Interactive Vector Field Feature Identification

Joel Daniels II, Erik W. Anderson, *Student Member, IEEE*,
Luis Gustavo Nonato
and Cláudio T. Silva, *Senior Member, IEEE*

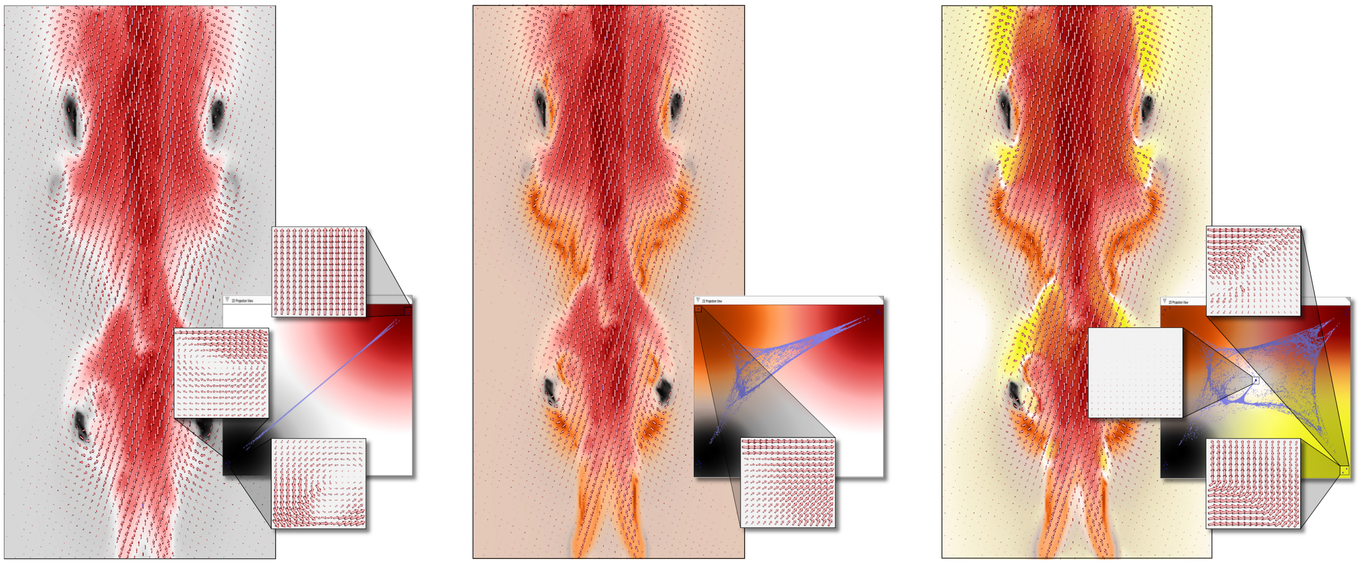


Fig. 1. Above is a sequence of interactions performed during an exploration session in our system (also demonstrated in our accompanying video). Our interactive framework allows the user to specify representative control points (insets) of desired feature types. These control points guide a mapping of the vector field points to the interactive texture canvas, where distances between the projected points encode similarities between their localized neighborhoods. Feature-based visualizations are generated through a painting interface, performed on this canvas.

Abstract—
We introduce a flexible technique for interactive exploration of vector field data through classification derived from user-specified feature templates. Our method is founded on the observation that, while similar features within the vector field may be spatially disparate, they share similar neighborhood characteristics. Users generate feature-based visualizations by interactively highlighting well-accepted and domain specific representative feature points. Feature exploration begins with the computation of attributes that describe the neighborhood of each sample within the input vector field. Compilation of these attributes forms a representation of the vector field samples in the attribute space. We project the attribute points onto the canonical $2D$ plane to enable interactive exploration of the vector field using a painting interface. The projection encodes the similarities between vector field points within the distances computed between their associated attribute points. The proposed method is performed at interactive rates for enhanced user experience and is completely flexible as showcased by the simultaneous identification of diverse feature types.

Index Terms—Vector field, data clustering, feature classification, high-dimensional data, user interaction

1 Introduction

Vector field visualization is a challenging research problem that receives continuous attention from researchers in the community. The ability to explore and analyze vector fields is becoming increasingly important as the datasets grow in size and complexity. Scientists are collecting more detailed measurements of real world phenomena, physically-based simulations are generating larger and more elaborate datasets, while computer hardware advances are improving computational power. These factors motivate the need for new interactive vector field data exploration and visualization tools.

- *Daniels, Anderson and Silva are with the School of Computing and Scientific Computing and Imaging Institute, University of Utah, USA. E-mail: {jdaniels,eranders,csilva}@cs.utah.edu.*
- *Nonato is with Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Brazil. E-mail: gnonato@icmc.usp.br.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

Integral curves are a powerful and popular vector field visualization technique [43], evident by the number of variants and related research efforts [26]. These techniques follow the motion of a particle through the vector field, visualizing its trajectory. The effectiveness of these techniques is based on their ability to mimic natural phenomena; such as, smoke highlighting the updraft above a flame or creamer depicting the swirling current of your morning coffee.

Despite the utility of integral curve based visualizations, they are not without their complications. Particle tracing, in particular streamlines, stream-surfaces and dye advection, requires intelligent placement of seed points in order to highlight interesting features. When performed manually, for user driven exploration, it is easy to overlook important structures in the vector flow. Furthermore, these methods provide an exploration tool but neither identify nor extract structural information of the vector field to facilitate other processing techniques.

Feature identification [31, 14], extraction [24, 23] and tracking [35, 3, 42] approaches address these concerns and have been developed for domain specific feature types. The structural representations of feature boundaries, automatically generated by these methods, prove useful while visually browsing large vector fields to quickly identify interesting components. This added structural information facilitates other vector field processing; for example, guiding the placement of seed points for integral curve visualizations.

Many algorithms explicitly define feature types, restricting them to field singularities and vortices. However, in real world applications, the definition of a feature is driven by the dataset, simulation and scientists. For instance, ocean and weather visualizations indicate an interest in shearing flows [10], engine combustion [13] and aerodynamics simulations [20] focus on vortices, while wildfire prediction models analyze updrafts and down wind laminar flows [28].

In this paper, we introduce a novel and flexible method to interactively visually highlight user-specified feature types within a vector field. Our feature exploration method adopts philosophies from database retrieval techniques that are used in a variety of applications from natural language processing [37] to geometric shape comparisons [41]. In these domains, feature vectors are associated with the database entries. Distance computations between feature vectors are used to infer similarity relationships between the associated entries. Similarly, we map the vector field samples into an attribute space where neighborhood queries convey relational properties to facilitate feature classification, illustrated in Fig. 1.

Contributions We introduce a novel interactive vector field visualization technique that supports flexible exploration and classification. We identify the following contributions:

- **Flexible feature identification:** The framework is not specific to any research domain and can adapt to user specifications by classifying any desired feature type.
- **Novel application of a projection:** We build on an approach, typically found in the information visualization domain, to solve a scientific visualization problem.
- **Interactive vector field exploration:** The implementation makes use of efficient linear system libraries, threaded computations, and texture-based visualizations to focus on providing an interactive exploration experience.

2 Related Work

The rendering and study of vector fields has been an ongoing source of research challenges for the past twenty years [27]. Numerical integration forms a popular subset of vector field visualization techniques based on the real-world observations of the effectiveness of particle movement to accentuate flow while tracking velocity [1]. Many varieties of numerical integration based techniques exist, rendering: tubular trajectory paths (streamlines) [4, 11], multiple particle traces [33], surfaces and volumes created by sets of trajectories [16, 25], and advect clouds of dye [39, 22] extended for large and time-varying flow fields [21, 8]. Seeding these algorithms is important for their effectiveness, and while approaches automatically address this challenge [43, 44], manual exploration can be problematic. Furthermore, numerical instabilities of the vector field integration may produce incorrect visualization results. Line integral convolution methods [6, 19] overcome seeding challenges using global advection of randomly generated textures. We similarly use texture mapping to achieve our visualizations, but do not advect points, rather we focus on data clustering.

Feature Characteristics While our framework develops visualizations of vector field data, its goal better aligns with feature classification and extraction methods [32]. Such feature-based visualization approaches rely on the computation of local neighborhoods in order to identify certain structural elements. For instance, critical points, where the vector magnitudes vanish, describe important features, i.e., sources, sinks, vortices and saddles. Detection of these points, studying the eigenvalues of localized Jacobian [15] and Hodge decompositions [31], is useful in topology-based segmentation solutions [45].

Vortex specific detection methods are based on the observation that these regions are described by high rotation [2]. Curvature-based methods for vortex identification approximate the curvature in the flow of localized neighborhoods [36, 29]. Velocity gradient-based approaches have also been used for this purpose [7], but problems arise with first-order approximations that higher-order methods try to overcome [34]. The λ_2 operator [18] accommodates for these challenges by refering to the second eigenvalue of the tensor $S^2 + \Omega^2$, being the symmetric and antisymmetric parts of the velocity gradients (Sec. 4.1).

User-driven Classification While the previous methods are successful in extracting feature information, they are restricted to specific feature types. Several methods for flow characterization aim for more generic feature identification solutions. Our work is most similar in nature to the flow characterization methods using pattern matching and neighborhood comparisons.

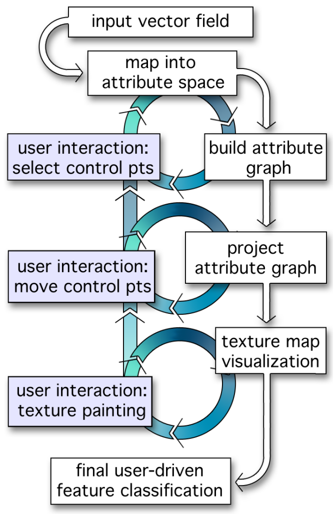
One such method compares localized vector neighborhoods to a set of idealized, normalized feature neighborhoods [14]. Our method is related in that we compare, albeit implicitly, the similarity of the vector field point neighborhoods to a set of desired feature neighborhoods. In contrast, by defining rotationally invariant attribute descriptors our framework does not require comparison to rotations of the input patterns. Our framework is more flexible, allowing on-the-fly creation of new features and facilitating interactive user-driven classification.

A second pattern matching scheme uses rotation, scale and translation invariant *moments* [38]. These moments are shown to be effective at differentiating flow patterns within 2D vector fields, and would straightforwardly apply within our attribute space computation. These invariant moments would provide an appropriate high dimensional space for feature classification within our framework. Instead, we rely on other computations widely used in feature extraction, i.e., λ_2, for their known extensions to volumetric flow, a future goal of this work.

Lastly, our work is most similar in goal and approach to multi-variate brushing [17]. In this work, the Fruchterman-Reingold graph layout algorithm projects a spanning tree, constructed over data points described in a high dimensional space by their multi-variate values, to the plane. The user interacts with the visualizations by brushing on the projection. Our framework is built on similar interactions, but implements a single direct solve for the parameterization, in contrast to iterative force-based minimization. Our direct solve enables interactive control of the projection and data clustering. We further distinguish our method by using control points representative of idealized or user-selected feature types, to aid navigation and user brushing of the 2D projections.

3 System Overview

Illustrated on the right, our method begins by mapping the vector field points into an attribute space (Sec. 4). Each coordinate of the attribute space evaluates a specific computation over localized vector neighborhoods. Next, a bidirectional graph is constructed from the attribute points that infers the global relationships within the vector field. The user identifies a set of control points with vector neighborhoods that are representative of the desired feature types. These points are mapped into the attribute space and integrated within the attribute graph. Then, a linear system solve projects the attribute points to our interactive texture canvas (Sec. 5), assigning the uv-coordinates to each vector


input vector field
map into attribute space
build attribute graph
project attribute graph
texture map visualization
final user-driven feature classification
user interaction: select control pts
user interaction: move control pts
user interaction: texture painting

field point used to color the final feature-based visualizations via texture mapping. The user customizes the final visualization by modifying the control point locations and designing a texture over the interactive canvas (Sec. 6). Our framework enables a flexible and interactive exploration of the vector field features (Sec. 7).

4 Attribute Neighborhood Graph

Our technique is founded on the observation that, while similar features within the dataset may be spatially disparate, they share similar vector field neighborhood characteristics. As such, we associate attribute feature vectors with each vector field sample, mapping the vector field points into a separate attribute space. In the remainder of the paper spatial vector field neighborhoods of a point are referred to as *VS-neighbors* while the neighborhoods in attribute space are denoted *AS-neighbors*. The *AS*-neighbors of a point provide insight into the relationships between different regions of the vector field (Fig. 2).

This section discusses the computation of the attribute feature vectors and, over them, the construction of an attribute neighborhood graph, necessary for the projection computations (Sec. 5). A binary space partitioning data structure facilitates distance computations and *AS*-neighborhood queries. Lastly, this section details the necessary precautions taken to build a graph with a single connected component.

4.1 Attribute Feature Vectors

An attribute is a measurable trait computed over the *VS*-neighborhood of a sample point in the vector field. Each attribute describes a single component of the attribute feature vectors associated with the points in the vector field, illustrated in Fig. 2. The following list names some attributes and describes their related computations used in the examples shown throughout this paper.

X/Y-Component The computation of the x- or y-component returns the corresponding value of the normalized vector for a given point within the vector field.

Speed The speed computation measures the magnitude of the average vector computed over the *VS*-neighborhood of a point.

Flux The flux attribute approximates the mass transport in/out of the point p_i's *VS*-neighborhood, $\mathcal{N}_i^{\text{VS}}$. The attribute is computed by first walking the boundary of the *VS*-neighborhood to evaluate a centroidal point, p_c. The flux is then approximated by summing the inner products of the *VS*-neighborhood vectors v_j, $\forall j \in \mathcal{N}_i^{\text{VS}}$, and the difference of the *VS*-neighbor point p_j and p_c: $\sum_j \langle v_j, p_j - p_c \rangle$. Because this method is summed over all points, rather than focusing on the flux through the boundary, it supports vector fields in which mass is injected. The mass transport is normalized by the area of p_i's *VS*-neighborhood to accommodate for boundary points.

λ_2 **Operator** The λ_2 attribute differentiates vortex *VS*-neighborhoods, a popular computation in the feature identification literature [18]. This component builds the Jacobian matrix J (the velocity gradient tensor) for a point, and computes the related matrices: $S = (J + J^T)/2$ and $\Omega = (J - J^T)/2$. The λ_2 attribute is the second eigenvalue of the matrix defined by $S^2 + \Omega^2$.

We individually normalize the attributes to the range $[-1, 1]$ so that each component of the attribute feature vector is equally represented. The l^2-norm of the difference vector between two attribute feature vectors v_i and v_j computes the distance d between them. This distance expresses the similarity of the *VS*-neighborhoods of v_i and v_j. It is straightforward to set the relative importance of different attributes by scaling the range over which they are normalized, increasing their maximum possible contribution to the l^2-norm, thus effecting the computation of *AS*-neighborhoods.

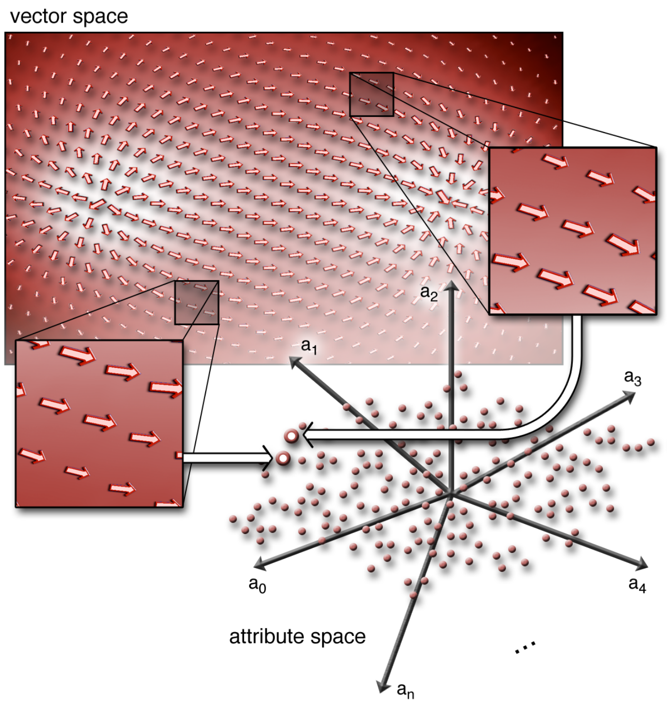
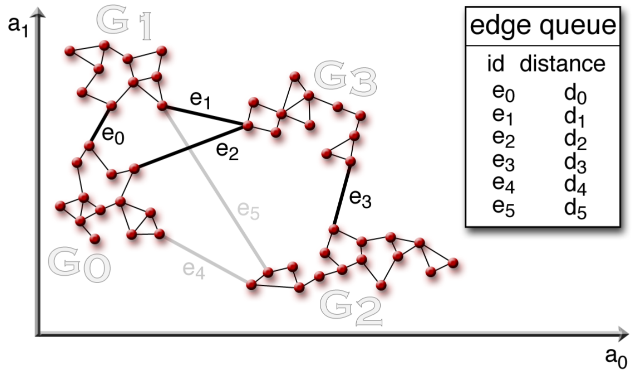


Fig. 2. The vector field points, colored based on the associated vector magnitudes, are mapped into an attribute space defined by attribute components $\{a_i\}_{i=0}^n$. Spatial proximity of two points in the attribute space relates to the similarity between them.

A binary space partitioning (BSP) tree accelerates nearest *AS*-neighbor queries. Each node is split into two based on the median value of a dimension of the node's attribute feature vectors. The sorting dimension used is computed based on the depth of the node within the BSP-tree, sequentially looping over the attribute space dimensions as one walks deeper into the tree structure. In practice, this splitting scheme is efficient because of our use of the Euclidean distance computation to compare the attribute feature vectors.

4.2 Building the Graph

The k-nearest *AS*-neighbor search drives the construction of the graph $\mathcal{G}$. Bidirectional graph edges are stored between each attribute feature vector and its k *AS*-neighbors. In practice $k = 13$, large enough to ensure a well connected graph, but not so large as to significantly



edge queue	
id	distance
e_0	d_0
e_1	d_1
e_2	d_2
e_3	d_3
e_4	d_4
e_5	d_5

Fig. 3. The initial k-graph constructed over the points in the attribute space may consist of multiple connected components $\mathcal{G}_i$. We resolve these by introducing new edges e_x between the closest pairs of connected components, illustrated above for a 2-dimensional example, until a single connected component is described.

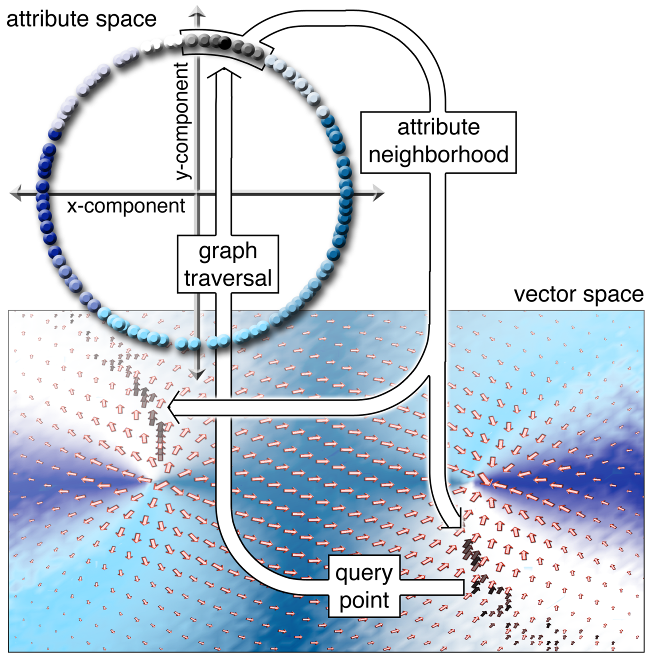


Fig. 4. *AS*-neighborhood queries initiate a traversal of the attribute graph from a chosen seed point, revealing vector field similarities. Above, the darkened vector glyphs correspond to neighboring points in the attribute graph. The attributes computed for the dataset above are the xy-components of the normalized vectors. As such, the neighborhoods are expected to contain points with similar vector directions.

slow down the construction. Note that after inserting the k nearest *AS*-neighbor edges, the number of edges emanating from each attribute feature vector is at minimum k. Augmented by the BSP-tree, the construction of an initial attribute graph is fast (Sec. 7).

It is possible that the initial graph $\mathcal{G}$ contains multiple connected components. While the connected components reflect the clusters of similar vector field samples, we require a single connected component for the projection technique discussed in Sec. 5. We generate a single connected graph structure, as illustrated in Fig. 3, by iteratively connecting the closest pairs of connected components. This is performed by constructing BSP-tree representations of each connected component to augment distance computations. A priority queue sorts the pairs of closest points, computed between each pair of graph components, from nearest to furthest. Finally, the algorithm inserts new edges into $\mathcal{G}$ until all components are connected by popping entries from the queue.

Attribute Neighborhoods *AS*-neighborhood queries of the attribute graph $\mathcal{G}$ allow a user to explore relationships within the vector field by quickly highlighting similar regions throughout the input dataset. Illustrated in Fig. 4, a graph traversal of $\mathcal{G}$ from a user selected seed point is used to visualize the regions of the vector field with comparable *VS*-neighborhoods. The vector field vertices are colored based on the number of graph edges between them and the seed point, becoming lighter as the path lengths grow.

5 Attribute-based Parametrization

In this section, we further exploit the attribute graph to project the attribute feature vectors onto our interactive texture canvas. The resulting projection provides the foundation of our painting interface (Sec. 6) to facilitate vector field exploration. The uv-parametrization associated with each point directly define the texture coordinates used for color look-ups while displaying a tessellation of the vector field.

5.1 Solving the Projection

The mechanism we use to map points from the attribute space to the interactive texture canvas (a unit square) is based on the LSP projection technique [30]. However, instead of dealing with normal equations as originally proposed in [30], we use the penalty method to impose constraints in the Laplace matrix so that our approach is computationally more efficient.

The projection of the attribute graph $\mathcal{G}$ onto the interactive texture canvas is computed by solving two harmonic fields, u and v. The functions each map the graph to the real numbers, $f : \mathcal{G} \to \mathbb{R}$, bounded by the range $[0, 1]$. We build the scalar fields by individually solving the Laplace equation for each u and v,

$$\Delta f = \nabla^2 f = 0,$$

User-specified Dirichlet boundary conditions, further discussed in Sec. 5.2, are defined at a subset of attribute feature vectors $\mathcal{C} \in \mathcal{G}$. The penalty method constrains the system so that it evaluates to the desired values for these points, $f_i = c_i, \forall i \in \mathcal{C}$.

Over the attribute graph, the Laplacian operator is discretized as

$$\Delta f = \sum_{j \in \mathcal{N}_i^{\mathrm{G}}} w_{ij}(f_j - f_i),$$

where $\mathcal{N}_i^{\mathrm{G}}$ is the set of vertices in the 1-ring neighborhood of $\mathcal{G}$ for the attribute feature vector i, and w_{ij} is the weight assigned the edge between the points v_i and v_j. In [30], Euclidean distances serve as the weighting metric assigned to the graph edges. However, because minimizing the stress error of the projection is not our primary aim, we found combinatorial weights $w_{ij} = 1.0$ for all ij-pairs to be sufficient. This avoids potential complications associated with the matrix rank.

The system of linear equations described above can be rewritten as a matrix operation $\Delta f = -Lf$. The matrix L encodes the graph edges

$$L_{ij} = \begin{cases} \sum_{j \in \mathcal{N}_i^{\mathcal{G}}} w_{ij}, & \text{if } i = j, \\ -w_{ij}, & \text{if } j \in \mathcal{N}_i^{\mathrm{G}}, \\ 0, & \text{otherwise.} \end{cases}$$

Because the edges of the attribute graph $\mathcal{G}$ are bi-directional, L is a symmetric, positive-definite sparse matrix.

We use the penalty method to impose constraints to the linear system [46]. Given the set of attribute feature vector constraints $\mathcal{C}$, the harmonic scalar field is obtained by solving the linear system,

$$(L + P)f = Pb, \tag{1}$$

where P is a diagonal matrix with non-zero entries, $p_{ii} = \alpha$, for the entries of constrained elements $i \in \mathcal{C}$. The penalty weight α is a very large value, in practice $\alpha = 10^8$. The vector b has the entries

$$b_i = \begin{cases} c_i, & \text{if } i \in C, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where c_i is the constraint value to be associated with the i^{th} attribute feature vector.

We opt to apply constraints with the penalty method for performance purposes. This scheme is able to make use of supernodal solution methods [9] to update (and downdate) the Cholesky factorization, making it possible to efficiently include and remove constraints [46]. This is a crucial property, as our conditions change frequently with user interactions. We implemented the linear system solves using the CHOLMOD libraries [12]. In the following section we further discuss how the user defines and interacts with the Dirichlet boundary conditions of the linear system to manipulate the projection of the attribute feature vectors to the plane.

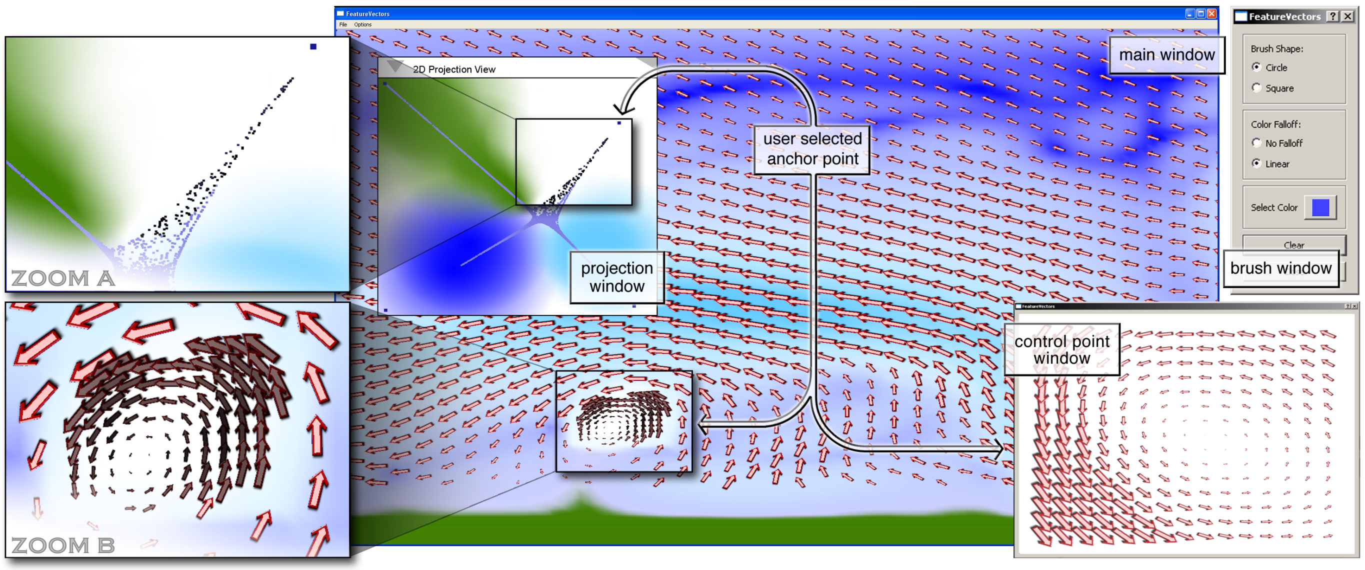


Fig. 5. The user interface is composed of multiple windows (labeled above) that manage a set of linked views of the dataset. The **main window** contains the vector field visualization with the user-designed texture as well as vector glyphs. The **projection window** provides the 2D canvas on which the user designs the highlighting texture, manipulates the location of control points, and views the projection of the attribute points. The **control point window** visualizes the selected control points, and, lastly, the **brush dialog** allows the user to affect their painting style.

5.2 Control Points

A control point is a representative point that describes a feature of interest and constrains the Laplacian system solve during the projection. To interact with the projection of the attribute feature vectors, a number of control points are defined within the attribute graph $\mathcal{G}$. These control points may be specified by the user in two ways: (1) loading pre-generated, idealized vector fields, and (2) selecting points of interest from the input vector field itself. Control points are assigned uv-coordinates within the interactive texture canvas, defining the Dirichlet boundary conditions of the u and v linear system solves, Equation 1.

We pre-generate a number of ideal vector fields centered around a feature point to be used as potential control points. For example, we build idealized source and sink singularities, stationary, laminar, divergent and convergent flows, as well as vortex and saddle vector field patterns. When these idealized fields describe feature types, the user loads them into our system through a menu drop down.

When the features are not known *a priori*, as in interactive exploration, or when the idealized vector fields are not representative of the features within the vector field, control points may be defined from the dataset itself. To do so, the user selects a vector field vertex. The *VS*-neighborhood of this point is duplicated and stored within the list of control points. This aspect of our approach allows the user to dynamically choose features of importance so that the method is not reliant on pre-defined feature types.

Completing the Graph When a control point is defined, the attribute graph $\mathcal{G}$ (Sec. 4) is augmented with the new attribute feature vectors. The control point p_i and its *VS*-neighborhood points $\{p_j\}_{j\in\mathcal{N}_i^{\text{VS}}}$ compute an associated set of attribute feature vectors v_i and $\{v_j\}$. These attribute feature vectors are inserted into the attribute graph $\mathcal{G}$ by including new bidirectional edges between the points $\{v_j\}$ and their k-nearest *AS*-neighbors within $\mathcal{G}$. Graph edges are inserted between v_i and each *VS*-neighborhood point $\{v_j\}$. Notably, we rely on the *VS*-neighborhood of a control point p_i, rather than its *AS*-neighborhood, to dictate the new graph edges. In practice, connecting the control points in this way improves the influence of these points, better distributing the projected points.

Completing the Solve Initially each control point is assigned the uv-coordinates $(0.5, 0.5)$, constraints defined in vector b (Equation 2). Consequently, the linear system solutions for u and v collapse the projection to a single point. In the following section, we discuss the system layout and user interactions that allow the projection to be intelligently distributed over the interactive texture canvas.

6 System Interface

We provide a suite of tools to modify the attribute point projections, design highlighting textures, and choose desired feature types in order to facilitate data exploration. The user interface is composed of multiple windows, a set of linked views, that enhances interactivity [5]. Illustrated in Fig. 5, the system interface is divided into a main window that displays both the vector and projected attribute spaces of the dataset. A second window displays a visualization of a control point's *VS*-neighborhood as the point is manipulated within the 2D projection view. Lastly, a third window provides the ability to modify the paint brush functionality.

Vector Field View The main window contains the primary view for vector visualization and data exploration. The flow fields are visualized with vector glyphs that provide contextual information. Selection of a vector glyph highlights the *AS*-neighborhood, illustrated in Fig. 5 as the blackened glyphs and projection points. While it is possible to load idealized vector field control points from file, within this main view it is also possible to *train the projection* by selecting control points directly from the input field. In practice, we have found this functionality to be pivotal in extracting unique features and differentiating between similar structures of the vector fields.

Projection View The 2D projection window displays the representation of the attribute space projected on the interactive texture canvas. In this window, the control points used to constrain the system solve (Sec. 5.1) are displayed in dark blue (Fig. 5). Moving a control point interactively changes the projected cloud of attribute points updating their uv-texture coordinates, demonstrating the relationships in the attribute space. The user designed visualizations are created by painting directly onto this view, rendering directly to a texture that is referenced while displaying the tessellation of the vector field domain. Brushing colored strokes over regions of this window containing subsets of the projection identifies interesting components of the vector field via tex-

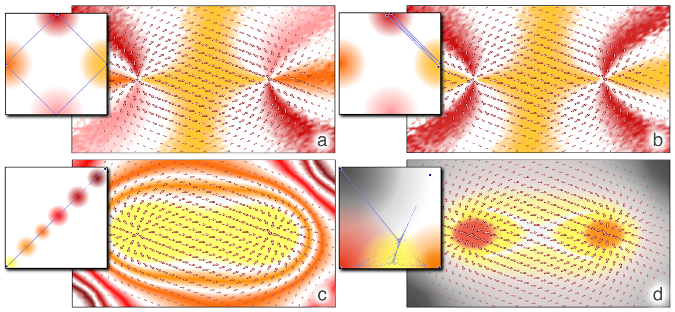


Fig. 6. Multiple visualizations for a magnet-like vector flow are generated by varying the attribute computations and control points used. Within the textures, projected points are light blue and control points are dark blue. Directional vectors are classified (a), and moving control points over the texture identifies two orthogonal vector directions (b). By changing the attribute computations and using a stationary and laminar flow control points, isolines of the vector magnitude are rendered via separate colored strokes (c). Additional control points describing divergence and convergence differentiate between the vector field critical points (d).

ture mapping. Illustrated in Fig. 5, the user defined texture classifies different features of the dataset.

Control Point View To facilitate the management of multiple control points, a second window displays their *VS*-neighborhoods. As the user manipulates them within the 2D projection view, this window updates the displayed *VS*-neighborhood glyphs to those of the chosen control point. This window plays a crucial role in differentiating between the separate control points.

User Interactions After loading a vector field dataset, users define control points to guide the projection of the attribute points. These control points may be loaded from a collection of analytically defined vector fields saved to file, or by *training the projection* extracting *VS*-neighborhoods directly from the dataset. Each control point is assigned an initial parameter value, in practice $(0.5, 0.5)$, such that the projection collapses to a single point.

The next task is to move the projected control points within the 2D projection window, distributing the attribute points over the canvas. Each control point constrains the Laplacian system (Sec. 5.1) by imposing Dirichlet boundary conditions for both the u- and v-solves, in accordance with their new positions on the interactive texture canvas. As the user modifies these boundary constraints, the system updates the uv-coordinates of all projected points. We are able to efficiently recompute the system solves and interactively update the projected attribute point locations by leveraging multi-nodal solution schemes for fast updates. In practice it is useful to separate dissimilar control points in the projection window to distribute the attribute points as they relate to each control. Sometimes additional training can be useful, extracting multiple control points from the dataset that are representative of the same feature type to better cluster the similar vector field points, illustrated in Fig. 1.

The final responsibility of the user is to design a texture within the projection window to highlight attribute points related to the various control points. This texture is loaded during the display of the vector field, instantaneously updating the resulting visualization. The use of the brush dialog, changing paint strokes and colors, drives our visually-based feature classification.

7 Results and Discussion

Before delving into results from complex datasets, we seek to enhance our understanding of the projection methodology and interaction process. Fig. 6 illustrates multiple visualizations of a manufactured magnet-like dataset produced using our framework. The vector

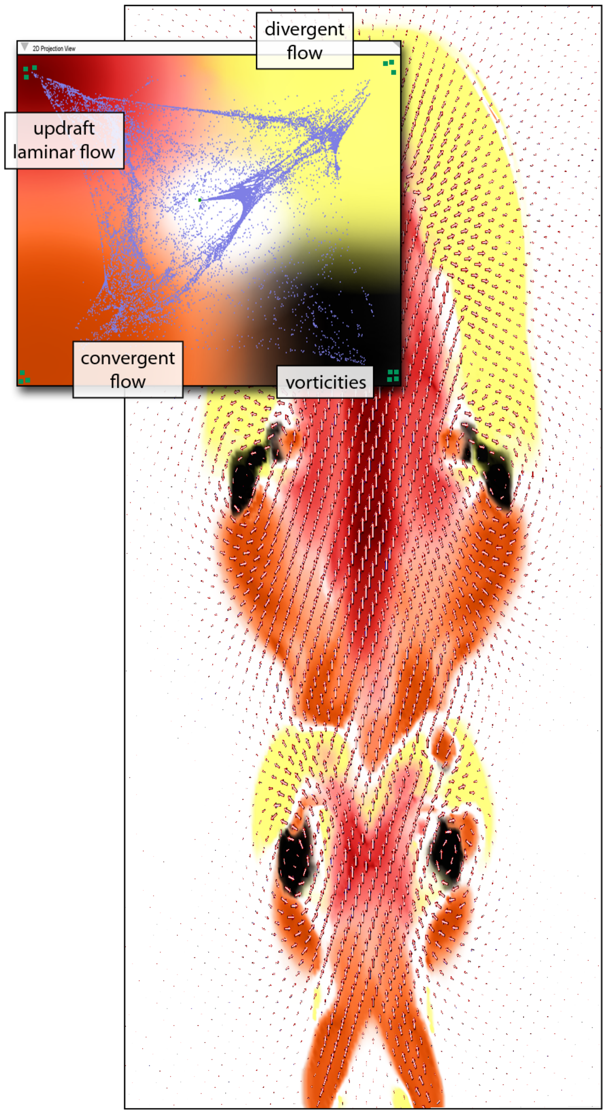


Fig. 7. Feature-based visualization of a combustion simulation shows four distinct flow types. Strong laminar flow (red) is near the center of the simulation. Convergent (orange), divergent (yellow) and vortex (black) flows are classified by projection training during the exploration.

field is generated with two singularity points, a source (left) and a sink (right), then small, random vector perturbations were introduced to each sample. We begin our discussion by describing the visualizations produced within this controlled environment.

The first two visualizations in Fig. 6 (a and b) have the visual effects of a normal map (a reproduction of the setup used in Fig 4). In this example, the attribute space is 2-dimensional, defined by the *X*- and *Y-component* attributes. The four control points are the laminar flows aligned in the positive and negative, X- and Y-directions. By arranging the four control points to the sides of the interactive texture canvas and painting different color strokes at each point (Fig. 6a), the visualization categorizes the vectors of the magnetic field based on their similarity to ideal laminar flows in each direction. Moving pairs of control points into the same locations (Fig. 6b) interactively updates the resulting projection and visualization.

The second visualization example (Fig. 6c) is similar to Fig. 2, extracting information concerning the vector magnitudes within the input field. In this scenario, the attribute space is 3-dimensional, described by the *speed*, *flux* and λ_2. The projected points are an affine combination of the two control points, the idealized stationary and laminar

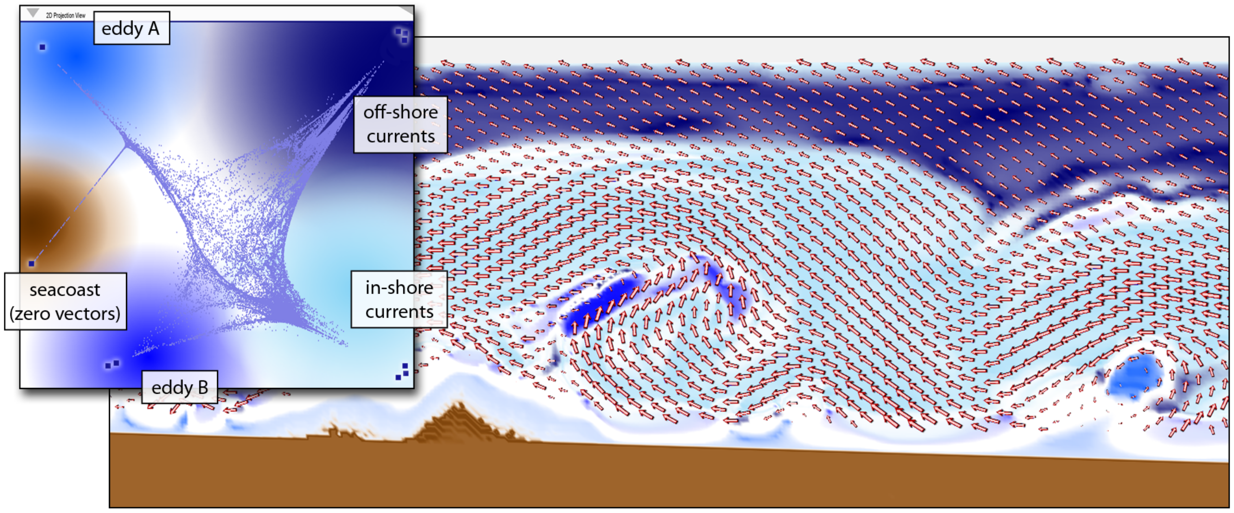


Fig. 8. Exploration of an ocean current simulation differentiating between two pairs of similar features: vortical eddies and laminar currents. Eddy A (light blue) highlights a circular vortex while Eddy B (dark blue) is elongated. In-shore currents (sky blue) are also differentiated from off-shore currents (navy blue).

flows, located on opposite sides of the texture canvas. Designing the texture with series of paint strokes along the projected points, varying in color from dark to light, identifies level-sets of the flow field with similar vector magnitudes.

The final magnet-like field example (Fig. 6d) demonstrates the effects of using additional control points. The attribute space is the same 3-dimensional coordinate system used in the previous example, and the four control points correspond to idealized stationary, laminar, divergent and convergent fields. The addition of the new control points enables the differentiation between source and sink singularities by classifying the projected points near to the divergent or convergent flow points, separately. This is highlighted in the image by using red and orange colors around the two idealized control points. The exploratory process for this example is also shown in the accompanying video.

7.1 Simulation datasets

To emphasize the effectiveness of our framework, we present the results of our software from several exploration sessions of different simulation datasets. These results showcase the flexibility of our system, identifying overlapping and unique features across the assorted models and sessions. The datasets are $2D$ slices extracted from the output volumes of (1) a combustion simulation of an explosion generated with four fuel injection points, and (2) an ocean current simulation along a section of the Pacific seacoast. The physically-based vector fields exhibit a dynamic nature with many interesting features, including swirling patterns, updrafts, shearing flows, among others.

Using our feature exploration framework, we highlight different components of the two datasets, illustrated in Figures 7 and 8. In the combustion results, the convergent (orange), divergent (yellow), vortices (black) and laminar updrafts (red) are individually characterized. The projected attribute points facilitate the global identification of the feature types defined by the user through the control point selection. For the ocean dataset, we further illustrate the power of our method by differentiating between variations of similar feature types. For instance, the results distinguish off-shore ocean currents (dark blue) from the shallower in-shore currents (sky blue), the difference being vector magnitudes. Similarly, we extract two eddies separately, one being oblong (blue) while the other follows a tight circular pattern (light blue).

In practice, the ability to train the projections is a useful functionality of our framework. We use the attribute graph AS-neighborhood queries to identify vector field points that represent interesting components for classification and use them as control points in the projection window. The textures are designed around the control points, highlighting each with unique colors to simultaneously classify distinct feature regions within the dataset. Feedback from engineers and domain scientists indicate that this framework is useful for the exploration of simulation data and debugging the code that generates it. Engineers exploring our system were able to immediately identify its utility for use in their everyday work ranging from analysis to visualization. For instance, a useful interaction is the identification of divergent flow in incompressible fluid simulations representing bugs in the underlying code.

7.2 Analysis and Comparison

The described method was designed in C++ using the CHOLMOD and Qt libraries. The running times for the assorted operations performed during the exploration sessions described in the previous sections were performed on commodity desktop computers with Linux, MacOS and Windows. Table 7.1 presents the median, minimum, maximum and standard deviation of the timings required to compute the attribute feature vectors, the initial graph setup, the single connected component resolution, and the uv-projection solves. While the attribute feature vectors and graph construction represent the most computationally heavy operations, they are one time pre-processes performed at load time and are highly parallelizable. Our system relies on fast computations during the projection phase, and for this, the average response time is well below 0.01s, maintaining an interactive experience. All timings presented are performed on a 2.26 GHz Quad-Core Intel Xeon with 16GB memory.

Limitations When projecting a high dimensional space to lower dimensions, information is lost. While the projection facilitates the exploration of the vector field and its feature regions, poor selection of the control points and/or their locations complicates the identification of separate feature regions. Our framework requires user knowledge about the data and its feature types, as well as careful manipulation of the control points to distribute the projection in a way that is easy to distinguish features.

Despite these challenges, navigation within the 2D projection space is advantageous in that users focus on neighborhoods qualities rather than the numeric quantities associated with them. The use of pre-generated analytic feature neighborhoods provides an initial semi-automated feature classification that facilitate the exploration process. Additional paint tools may directly expose numerical measures of each attribute to support queries based on statistical information. Further,

Dataset	Size	Attribute	Graph Build	Component Reduction	Anchor Insertion	System Solve
	|V|	(min, median, max)	(min, median, max)	(min, median, max)	(min, median, max)	(min, median, max)
Combustion	40k	(0.142, **2.656**, 2.678)	(0.034, **0.232**, 0.296)	(0.011, **0.011**, 0.029)	(0.006, **0.059**, 0.136)	(0.000, **0.001**, 0.048)
Ocean	49.2k	(0.226, **0.2405**, 3.286)	(0.046, **0.1065**, 0.277)	(0.015, **0.016**, 0.091)	(0.005, **0.041**, 0.094)	(0.003, **0.004**, 0.098)
Magnet	100k	(0.335, **0.499**, 6.734)	(0.084, **0.194**, 0.365)	(0.027, **0.029**, 0.128)	(0.049, **0.087**, 0.188)	(0.004, **0.005**, 0.097)

Table 1. The system timings (in seconds) accumulated for operations performed during the exploration sessions described in Sec. 6 with dataset sizes given in terms of the number of vertices (|V|). While initialization of the graph structure is expensive, the interactivity of the system relies on fast performance for the projection method. Note that the timing variances recorded during the attribute space construction are reflective of the computational complexity associated with different attributes.

developers may enrich the attribute space to better differentiate new and specific feature types.

8 Conclusion

We introduce a system for the interactive exploration of vector data by adopting ideas from database retrieval techniques. The method maps the vector field points into an attribute space, where distance computations convey similarity between different points. In practice, we found that an attribute space with a relatively low dimensionality was sufficient in extracting the desired information; however, an arbitrary number of attributes can be used.

The discussed framework enables a flexible feature classification based method that is independent of the desired feature types and research domain. The user-driven identification process is enhanced by providing on-the-fly control point selection and texture painting. In our results, we illustrate the ability of our technique to not only extract different feature types from a dataset, but also, differentiate between varying types of the same feature structure. These abilities are demonstrated in the accompanying video.

The linear system solver used in our implementation efficiently evaluates the projection of the attribute points to the unit square. The projection provides uv-coordinates for the vector field points to enable texture mapped visualizations. The method scales to allow the exploration of large datasets while maintaining an interactive experience.

We propose a novel application building on information visualization based multi-dimensional scaling methods in conjunction with scientific visualization techniques to improve exploration of vector data. Our approach is noteworthy in that it supports a combination of idealized and data-specific features. We make use of efficient linear system libraries, threaded computations, and texture-based visualizations. This allows us to provide an interactive exploration experience that uniquely contributes to the field of feature-based visualization.

Future Work This research introduces many interesting extensions for future study, including augmentation of dye advection and LIC visualizations as well as the improving placement of seed points for streamline computations. Most immediately, however, we are exploring the extension of our techniques to volumetric vector fields and time varying datasets. Extension to volumetric datasets can make use of the existing framework, mapping the data samples to the high dimensional attribute space then projecting downward to the 2D canvas; however, it means that attribute neighborhood computations are adapted for the added dimension. The interactive tools are complicated by navigation of volumetric datasets, demanding the implementation of additional interface machinery [40]. Moreover, we are currently working on replacing the texture mapping scheme with volume rendering techniques. Lastly, we are investigating different methods of handling time varying data, weighing the benefits of adapting the graph construction versus considering temporal attribute neighborhood computations.

Acknowledgments We would like to thank the anonymous reviewers for their constructive comments. We thank Chems Touati for his time and help during the production of our accompanying video. This research has been funded by Fapesp-Brazil (#2008/03349-6), CNPq-NSF (#491034/2008-3), NSF (grants IIS-0905385, IIS-0844546, CNS-0855167, IIS-0746500, ATM-0835821, CNS-0751152, IIS-0713637, OCE-0424602, IIS-0534628, CNS-0514485, IIS-0513692, CNS-0524096), DoE, and IBM Faculty Awards.

References

[1] R. J. Adrian. Particle-imaging techniques for experimental fluid mechanics. *Annual Review of Fluid Mechanics*, 23:261–304, 1991.

[2] D. Banks and B. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[3] D. Bauer and R. Peikert. Vortex tracking in scale-space. *IEEE TCVG Symposium on Data Visualization*, 22:233–240, 2002.

[4] S. Bryson and C. Levit. The virtual windtunnel: an environment for the exploration of three-dimensional unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, pages 17–24, 1991.

[5] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. *IEEE Transactions on Visualization and Computer Graphics*, pages 156–163, 1991.

[6] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. *ACM SIGGRAPH*, pages 263–270, 1993.

[7] R. Cucitore, M. Quadrio, and A. Baron. On the effectiveness and limitations of local criteria for the identification of a vortex. *European Journal of Mechanics. B, Fluids*, 18(2):261–282, 1999.

[8] N. Cuntz, A. Kolb, R. Strzodka, and D. Weiskopf. Particle level set advection for the interactive visualization of unsteady 3d flow. *IEEE EuroGraphics Symposium on Visualization*, 27(3), 2008.

[9] T. Davis and W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Transactions Mathematics Software*, 35(4):1–23, 2009.

[10] A. Defant. *Physical Oceanography*. Pergamon Press, New York, 1961.

[11] M. H. Everts, H. Bekker, J. B. Roerdink, and T. Isenberg. Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15:1299–1306, 2009.

[12] N. I. Gould, J. A. Scott, and Y. Hu. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM Transactions on Mathematics Software*, 33(2):10, 2007.

[13] Z. Han and R. D. Reitz. Turbulence modelling of internal combustion engines using κ - ϵ models. *Combustion Science and Technology*, 106(4–6):267–295, 1995.

[14] E. Heiberg, T. Ebbers, L. Wigstrom, and M. Karlsson. Three-dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):313–319, July 2003.

[15] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flows. *IEEE Computer*, 22(8):27–36, 1981.

[16] J. P. Hultquist. Constructing stream surfaces in steady 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics*, pages 171–178, 1992.

[17] H. Janicke, M. Bottinger, and G. Scheuermann. Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1459–1466, 2008.

[18] J. Jeong and F. Hussain. On the identification of a vortex. *Fluid Mechanics*, 285:69–94, 1995.

[19] B. Jobard, G. Erlebacher, and Y. Hussaini. Hardware-accelerated texture advection for unsteady flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 6:155–162, 2000.

[20] D. Kenwright and R. Haimes. Vortex identification–applications in aerodynamics: a case study. *IEEE Transactions on Visualization and Computer Graphics*, 3:413–422, 1997.

[21] H. Krishnan, C. Garth, and K. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1267–1274, 2009.

[22] G.-S. Li, X. Tricoche, and C. Hansen. Physically-based dye advection for flow visualization. *Computer Graphics Forum*, 27(3):727–734, 2008.

[23] H. Li, W. Chen, and I.-F. Shen. Segmentation of discrete vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):289–300, 2006.

[24] K. Mahrous, J. Bennett, G. Scheuermann, B. Hamann, and K. I. Joy. Topological segmentation in three-dimensional vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):198–205, 2004.

[25] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 19–24, 1993.

[26] T. McLoughlin, R. Laramee, R. Peikert, F. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 2010.

[27] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow. *EuroGraphics*, 2009.

[28] N. Mölders and G. Kramm. Influence of wildfire induced land-cover changes on clouds and precipitation in interior alaska – a case study. *Atmospheric Research*, 84(2):142–168, 2007.

[29] H. Pagendarm, B. Henne, and M. Rutten. Detecting vortical phenomena in vector data by medium-scale correlation. *IEEE Transactions on Visualization and Computer Graphics*, 5:409–412, 1999.

[30] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

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

[32] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualization: Feature extraction and tracking. *EuroGraphics*, 22(4):1–17, 2003.

[33] S. Rogers, P. Buning, F. Merritt, and S. Follin. Distributed interactive graphics applications in computational fluid dynamics. *International Journal of High Performance Computing Applications*, 1(4):96–105, 1987.

[34] M. Roth and R. Peikert. A higher-order method for finding vortex core lines. *IEEE Transactions on Visualization and Computer Graphics*, 4:143–150, 1998.

[35] I. Sadarjoen and F. Post. Detection, quantification, and tracking of vortices using streamline geometry. *Computers and Graphics*, 24(3):333–341, 2000.

[36] I. Sadarjoen, F. Post, B. Ma, D. Banks, and H. Pagendarm. Selective visualization of vortices in hydrodynamic flows. *IEEE Transactions on Visualization and Computer Graphics*, 4:151–158, 1998.

[37] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[38] M. Schlemmer, M. Heringer, F. Morr, I. Hotz, M.-H. Bertram, C. Garth, W. Kollmann, B. Hamann, and H. Hagen. Moment invariants for the analysis of 2d flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007.

[39] H.-W. Shen, C. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. *Symposium on Volume Visualization*, 2:63–70, 1996.

[40] D. Speray and S. Kennon. Volume probes: interactive data exploration on arbitrary grids. *Workshop on Volume visualization*, pages 5–12, 1990.

[41] J. Tangelder and R. Veltkamp. A survey of content based 3d shape retrieval methods. *Shape Modeling International*, pages 145–156, 2004.

[42] H. Theisel and H.-P. Seidel. Feature flow fields. *IEEE Symposium on Data Visualisation*, pages 141–148, 2003.

[43] G. Turk and D. Banks. Image-guided streamline placement. *ACM SIGGRAPH*, pages 453–460, 1996.

[44] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. *IEEE Transactions on Visualization and Computer Graphics*, 6:163–170, 2000.

[45] R. Westermann, C. Johnson, and T. Ertl. Topology preserving smoothing of vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 7:222–229, 2001.

[46] K. Xu, H. Zhang, D. Cohen-Or, and Y. Xiong. Dynamic harmonic fields for surface processing. *Computer Graphics*, 33(3):391–398, 2009.