can be selected by clicking on one of the buttons located in the lower middle. The colours provided comply with Ware's unique hues [29]. They are chosen to have low saturation for the first tree, where the selection is made, to be distinguishable from the colours of the automatically extracted contours of the other tree (see Fig. 3). In addition, we decided to draw the contours of the first tree translucently as the automatically extracted contours of the second tree are usually smaller and fully or partly covered. Using the same hue for contours extracted in the process of a manual selection shall communicate that the contours have some sort of relation, which, in our case, is volumetric similarity.

Apart from the described colour scheme, saddles in the contour tree

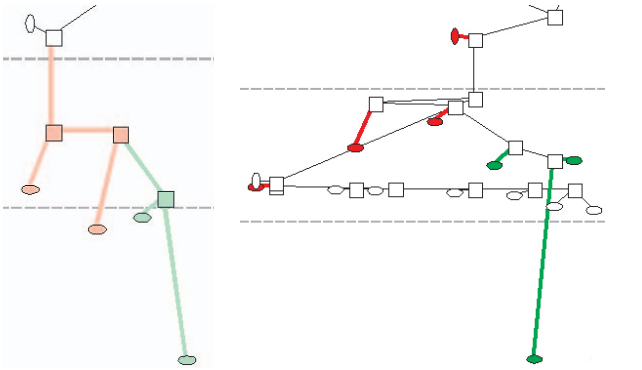


Fig. 3. Close up views of contour trees of density and pressure for blunt fin dataset. The left image shows two marked subtrees: the green and the red one. Right image shows the corresponding largest contours highlighted in the second contour tree.

are represented by boxes, minima by horizontally aligned ellipses and maxima by vertically aligned ellipses. Finally, straight lines are connecting these nodes.

The procedure for subtree extraction used for a selected node works as follows: From any manually selected node, we traverse the subtree in the chosen *direction*. Thus, we extract the subtrees connected to the node by a descending superarc if downstart is selected or subtrees connected by an ascending superarc if upstart is selected. In either case, the selected node itself and superarcs going out in the opposite *direction* are extracted as well. The latter are used to extract seeds for the surfaces displayed in the main render view (see Section 5). During the subtree extraction process all vertices of the highlighted subtrees are collected and transfered to the other contour tree. There, all largest contours, i.e. features, belonging to the transferred vertices are identified based on the method described in Section 5.1 and visualized as described above.

4.3.2 Advanced Interaction

So far, we have only described the results obtained from a click on one node. However, there are more complex types of selection possible. Consider a complete subtree to be highlighted with a certain colour (e.g. red) due to the selection of a saddle. Now, it is not only possible to mark a different independent subtree by selecting an appropriate node, but also to select a node in the already highlighted subtree. In the latter case, the subtree corresponding to the last selected node is deselected. If the same node is selected again but with a different colour (e.g. green), the result is a subtree highlighted in the new colour *in* the first subtree highlighted with the first colour (see Fig. 3 left). If the disable button is not selected the render view will, of course, show the corresponding contours in the correct colours. There is no restriction in the combination of colours and selecting subtrees as described above. Thus, the user is provided with a completely flexible interactive interface to the contour tree, its corresponding generalised largest contours and combinations of the latter.

One of the standard tasks using the interface is to mark a certain saddle and inspect its generalised largest contour. Among other things, this enables the user to investigate the effects of simplifications because the generalised largest contour of a saddle corresponds to the largest contour of an extremum that would result from the simplification of the complete subtree of the saddle collapsing it to one extremum.

5 Implementation Details

5.1 Similarity Measure

In order to compute the similarity measure defined in Equation (1) we need the volume of each of the features and the volume of the region where the two features overlap. The volume of each feature in both scalar fields can be approximated by counting the vertices in the associated vertex set. If the associated vertex sets of two features are intersected we obtain an approximation of the shared volume.

This is achieved by running through the features of scalar field S_B and label the vertices of the associated vertex set with a unique feature identifier. The result is a simple lookup table that maps vertex ID's to feature ID's. Now the algorithm runs through the features in scalar field S_A looking up every vertex of the associated vertex set to which feature in scalar field S_B it belongs. Bearing this in mind, we can now identify overlapping features and simply count the shared vertices for every pair of overlapping features. It is of course possible to use more sophisticated methods, for example to weight the vertices with the volume of the cells connected to that vertex, to account for grids where features extend over different cell scales. However, the vertex based method performs completely sufficient for our datasets. Thus, in terms of quality, there was no need for a more complex and time consuming method.

In terms of speed, the algorithm takes $O(n)$ time, where n denotes the number of vertices in the grid. The first stage of the algorithm is to label all features in scalar field S_B which takes time proportional to the number of vertices in the features, which is $O(n)$. The second stage is to perform a lookup for every vertex which belongs to a feature of scalar field S_A which is, again, an $O(n)$ task. Therefore, overall complexity for computing all similarities is $O(n)$.

Table 1. Parameter settings used for the different datasets

Data set	Variable	Percent of vertices	Similarity threshold
Delta wing	Eddy viscosity Signed helicity	0.03% 0.03%	0.2
Turbine draft tube	λ_2 Pressure	0.1% 0.02%	0.2
Cuboid	λ_2 Pressure	0.03% 0.01%	0.2
Blunt fin	Density Pressure	0.1% 0.1%	0.2

5.2 Thumbnails

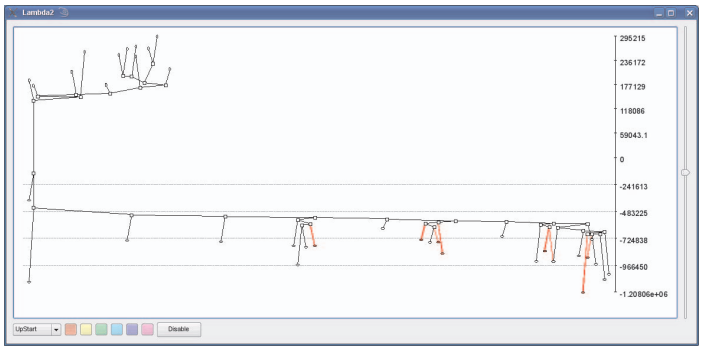
For the automatic thumbnail generation mentioned above, a suitable perspective must be chosen for each of the images. This is achieved by performing a standard principal component analysis (PCA) for the vertices belonging to both features to be depicted in the thumbnail image. From the obtained PCA vectors the one with the smallest eigenvalue is chosen to be the viewing direction of the camera. The remaining two vectors, which are perpendicular to the viewing vector, exhibiting higher eigenvalues corresponding to higher variance thus maximise projection area on the screen. The camera is then adjusted to look at the mean of the vertex positions. Finally, the bounding box of the vertices is obtained to fit the contours to the size of the desired thumbnail. Examples can be found in Figures 2(b), 4(c) and 5(c).

5.3 Overlapping Boundary Surfaces

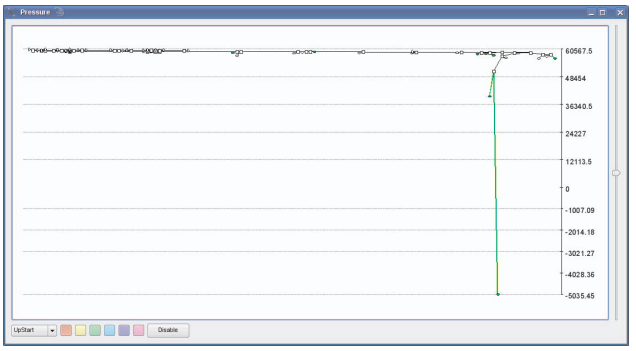
In the process of comparing two scalar fields it might be the case that boundary surfaces of contours overlap. Therefore, the boundary surface of the contour with the higher vertex count is moved away from the grid boundary because we assume that this contour would enclose the one with the smaller vertex count. However, every vertex of the surface to be moved away that was involved in the boundary surface extraction is moved along its vertex normal, which is obtained by averaging the surface normals of the neighbouring triangles. These are the same vertex normals as used for shading. As a heuristic, the distance by which the vertices are moved away from their original position is determined by the first edge found in the first cell of the dataset. Alternatively, the distance can be defined manually.

6 Results

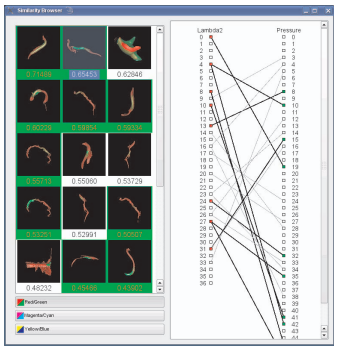
In this section, we will provide a number of examples that demonstrate the discussed visualization methods on the CFD application datasets



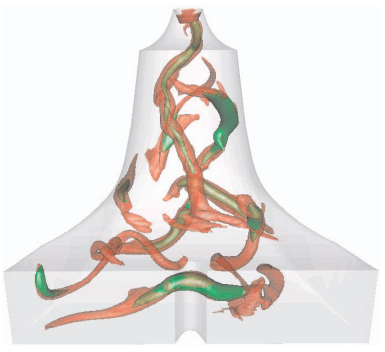
(a) λ_2 Contour Tree



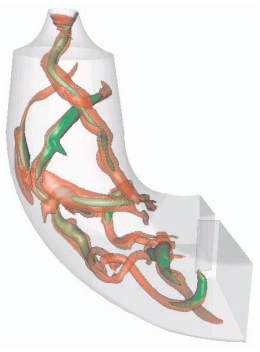
(b) Pressure Contour Tree



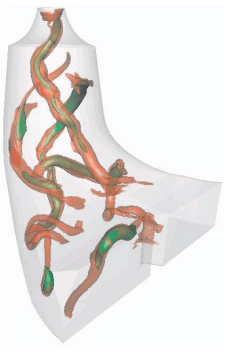
(c) Similarity Browser



(d) Front



(e) Left



(f) Overview

Fig. 4. Turbine data set: (a),(b) Contour trees for pressure related to vortices. (c) Similarity Browser shows selected contours. Not all vortical structures have been selected to avoid visual clutter in render view. (d)-(f) Different views of the selected contours.

presented in Section 2. Most of the datasets are given on hexahedral grids while one, the delta wing, is given on an unstructured grid with mixed cell types. As the original contour tree implementations can only handle simplicial meshes, we perform a preprocessing step to convert all grids using an algorithm proposed by Wiebel [32].

Table 1 lists the percentage thresholds used for simplifying the contour trees of the different scalar fields and the minimal similarity of thumbnails shown in the browser. The simplification thresholds are given as percentage of the number of total vertices in the dataset which are easier to handle since different quantities have different units and scales. As already mentioned, in terms of largest contours, the threshold can be interpreted as a lower bound for the size (number of vertices) of largest contours appearing in the contour tree.

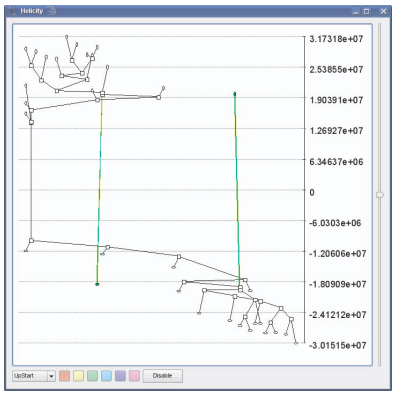
Cuboid The cuboid dataset is an example where a strong correspondence between different scalar quantities can be found. The render view in Figure 2(d) shows overlapping contours of λ_2 and pressure. It is known that both exhibit small values in regions of vortical behaviour. This fact can be easily identified by the contours drawn in the render view. Each pair of corresponding contours identifies the region of a hairpin vortex.

We have labelled four different contour pairs with letters in all views to show the connections of the views. For example, consider the hairpin vortex labelled *b* in Figure 2(d). It is easily recognisable in the *similarity browser* 2(b). The browser also shows that the contours of λ_2 and pressure belonging to the hairpin vortex have a large similarity. This confirms what can be seen in the render view. The graph view reveals that the overlapping contours are the third largest of λ_2 contours and the second largest for pressure. Additionally, it can be seen immediately that the considered contours have a one-to-one correspondence, i.e. no other contours overlap them, as both nodes have only one edge connected to them. Finally, the contours are also labelled in the contour trees (Figures 2(a) and 2(c)). The relations of the different contours within a single scalar field and the scalar values (see the scale on the right side of Figure 2(c)) become clear in this view. Obviously, the two vortices closest to the cuboid are the strongest as they exhibit the lowest λ_2 and the strongest pressure minima. This is not surprising, as the vortices are induced by the cuboid and are damped by the surrounding flow downstream.

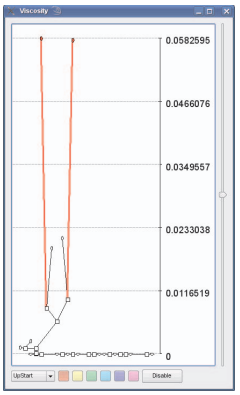
Turbine Draft Tube Figure 4 illustrates the application of the comparison techniques provided by the *similarity browser* to the tube dataset. The browser shows a large number of thumbnails for similar features in λ_2 and pressure. While the number and the size of the contours depend on the chosen simplification threshold, the large number of matches still indicates a strong correlation of the compared quantities. The graph view of the browser shows that most feature correlations are one-to-one as desired. There are only few nodes with two edges connected to them and no nodes of higher degree. The selected overlapping features are shown from three different perspectives in images 4(d) to 4(f). It is easily visible that both quantities, λ_2 (red) and pressure (green), nicely capture the main vortex that is connected to the upper end of the tube. Even the second largest vortex, visible in the upper right part of Figure 4(d), is identified by large overlapping contours.

Delta Wing Without further knowledge we applied our scalar field comparison technique to the delta wing data set. We compared some of the available scalar variables. Not surprisingly vorticity and pressure yielded nicely matching contours covering the vortices. A comparison of signed helicity[2] and eddy viscosity, however, was more interesting. While we already knew of the connection between helicity and vortex breakdown as mentioned above, we found that eddy viscosity is also connected to the phenomenon. The *similarity browser* (Figure 5(c)) shows only four contour pairs with a similarity above the user selected threshold of 0.2. Selecting the two pairs with the significantly larger similarity yields the images in Figures 5 and 6. The

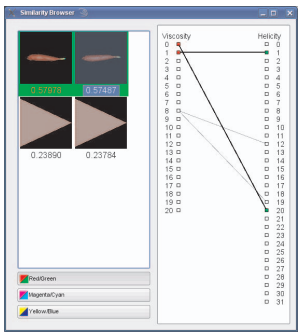
[2]Throughout this paper *signed helicity* refers to the dot product of the velocity and the vorticity vectors: $v \cdot (\nabla \times v)$



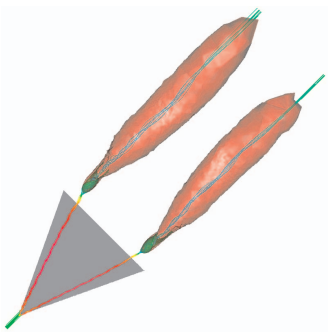
(a) Helicity Contour Tree



(b) Eddy Viscosity Contour Tree

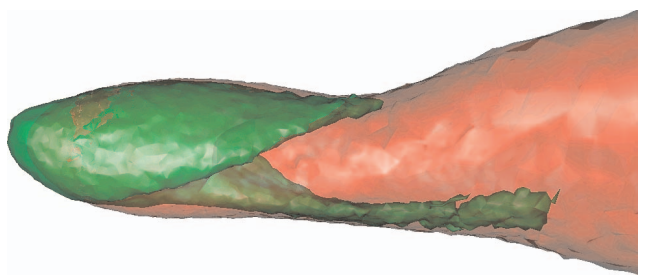


(c) Similarity Browser

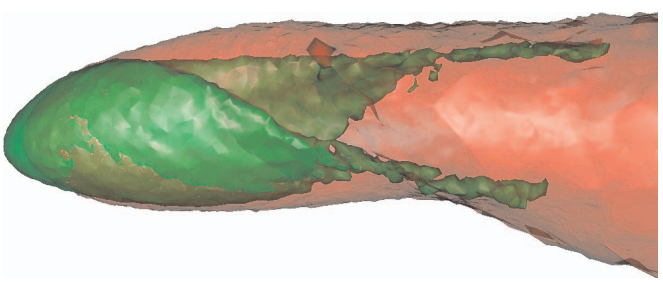


(d) Overview

Fig. 5. Contour trees for helicity (a) and eddy viscosity (b) of delta wing dataset. Selecting helicity maximum and minimum in (a) causes two viscosity maxima in (b) to be highlighted. Image (d) shows the corresponding largest contours in the main render view. The vortex breakdown bubbles are immediately visible. Image (c) shows the similarity browser.



(a) Left Bubble



(b) Right Bubble

Fig. 6. Close up renderings of the two vortex breakdown bubbles of the flow around the delta wing.

overlapping parts of the contours clearly identify the two vortex breakdown bubbles (Figure 6).

We contacted our cooperation partner from the German Aerospace Center who performed the simulation to dicuss our exploratory findings. He told us that small scale turbulence appears during the transition from the vortex to the bubble. As eddy viscosity describes the viscosity caused by turbulence, this means that it is indeed connected to vortex breakdown.

As explained above, we found the connection by using the *similarity browser*. Considering images 5(a) and 5(b) it is obvious that the connection is also detectable by just using the contour tree views. The two signed helicity extrema belonging to the bubbles are clearly standing out in Figure 5(a). Selecting one of the extrema causes the corresponding eddy viscosity maximum (also very prominent) to be highlighted in the second contour tree view (Figure 5(b)). The contours in the main render view are the same as when using the *similarity browser*, thus making the bubbles immediately visible.

Blunt Fin The blunt fin dataset is a perfect example where the comparison of feature related quantities immediately reveals the feature itself. As mentioned, it is known that a shock can be described by a high gradient in density and pressure [27]. Indeed, the comparison of the topology of pressure and density immediately identifies the shock in front of the fin as the contours with the largest similarity values (see Figure 7) in the *similarity browser*.

7 Conclusion and Future Work

We have described the application and implementation of a new interface for the comparison of two scalar fields and applied it to a number of fluid flow datasets. The comparison is based on overlapping largest contours. The largest contour around a local extremum is defined as a feature of the scalar field. The spatial overlap of such features between different scalar fields is calculated in a symmetric way and called similarity. Our system calculates all overlapping features based on simplified contour trees. The found overlaps are sorted based on similarity and presented in the *similarity browser*. This browser uses automatically generated thumbnail images and a bipartite graph to visually represent the found similarity of the two scalar fields. Enriched with linked views of the two underlying contour trees and a 3D render view, we obtain an interface for visual comparison.

Using the interface we were able to visualize the relation of certain scalar quantities to the existence of vortices. Additionally, we "rediscovered" the connection between vortex burst and high eddy viscosity by an exploratory usage of the browser. The graph view in the *similarity browser* shows the connections between different features allowing to search for one-to-one or one-to-many correspondences of contours. Furthermore, linked contour tree views allow for a more detailed interactive comparison analysis. Although we focused on flow data in this paper, we believe that the analysis of other data, for example from medical imaging, can benefit from the presented techniques as well.

Albeit the chosen similarity measure of spatial overlap worked well and is intuitive, dependencies which are not manifesting in such a way are not detected. Thus, different or more sophisticated similarity measures might be better suited for other domains. The same argument applies to the simplification measure where different measures like persistence or hypervolume might yield better results for other domains or different problems.

We plan to augment the contour tree views with thumbnails. Unfortunately, extending our work to time-varying data, i.e. time-dependent contour trees, is not straight forward and thus is also part of future work. As mentioned throughout the paper we are only dealing with simplicial meshes so far and use simplicial subdivision to be able to treat non-simplicial meshes. We are aware of the possible sampling artifacts [4] and plan to extend our method to be able to treat hexahedral cells directly as described in [5]. Finally, the overlapping contour display could benefit from the application of a technique presented by Weigle et al. [31]

Acknowledgements

We wish to thank Markus Rütten from German Aerospace Center (DLR) in Göttingen for providing the delta wing data set and the verification of our assumptions concerning the connections between vortex

breakdown and eddy viscosity. Thanks also go to VA Tech Hydro and Ronald Peikert from ETH Zürich for providing the tube datasets. This work was partially supported by DFG grant SCHE 663/3-7.

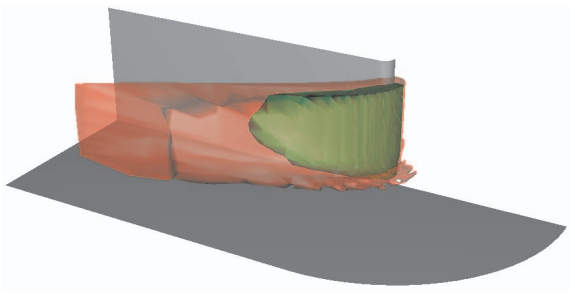


Fig. 7. Shock wave in blunt fin data set. Overlapping contours of density (red) and pressure (green).

REFERENCES

[1] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time- varying multivariate volume data. In *Eurographics / IEEE Symposium on Visualization (EuroVis)*, pages 115–122, 2007.

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The Contour Spectrum. In *Proceedings of Visualization 1997*, pages 167–173, 1997.

[3] R. L. Boyell and H. Ruston. Hybrid Techniques for Real-time Radar Simulation. In *Proceedings of the 1963 Fall Joint Computer Conference*, pages 445–458. IEEE, 1963.

[4] H. Carr, T. Möller, and J. Snoeyink. Artifacts Caused by Simplicial Subdivision. *IEEE Trans. on Vis. and Comp. Graphics*, 12(2):231–242, 2006.

[5] H. Carr and J. Snoeyink. Representing interpolant topology for contour tree computation. In *Proc. Topology-based Methods in Visualization 2007 Workshop*. Springer. to appear.

[6] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 49–58, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[7] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. IEEE Visualization '04*, pages 497–504, Washington, DC, USA, 2004. IEEE Computer Society.

[8] C. Garth, X. Tricoche, and G. Scheuermann. Tracking of Vector Field Singularities in Unstructured 3D Time-Dependent Datasets. In H. Rushmeier, G. Turk, and J. J. van Wijk, editors, *Proc. of the IEEE Visualization 2004 (VIS'04)*, pages 329 – 336. IEEE Computer Society, October 2004.

[9] L. J. Gosink, J. C. Anderson, E. W. Bethel, and K. I. Joy. Variable interactions in query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1400–1408, November/December 2007.

[10] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, 1998.

[11] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *SIGGRAPH 2001*, pages 203–212, 2001.

[12] C. Hung and P. Buning. Simulation of blunt-fin induced shock wave and turbulent boundary layer separation, AIAA Paper 84-0457. In *AIAA Aerospace Sciences Conference*, Reno, NV, January 1984. AIAA.

[13] T. Itoh, Y. Yamaguchi, and K. Koyamada. Fast Isosurface Generation Using the Volume Thinning Algorithm. *IEEE TVCG*, 7(1):32–46, 2001.

[14] H. Jänicke, A. Wiebel, G. Scheuermann, and W. Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1384–1399, 2007.

[15] J. Jeong and F. Hussain. On the Identification of a Vortex. *Journal of Fluid Mechanics*, 285:69 – 94, 1995.

[16] G. Ji, H.-W. Shen, and R. Wenger. Volume tracking using higher dimensional isosurfacing. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, pages 209–216, Washington, DC, USA, 2003. IEEE Computer Society.

[17] J. Kniss, G. Kindlmann, and C. D. Hansen. Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. In *Proc. of Visualization 2001*, pages 255–262, 562, 2001.

[18] R. S. Laramee, J. Schneider, and H. Hauser. Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics . pages 85–90, Konstanz, Germany, 2004. Eurographics Association.

[19] H. J. Lugt. *Introduction to Vortex Theory*. Vortex Flow Press, Inc., Potomac, Maryland, 1996.

[20] E. M. M. Manders, R. Hoebe, J. Strackee, A. M. Vossepoel, and J. A. Aten. Largest contour segmentation: A tool for the localization of spots in confocal images. *Cytometry*, 23:15–21, 1996.

[21] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, Sept-Oct 2006.

[22] D. Silver. Object-oriented visualization. *IEEE CGA*, 15(3):54–62, 1995.

[23] B.-S. Sohn and C. L. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006.

[24] S. Stegmaier, U. Rist, and T. Ertl. Opening the Can of Worms: An Exploration Tool for Vortical Flows. In C. Silva and E. Gröller and H. Rushmeier, editor, *Proceedings of IEEE Visualization '05*, pages 463–470. IEEE, 2005.

[25] D. Sujudi and R. Haimes. Identification of Swirling Flow in 3-D Vector Fields. In *12th AIAA CFD Conference*, San Diego CA, June 1995.

[26] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson. Introducing topological attributes for objective-based visualization of simulated datasets. In *Proc. Volume Graphics 2005*, pages 137–146,236, 2005.

[27] E. Truckenbrodt. *Fluidmechanik*, volume 2. Springer-Verlag, 1980.

[28] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. R. Schikore. Contour Trees and Small Seed Sets for Isosurface Traversal. In *Proc. 13th ACM Symp. on Comput. Geometry*, pages 212–220, 1997.

[29] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[30] G. Weber, S. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):330–341, March/April 2007.

[31] C. Weigle and R. M. T. II. Visualizing intersecting surfaces with nested-surface techniques. In *IEEE Visualization*, page 64. IEEE Computer Society, 2005.

[32] A. Wiebel. Tetrahedrisierung irregulärer Gitter und 3D-Helmholtz-Hodge-Zerlegung von Vektorfeldern. Projektarbeit, University of Kaiserslautern, Fachbereich Informatik, AG Visualisierung, July 2003.

[33] A. Wiebel and G. Scheuermann. Eyelet Particle Tracing - Steady Visualization of Unsteady Flow. In *IEEE Visualization 2005 - (VIS'05)*, pages 607–614. IEEE Computer Society, October 2005.

[34] A. Wilhelm. *Handbook of Data Visualization*, chapter II.9. Springer Handbooks Comp. Statistics. Springer, 2008.

[35] J. Woodring and H.-W. Shen. Multi-variate, time-varying and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):909–916, Sept-Oct 2006.

[36] X. Zhang, C. L. Bajaj, and N. Baker. Fast Matching of Volumetric Functions Using Multi-resolution Dual Contour Trees. Technical report, Texas Institute for Computational and Applied Mathematics, Austin, Texas, 2004.