Exploring Vector Fields with Distribution-based Streamline Analysis

Kewei Lu, Abon Chaudhuri, Teng-Yok Lee, Han-Wei Shen
The Ohio State University*

Pak Chung Wong
Pacific Northwest National Laboratory†

ABSTRACT

Streamline-based techniques are designed based on the idea that properties of streamlines are indicative of features in the underlying field. In this paper, we show that statistical distributions of measurements along the trajectory of a streamline can be used as a robust and effective descriptor to measure the similarity between streamlines. With the distribution-based approach, we present a framework for interactive exploration of 3D vector fields with streamline query and clustering. Streamline queries allow us to rapidly identify streamlines that share similar geometric features to the target streamline. Streamline clustering allows us to group together streamlines of similar shapes. Based on user's selection, different clusters with different features at different levels of detail can be visualized to highlight features in 3D flow fields. We demonstrate the utility of our framework with simulation data sets of varying nature and size.

1 INTRODUCTION

Visualization and exploration of vector fields plays an important role in understanding the data generated from simulations in many science and engineering disciplines. Among the various methods that are currently in use, streamline based techniques continue to play a central role in vector field exploration. When dealing with very large scale vector fields, computing a large number of streamlines is often necessary, where different shapes and orientations of the streamlines represent different flow features in the underlying field. A direct visualization of all the densely computed streamlines, however, is seldom useful due to the difficulties caused by visual cluttering and occlusion. In addition, the user is often more interested in specific flow features, for example, the occurrences of a specific type of vortex at a certain scale. Hence, it is important to be able to classify the streamlines based on their similarity and display only those that are relevant to the user's objective.

Streamlines can be classified based on the similarity in their shapes. Previously, several streamline similarity measures have been developed. A majority of these methods treat streamlines as a points trace and the similarity between streamlines is computed by the distance between the points, for example, the *mean of closest point distance* [5] and the *hausdorff distance* [14]. Although these methods are convenient and easy to understand, they are sensitive to translation and rotation since their distance measures are based on the points' Euclidean distances. Given two streamlines, if they are translated or rotated individually, their distance will change. A different method to measure streamline distances is to calculate some geometric measures such as curvature or torsion along the streamline, and then combine these measures into a feature vector to describe the shape of the streamline. The similarity between streamlines is then measured through the distance between the feature vectors, for example, the edit distance [20, 21]. However, directly apply these similarity metrics usually bias to length. Streamlines that have a larger difference in length tends to have larger distance even though they may have similar shape. For example, two vortices with different sizes.

In this paper, we propose a novel idea that uses the distribution of feature measures over a streamline to be the descriptor, and use this descriptor to measure the similarity between streamlines. Our research is motivated by the following observation: The features of a streamline may vary widely along the curve, leading to a wide variation in the measured values (Figure 1(a)). Also due to noises in the data and sharp turns or twists at certain locations, sudden fluctuations in the values of the measure may occur on the streamline (Figure 1(b)). However, individual low or high values do not necessarily represent a feature (Figure 1(c)). Hence, dealing with the measured values at each point, which is the finest level of granularity, often is not robust enough. Besides, point-wise distance calculations between two streamlines can be expensive since a streamline can easily contain hundreds of sample points, and position-based distance metrics are not invariant to translation and rotation.

In our framework, a streamline is divided into segments based on the feature's distribution along it. For each segment, a 1D histogram is constructed to represent its feature distribution and these 1D histograms are concatenated to produce a 2D histogram to represent the entire streamline. Distances between the streamlines can be calculated by computing the distance between the 2D histograms. To match the segments between two streamlines that have different lengths or different numbers of segments, Dynamic Time Warping(DTW) is used. Using the distribution-based streamline distance metric has three major properties:

- Unlike point location-based distance metrics such as the *hausdorff distance* and the *mean of closest point distance*, the distributions of certain geometric properties on a streamline such as curvature and torsion are invariant to translation and rotation.
- Since our method is based on distributions, it does not need similar streamlines to have similar length. As long as they have similar distribution they will have small distance.
- Our method is much faster than other methods such as those that use the *hausdorff distance* and the *edit distance*. The timing performance of our method is summarized in section 5.

We demonstrate the utility of our approach with several examples of 3D flow field exploration. At run time, when the user finds a streamline with interesting feature, he can select this streamline and query for streamlines that have similar features. The similar streamlines will be extracted based on the distribution-based distance metric and rendered on the display window to highlight the feature of interest over the entire domain. Besides the interactive user-centric query, our system can create hierarchical streamline clusters for the user to explore particular flow features at different levels of detail. In all these applications, we show that the computation of streamline distance can be done very efficiently.

2 RELATED WORK

Streamlines have been studied as a major technique for flow visualization over the past decades. Quantifying the importance of a streamline in terms of representing flow features and measuring the similarity between streamlines in terms of shape and location

*e-mail:{luke,chaudhua,leeten,hwshen}@cse.ohio-state.edu
†e-mail:pak.wong@pnnl.gov

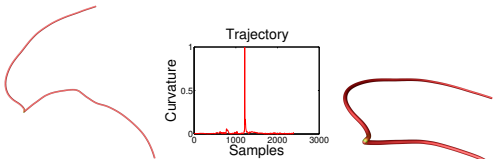

Figure 1: Limitation of point-based metric computation. (a) A streamline (color coded by curvature) with high range of curvature along the length. (b) The curvature values ordered along the length. (c) The zoomed in view of the point with highest curvature.

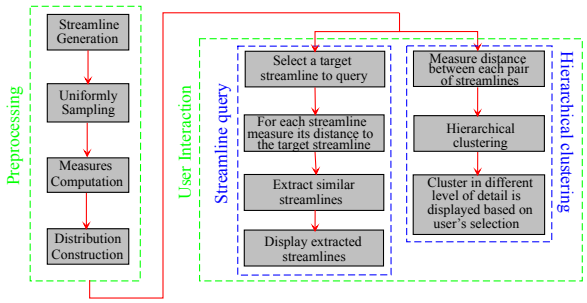

Figure 2: The major steps for our distribution-based flow analysis framework

are identified as two key problems. Moberts et al. [12] evaluated the role of four different similarity metrics, namely mean of closest points, closest points, hausdorff [5] and end points distance [3], in clustering DTI fibers, which are geometrically identical to streamlines. Pointwise Euclidean distance based similarity metrics [11] are widely used for clustering. Shi et al. [18] utilized the variation of different geometric properties of pathlines as a mean to classify them. Schlemmer et al. [16] presented an approach to extract and visualize flow patterns for 2D flow fields based on moment invariants. A more recent work [14] proposed Hausdorff distance as a similarity measure for streamlines and achieved meaningful clustering based on that. However, none of the above methods compute and utilize distributions from streamlines. Our proposed technique shows that distributions computed from streamlines can be effective as a distance measure, which, as opposed to some existing techniques, is less sensitive to location and orientation and faster to compute.

We are able to achieve streamline clustering and querying based on our proposed metric. Clustering of streamlines [14, 22] and equivalently, clustering of the underlying field [19, 7] can be achieved in different ways. Querying and selection of streamlines is not trivial mainly because there is no standard way to formulate such queries. A query-by-example scheme for streamlines is proposed by Wei et al. [21] which accepts shape templates from the user. Any form of query that can be converted to a distribution should be applicable to our proposed method.

3 Proposed Method

The main idea behind the proposed method is to represent each streamline as one or more distributions of geometric measures. These distributions are used as the feature signatures to efficiently compute the similarity between any pair of streamlines. Given a large number of streamlines, the distributions can be used to compute clusters in an unsupervised manner. We illustrate the main steps of our framework in Figure 2.

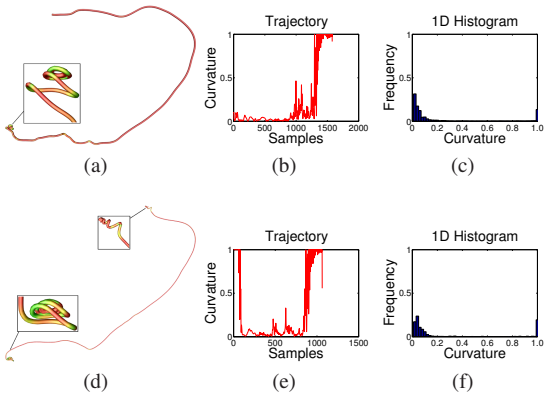

Figure 3: Limitation of the 1D histogram representation.

3.1 Construction of Distribution From Streamlines

Given a user-specified measure, statistical distributions can be used to describe its values and spread along the trajectory of a streamline. In this paper we demonstrate that several geometric measures such as curvature and torsion can be used to describe the feature of streamline. When the streamline is sampled discretely, different distributions will be produced with different degrees of sample density on the streamline. In our implementation, we do evenly-spaced sampling and the distribution is constructed from the measurements collected along the streamline using the same non-adaptive step size as the Runge-Kutta 4th order numerical integrator.

Assuming k measures are selected for analysis, we compute each of them at every sample point s_p along a streamline, resulting in a k-tuple measure $\{m_1^p, m_2^p, \ldots, m_k^p\}$ for each point. For each measure, we normalize the value to 0 and 1 based on the range of the sample values on all streamlines. We compute a distribution separately from each measure, each of which either can be represented *analytically* by a set of statistical properties such as measures of central tendency, moments and central moments that characterize the distribution in a compact way, or can be expressed *empirically* by a histogram, which is a discrete approximation of the underlying distribution. Analytical representations work well when the nature of the distribution is known. For example, a normal distribution can be represented by only its mean and standard deviation. However, the exact type of distribution for the geometric measures on a streamline is often unknown. In such cases, histogram based representation is more suitable, and hence is adopted in our method. Remember that we need to normalize the measurements to the range between 0 and 1. There is an issue when a few sample points having extreme values compared with other sample points, because the normalization will destroy the histogram. In order to address this issue, we clamp the top 5 percentage of measurements to 1 and then do normalization on the other 95 percentage of measurements in our implementation.

3.1.1 Histogram Construction

A histogram is a discrete representation of a probability distribution. Since the property of a streamline can vary from point to point, simply keeping a single 1D histogram may not be sufficient because the order of the measured values along the streamline can be essential for describing its feature and for comparing with other streamlines. For example, Figure 3(a) and 3(d) show two different streamlines color coded by curvature, one is more turbulent at the end and the other is turbulent at the beginning and the end. However, these two streamlines are undistinguishable by their 1D histograms since they are very similar, as shown in Figure 3(c) and 3(f).

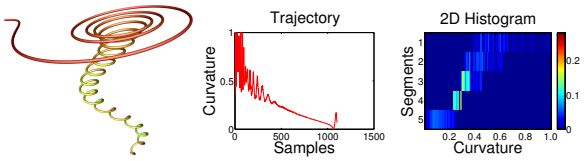

Figure 4: (a): The selected streamline color coded by curvature. (b): The curvature values ordered along the streamline. (c): The 2D histogram representation for this streamline, color represents frequency.

To resolve the ambiguity, we use 2D histograms instead of the simple 1D histograms to represent a streamline. Given a streamline, it is divided into segments. The goal of dividing a streamline into multiple segments is to preserve the order of feature measured along a streamline, albeit loosely, while still being able to use histograms to describe a streamline. With the segmentation, now the value stored at each sample point s_p, is no longer a scalar value, but a 2-tuple (x,k), where x denotes which segment this sample point is in and k represents the measurement. Then we can concatenate the 1D histograms of different segments in order, one in each row, and produce a 2D histogram representation for the entire streamline. The use of 2D histograms is a compromise between the use of point-based metrics and 1D histograms. At one extreme when every point represents a segment, the 2D histogram becomes a point-based metric. On the other extreme, if the entire streamline is treated as a segment, the 2D histogram becomes a 1D histogram. However, comparing with the point-based metric, computing the similarity based on 2D histograms is much faster and less sensitive to length, while comparing with the 1D histograms, 2D histograms preserve the order information.

An issue related to histogram construction is to decide the optimal number of bins (or the bin width) of each 1D histogram. In general, determining the optimal number of bins is non-trivial, and can be computationally expensive. However, there exist several empirical rules for this purpose. In our framework, we use Scott's choice [17] to decide the bin width for the 1D histogram within each segment of a 2D histogram. Scott's choice, decides the bin width based on the samples' standard deviation and the number of samples: $K = \frac{3.5\sigma}{N^{1/3}}$. where K is bin width, σ is the sample standard deviation, and N is the number of samples. Furthermore, we use the same number of bins for every segment, the reasons being:

- It speeds up the computation of distance between a pair of histograms, leading to interactive query processing and fast hierarchical clustering.
- It allows the user to compare the variances of features between segments by looking at the heatmap visualization of the 2D histogram.

In our implementation, the standard deviation σ is the average standard deviation of feature measures of all the segments and the sample size is the average sample points per segment: $K = \frac{3.5\sigma}{(\frac{N}{M})^{1/3}}$, where N is the total number of sample points, M is the total number of streamline segments.

An example of a 2D histogram constructed in this way is shown in Figure 4. Figure 4(c) shows the 2D histogram of curvature for the streamline shown in Figure 4(a). Based on the 2D histogram representation of the streamline, we can understand some underlying features about the streamline. For example, from Figure 4(c) the represented streamline has high curvature with high variation at the beginning, but it gradually decreases to low curvature. This can be observed from the curvature values ordered along the streamline, as shown in Figure 4(b). Since the 2D histogram incorporates the order information to some extent, it can better differentiate streamlines than simply using 1D histograms. The 2D histograms for the streamlines shown in Figure 3(a) and 3(d) are shown in Figure 5. Based on the two 2D histogram, it is easy to distinguish these two streamlines. The streamline shown in Figure 3(a) has high curvatures only at its end, while the one shown in Figure 3(d) has high curvatures at both the beginning and the end.

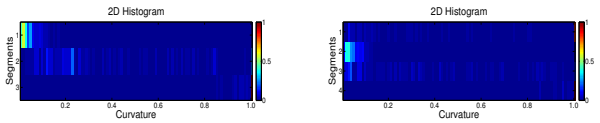

Figure 5: From left to right, the 2D histogram for the streamline in Figure 3(a) and (b)

3.2 Streamline Segmentation

As discussed in the previous section, a streamline needs to be divided into segments to preserve the order of features measured along the flow direction. In order to preserve the order information as much as possible, we do segmentation based on the difference between distributions of features in each possible segments, so that the features within each segment will be similar. To do this, given a streamline and a initial threshold ε, we choose a candidate split point where the difference in the distributions of the two streamline segments is the maximum. If the two distributions are bigger than the threshold ε, we split this streamline, increase the threshold by the initial threshold ε and continue to do so recursively on the resulting two halves. We terminate the segmentation process until the difference between two halves is smaller than ε or the streamline is too short to be segmented. The results of segmentation for two streamlines based on curvature are shown in Figures 6 and 7(a).

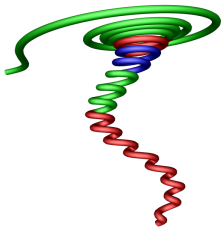

Figure 6: A segmentation result where segments are shown by colors.

Our streamline segmentation process requires two parameters: one is the minimum length of streamline segments and the other parameter is a threshold ε used to decide whether or not we should split a streamline segment. The first parameter makes sure that for each segment we have enough sample points to construct a meaningful distribution. In our implementation, we empirically set it to 100. The second parameter decides how many segments we have for a streamline. The smaller this parameter is, the more segments we have for a streamline and the more features order information is preserved. The larger this parameter is, the fewer segments we have and then less features order information is preserved. In Figure 7 (a)-(c) we show the segmentation results based on three different thresholds, and the corresponding 2D histograms are shown in Figure 7 (d)-(f). Our similarity metric is not very sensitive to the number of segments. First, slightly change the threshold will not change the segmentation results too much; second, once the threshold is changed, every streamline will be affected and new segments are produced, hence, comparisons between any two of them will still remain valid; third, the streamline might be over-segmented due to small threshold as shown in Figure 7(a). However, when measuring the similarity, as described in Section 3.3, we use Dynamic Time Warping [1] to map the segments in two streamlines, which can still map the redundant segments and is thus less sensitive to over-segmentation.

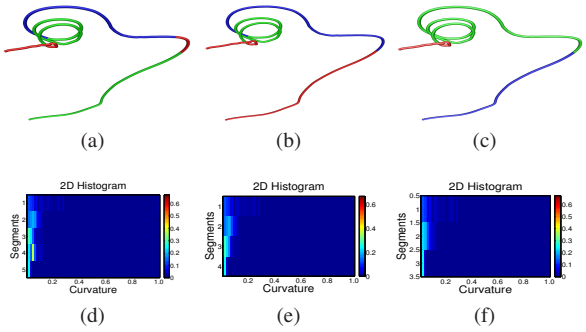

Figure 7: Segmentation results based on different threshold. From (a) to (c), the threshold is 0.1, 0.12 and 0.15. (d) to (f) show the corresponding 2D histograms.

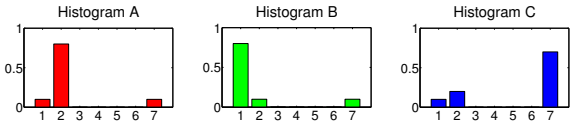

Figure 8: Three 1D histograms A, B and C for three different streamlines. H(A)=(0.1 0.8 0 0 0 0 0.1); H(B)=(0.8 0.1 0 0 0 0 0 0.1); H(C)=(0.1 0.2 0 0 0 0 0 0.7).

3.3 Histogram Similarity Measure

With the 2D histogram computed from each streamline, we can compute the streamline similarity based on the distance between the histograms. We consider two streamlines to be similar to each other in term of the user specified feature if they have similar feature distributions. Recall that each 2D histogram from a streamline consists of a sequence of 1D histograms, one from each streamline segment. Because different streamlines may have different numbers of segments, i.e., different rows in the 2D histograms, a mapping between the segments is needed before calculating the streamline similarity. In our framework, we use Dynamic Time Warping(DTW) [1] to find the mapping between the segments in two streamlines.

DTW is an algorithm based on dynamic programming that finds an optimal mapping between two sequences. In our framework, we treat each streamline as a sequence of streamline segments. For each pair of segments, one from each streamline, the similarity between them is calculated as the distance between the two distributions (described below). From the distance between any pair of segments in the two streamlines, DTW computes an optimal mapping between the two streamline sequences, where the cost is used as the distance between the streamlines.

There exist several standard methods to measure the distance between 1D histograms. Examples are L1 distance, Euclidean distance (L2), Kullback-Leibler distance(KL) [10], Bhattacharyya distance [2] and earth mover's distance(EMD) [15]. These distance measures can be classified into two categories: *bin-to-bin distance functions* and *cross-bin distance functions*. The bin-to-bin distance functions such as L1 distance, Euclidean distance, Kullback-Leibler distance and Bhattacharyya distance do not consider the correlation between non-corresponding bins when computing the distance between histograms. On the other hand, the cross-bin distance function such as the earth mover's distance considers the cross-bin relationship, albeit at higher computation cost.

In the context of our problem, considering cross-bin relationship is important. To show an example, assume that we have three 1D histograms for three streamlines as shown in Figure 8. Let A be the target histogram, the distance from B and C to A based on different distance functions is summarized in Table 1. It can be noted that

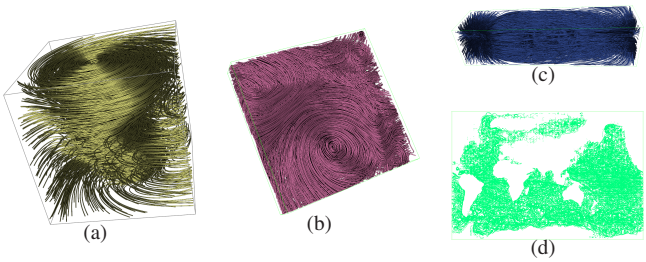

Figure 9: Visualization of the data sets Tornado, Hurricane Isabel, Solar Plume and Ocean. The number of testing streamlines are 1000, 2000, 2000, and 4800, respectively.

Table 1: Comparison of different distance measurement functions

	L1	Euclidean	KL	Bhattacharyya	EMD
D(A,B)	1.4	0.99	1.46	0.41	0.7
D(A,C)	1.2	0.85	0.91	0.27	3.0
Similar one	C	C	C	C	B

all the bin-to-bin distance functions choose C to be more similar to A than B. Only EMD declares B to be more similar to A. Now, if these three histograms were representing curvature distribution of three streamlines, both A and B would have segments dominated by low curvature and C would have segments dominated by higher curvature for a majority of points. This is why we consider Earth mover's distance to be more suitable for our framework and use it in our implementation.

Computation of EMD can be modeled as a classic optimization problem, known as supply-demand transportation problem, which tries to minimize the cost of converting one histogram into another by transporting units of mass from one to the other [15]. The transportation cost between two histograms P and Q can be expressed as:

$$EMD(P,Q) = \min_{\{F=f_{ij}\}} \frac{\sum_{i,j} f_{ij} d_{ij}}{\sum_{i,j} f_{ij}}$$

where d_{ij} is a pre-defined ground distance between supplier i and consumer j, and $F = \{f_{ij}\}$ is a set of flows which defines the amount of mass transferred from supplier i to consumer j.

Furthermore, since all the 1D histograms have cumulative histograms whose total areas will be the same, we can estimate EMD with much lower computation cost by the L1-distance between two 1D cumulative histograms [4]:

$$d(P,Q) = \sum_{i=0}^{n} |P_{cdf}(i) - Q_{cdf}(i)|$$

Given two streamlines $X = (x_1, x_2, ..., x_N)$ and $Y = (y_1, y_2, ..., y_M)$ where $x_1, ...x_N$ and $y_1, ..., y_M$ are the streamline segments. Suppose $V = (v_1, v_2, ..., v_L)$ where $v_l = (n_l, m_l) \in [1:N] \times [1:M]$ provides a mapping from X to Y. The distance between two 2D histogram H_1 and H_2 is:

$$Dist(H_1, H_2) = \min_{v} \left(\sum_{l=0}^{L} d(x_{n_l}, y_{m_l}) \right)$$

which is computed using the DTW algorithm.

4 Interactive Visualization Framework Design

In this section, we describe the use of our interactive visualization and analysis framework based on the distribution-based approach for two different visualization applications: similar streamline query and hierarchical clustering. Several 3D flow field data

sets are used, including Tornado, Hurricane Isabel, Solar Plume, and Ocean. Tornado is a $48 \times 48 \times 48$ synthetic data set. Hurricane Isabel is a data set with a resolution of $500 \times 500 \times 100$ that models a strong hurricane in the west Atlantic region in September 2003. Solar Plume was generated from a simulation of the solar plume on the surface of the sun with a resolution of $126 \times 126 \times 512$. The Ocean data set with a resolution of $3600 \times 2400 \times 40$ was produced by a high resolution eddy resolving simulation. We place a large number of seeds in the domain to generate streamlines, and collect distributions from the streamlines to analyze the flow features. Visualizations generated from these data sets are shown in Figure 9.

4.1 Feature Selection

The distribution-based approach presented in this paper can work with any suitable feature measure or a combination of measures based on the underlying application. For example, when the user is looking for some specific type of feature such as vortices, feature-specific metrics such as winding number [13], helicity [6] and $Lambda_2$ [8] can be used. When the user wants to identify streamlines with similar shapes, curvature and torsion can be used since according to the fundamental theorem of space curves, two space curves, streamlines in our case, defined on the same interval with the same torsion and non-zero curvatures can be translated into one another by application of an Euclidean transformation[9]. In this section, we use curvature, torsion, and curl as the measures to demonstrate our distribution-based approach.

4.1.1 Curvature

The curvature describes the rate of change in the tangent vector over a space curve with respect to the length of arc. Let T denote the unit tangent vector at a point on a streamline and S denotes the length of arc, the curvature κ is the magnitude of the rate of change of T: $\kappa = \| \frac{dT}{dS} \|$.

4.1.2 Curl

Given a vector field V, curl is denoted as $\nabla \times V$. Curl is a vector measure of the rotation at a point P in the vector field. The vector itself represents the axis of the rotation, and the magnitude of the vector denotes how fast it rotates. Only the magnitude of curl is used in our case since we are more interested in the tendency of circulation in a flow field. The curl of a point in vector field V can be defined as:

$$\nabla \times V = (\frac{\partial V_z}{\partial y} - \frac{\partial V_y}{\partial z})\mathbf{i} + (\frac{\partial V_x}{\partial z} - \frac{\partial V_z}{\partial x})\mathbf{j} + (\frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y})\mathbf{k}$$

4.1.3 Torsion

The torsion of a streamline measures the degree of twisting around its binormal vector. It also describes the rate of change of the streamline's osculating plane. If the torsion is zero, it means there is no twisting around the binormal vector, and the streamline completely lies on the osculating plane defined by the tangent and normal vectors. The torsion τ of a streamline is given by: $\tau = -N \cdot B'$; $B = T \times N$, where N, T and B are the normal, tangent and binormal vectors respectively and B' is the derivative of the unit binormal vector.

4.2 Similar Streamline Query

Visualizing three-dimensional streamlines can be difficult due to occlusion and visual cluttering. To explore a large number of pre-computed streamlines, users can request to display only streamlines that have a similar shape to that of the target streamline, for example, a streamline with a high degree of swirl that the user spots during exploration. To support this, we can perform a similarity query where all streamlines are compared with the target streamline. Since the user can frequently change the target streamline to look for different features, such distance comparisons must be done fast. In this section, we study the effect of using our distribution-based distance metric described in section 3.3 to perform streamline query. Performance results are provided in section 5.

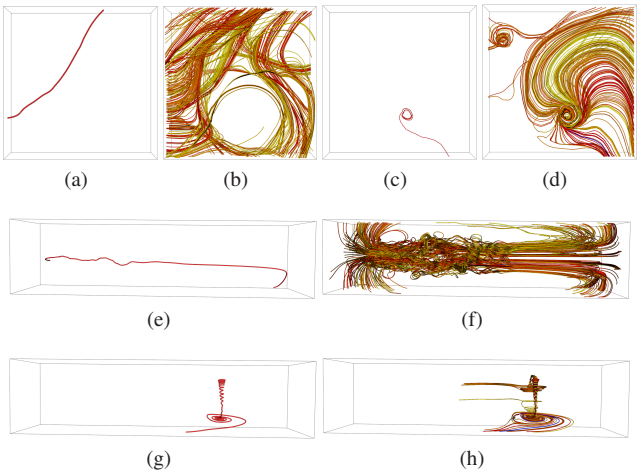

Figure 10: Four query results for Isabel data and Plume data. (a),(c),(e) and (g) show the four target streamlines. (b),(d),(f) and (h) show the query results for (a),(c),(e) and (g) respectively. In the results, streamlines are colored from red to yellow where red means more similar and the target streamline is in blue.

As mentioned earlier, curvature and torsion can be used to characterize space curves [9]. In our studies, we use a combination of curvature and torsion as the feature to query streamlines. To compute the distance between two given streamlines, we first compute the distance between their curvature histograms and the distance between their torsion histograms. Then we sum these two quantities to get the final distance. Figure 10 shows four different query results. In our first example, we use a streamline with low curvature and low torsion in the Isabel data set as the target streamline, shown in Figure 10(a). During the exploration, user can use a slide bar to show top N similar streamlines. In Figure 10(b), the top 400 most similar streamlines are shown, where red means more similar and yellow means less similar. For the second query, we use a streamline around the hurricane eye in Isabel as the target. The top 200 streamlines are selected and these streamlines are shown in Figure 10(d). The most similar streamlines are all around the hurricane eye and another swirl area in top left is detected. In the third query, the target is a straight streamline passing through the center of the Plume data as shown in figure 10(e). The top 200 similar streamlines are shown in Figure 10(f). A highly swirling streamline in Plume as shown in Figure 10(g) is selected as the target in the last query. The top 20 similar streamlines are selected and shown in Figure 10(h).

These four examples show the effectiveness of our query on streamlines with different natures such as straight, medium swirl and highly swirl. By using the combination of curvature and torsion, we are able to find streamlines with similar patterns as the target. In our system, the user can select any interesting streamline he observes to query during exploration, and the distances between the target and all other streamlines will be calculated. The user can use a slider bar to show the top N similar streamlines to explore the flow field. Our query can also be used to extract features such as swirl and vortex in a flow field. Figure 11 shows an example of using our query method to extract swirling flow by using curvature and torsion. The target streamline is a streamline with a highly swirl part as shown in Figure 11(a). The top 500 similar streamlines are displayed in Figure 11(b). The blue streamline is the target and another five big and one small swirling areas in the flow field are extracted.

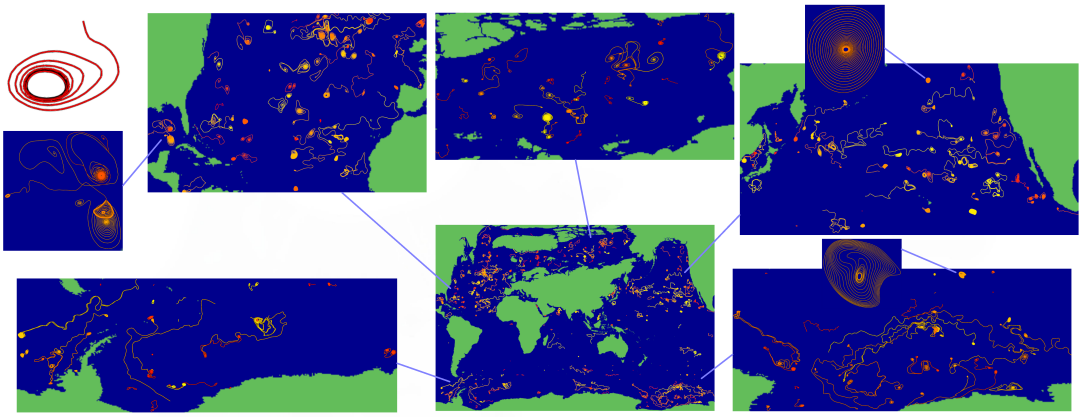

Figure 12: Effectiveness of our streamline query. The top left corner shows the target streamline which indicates an vortex. By querying similar streamlines, lots of vortices are found in the Ocean data set.

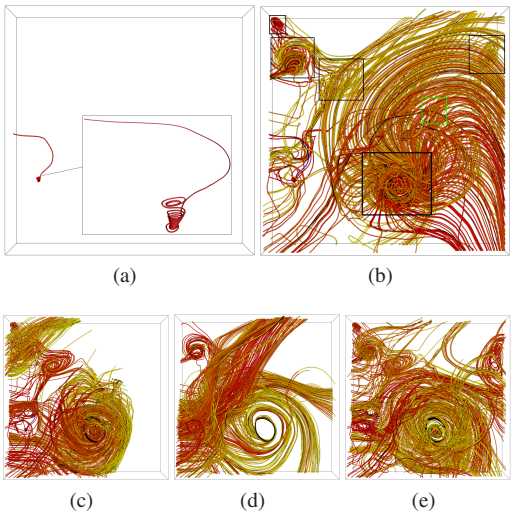

Figure 11: (a): The target streamline with a highly swirling part in the end. (b): The top 500 similar streamlines based on our distribution method. Five big swirling areas highlighted by black squares and one small swirling area highlighted by a green square are extracted. (c),(d) and (e) show the top 500 similar streamlines based on the *hausdorff*, *mean of closest point* and *edit distance*.

In Figure 11(c), 11(d) and 11(e), we show query results based on the *hausdorff*, *mean of closest point* and *edit distance* [21]. The point location based metrics such as the *hausdorff* and the *mean of closest point distance* tends to extract streamlines which are closed to the target streamline in space but not necessarily focus on the similarity between streamlines' features. The *edit distance* founds the similar features in the vector field as our method, but it is much slower than our method. The timing performance is reported in Table 3. Figure 12 shows another example using the Ocean data set.

4.3 Hierarchical Streamline Clustering

In addition to streamline query, another way to explore flow fields is through hierarchical streamline clustering. We use a bottom up agglomerative clustering method in our implementation. We begin with the bottom level where each cluster contains only one single streamline. Then at each step, we merge two clusters with a minimum distance until there is only one cluster left and a binary tree representation of the hierarchical cluster is generated. Different methods exist to calculate the distance between clusters such as complete-linkage clustering, single-linkage clustering and the mean distance between elements in each cluster. Complete-linkage clustering takes the maximum pair distance between elements in two clusters. Single-linkage clustering takes the minimum pair distance between elements in two clusters. The mean distance takes the average pair distance between elements in two clusters. Suppose P and Q are two clusters, these distances are defined as following:

- Complete-linkage: $max\{d(x,y) : x \in P, y \in Q\}$;
- Single-linkage: $min\{d(x,y) : x \in P, y \in Q\}$;
- mean distance: $\frac{1}{|A|\cdot|B|}\sum_{x\in Q}\sum_{y\in P} d(x,y)$.

Our experiments show that directly applying one of these three distance metrics to compute distances between clusters may produce unbalanced binary trees, in the sense that the sizes of clusters at a particular level may differ a lot. To avoid this problem, when computing the distance between two clusters, we add one more term to penalize a merge operation when it generates an unbalanced binary tree. The new formula to compute the distance between two clusters P and Q is:

$$dist(P,Q) = d(P,Q) + w\frac{s_P+s_Q}{S}$$

where the first term $d(P,Q)$ is the distance calculated by using one of the three methods discussed above and normalized to be 0 to 1. In the second term, s_P and s_Q are the numbers of streamlines in cluster P and Q and S is the total number of streamlines. w is the weighting parameter between 0 and 1. We call w the balance parameter. When w is larger, the smaller size clusters are forced to be merged early and the final clustering result will be more balanced. When w is smaller, small size clusters which correspond to small features in the data are more likely to be shown in the final clustering result. In our system, user can change w during the exploration to experiment clustering results with different degrees of balance.

To demonstrate the effect of hierarchical clustering based on our distance measures, we start with a simple 2D vector field. Curvature is used as the feature descriptor of streamlines in 2D. To evaluate our distribution-based approach, we also show the hierarchical clustering results using the *hausdorff distance*, *mean of closest point distance*, and *end point distance* respectively. Figure 13 shows the hierarchical clustering results. This example also demonstrates one advantage of our distance measure over other traditional point location based distance measures, that is, our method is invariant to translation and rotation of individual streamlines, and

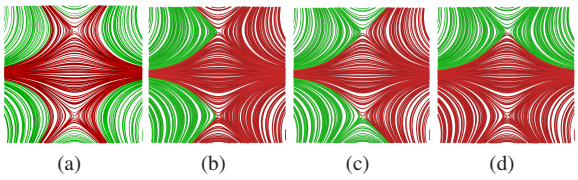

Figure 13: Hierarchical clustering results based on different distance metrics, from left to right: our method, *hausdorff*, *mean of closest point* and *end point distance*. The balance parameter $w = 0.5$. Different clusters are assigned a different color.

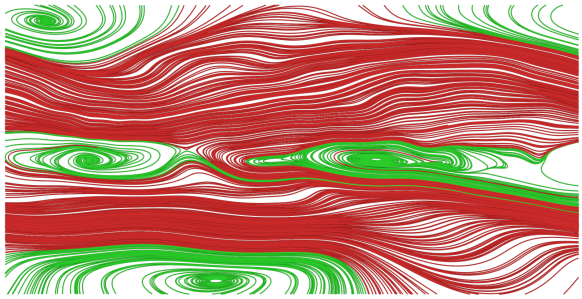

Figure 14: Clustering results based on curvature distribution. The balance parameter $w = 0.7$. The green cluster corresponds the vortex flow and the red one corresponds to straight flow.

Table 2: Timings for Streamline Computation and Feature Evaluation

Data Set	Avg. Len.	Advect. (sec)	Feature Comp. (sec)			Seg. (sec)
			Curv.	Curl	Tors.	
Tornado	725	5	4	3	38	55
Isabel	2087	36	21	18	61	232
Plume	2211	33	24	22	153	707
Ocean	2919	60	79	67	596	3495

Table 3: Timings for Streamline Query

Data Set	# Bins		Timing(sec)				
	Curv.	Tors.	Hist.	Ours	d_h	d_m	d_e
Isabel	43	20	0.28	0.02	281	287	270
Plume	18	8	0.30	0.02	340	351	307
Ocean	12	1	0.96	0.03	561	575	380

Table 4: Timings for Hierarchical Streamline Clustering

Data Set	# Bins			Timing(sec)		
	Curv.	Tors.	Curl	Hist.	Dist. Matrix	Clust.
Tornado	N/A	N/A	7	0.06	1.62	2.92
Plume	18	8	N/A	0.31	12.42	22.39

mainly focused on the intrinsic nature of shape. Using other metrics, streamlines with similar shapes but different orientations and far from each other will not be grouped into the same cluster, for example, the streamlines in the four corners of the data in the image.

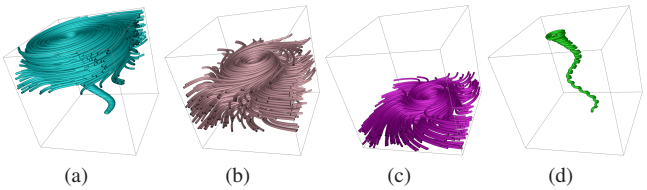

Figure 15: Hierarchical streamline clustering results for Tornado data set based on the magnitude of curl. The balance parameter $w = 0.0$. The whole set of streamlines are cut into four parts with different swirling intensity.

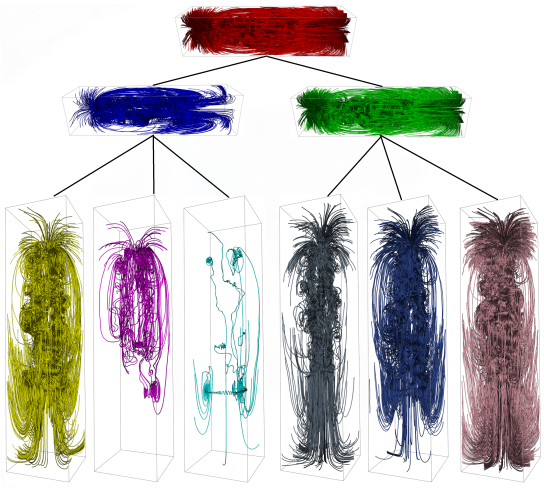

Figure 16: Hierarchical streamline clustering results for Plume data set based on the combination of curvature and torsion. The balance parameter $w = 0.7$. The whole set of streamlines are cut into different parts based on their shape difference.

In Figure 14 we show the hierarchical clustering result for another 2D vector field based on curvature distribution. In this example, vortex and straight flow are separated and assigned to two different clusters since they have different curvature distribution.

We also perform experiments on the Tornado data set. The measure used is the magnitude of curl which represents how fast the flow rotates in the field. Based on this measure, tornado can be separated to different parts with different degrees of rotation. Figure 15 shows four clusters. We set the balance parameter w to be 0 during the exploration, so that a small cluster corresponding to the region around the tornado center is extracted as shown in Figure 15(d).

Besides the 2D flow fields and the simple synthetic 3D flow field, we also tried our hierarchical clustering method on the Plume data set, a more complex 3D flow field. The combination of curvature and torsion is used to cluster the streamlines. The result is shown in Figure 16. The whole set of streamlines are grouped into several clusters and each cluster has streamlines with similar shapes. By displaying each cluster separately, the user is able to inspect different part of the flow field and understand the data more easily.

5 Performance

Our streamline exploration framework consists of two main stages. The first stage is preprocessing, and can be further divided into four steps: streamline generation, segmentation, feature evaluation, and histogram construction. Currently, we evaluate the three geometric features: curvature, curl and torsion for every sample point. The second stage of the framework is interactive visualization, and it includes similar streamline query and hierarchical streamline cluster-

ing. Streamline computation, segmentation and feature evaluation were done on a Linux server with an Intel Xeon CPU and 24 GB of memory. Histogram construction and interactive visualization were performed on a desktop computer with an Intel(R) Core(TM) i7-2600 CPU 3.4GHz processor with 16GB memory. The performance numbers for streamline computation, segmentation and feature measurements are summarized in Table 2.

In Table 3, we summarize the timing of streamline query for Hurricane Isabel(Figure 11), Plume(Figure 10(g)) and Ocean(Figure 12) data sets. All the queries used a combination of curvature and torsion. The computational cost for our streamline query depends on the number of streamlines, the number of segments, and the number of bins for the histogram in each segment. We compare the performance of our distribution-based method with other distance measures such as the *hausdorff distance* d_h, the *mean of closest point distance* d_m and the *edit distance* based on curvature and torsion d_e. It can be seen that our method outperformed the other methods by several times in terms of speed. This is important for the users to get a rapid feedback for their queries.

Table 4 shows the timings for hierarchical streamline clustering based on our method for Tornado(Figure 15) and Plume(Figure 16). All timings were measured in seconds. For Tornado, we use the curl to separate streamlines based on the degree of rotation. For Plume, we use a combination of curvature and torsion. We did not measure the performance of hierarchical clustering based on other distance measures such as the *hausdorff distance* since the computational time would be prohibitively long as we can see from the query results.

6 Conclusion And Future Work

In this paper, we propose a new method to measure the distance between streamlines based on the statistical distributions of geometric measures along the trajectories of streamlines. Compared to some existing methods, our method is invariant to translation and rotation, and we can evaluate the distance between streamlines much faster. Based on our distance metric, we design a framework to interactively explore 3D vector fields through query and clustering.

Our framework can be extended along several directions. First, we will extend our framework to handle time-varying vector data, so that query and clustering can be done for pathlines or streaklines. Second, besides the three geometric measures, curvature, curl and torsion, additional measures including domain-specific physical quantities can be included to explore 3D vector fields. We believe that combining different feature measures can produce more robust query and clustering results, and reveal different features in the vector field. Third, in order not to miss any important features of the flow and make the query more efficiency, our method can be combined with feature-driven streamline placement. Fourth, the user interface could be further improved. For example, instead of showing a given number of similar streamlines, the user can choose to show the streamlines whose distance to the target streamline is within a threshold, also we could provide some heuristic such as 1D histograms to assist the user to select target streamlines. Fifth, in the hierarchical clustering, we use a balance parameter and the size of clusters to control the clustering results. In the future, other metrics will be investigated to control the clustering results. Furthermore, we will investigate the effect of constructing 2D histograms with different bin size for different streamlines and study how it influences the results and performance.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. The data sets used in the paper are courtesy of Roger Crawfis (Tornado), Mathew Maltrud (Ocean), John Clyne (Plume) and W. Wang, C. Bruyere, B. Kuo, and others (Isabel). This work was supported in part by NSF grant IIS-1017635, US Department of Energy DOE-SC0005036, Battelle Contract No. 137365, and Department of Energy SciDAC grant DE-FC02-06ER25779, program manager Lucy Nowell.

References

[1] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD Workshop*, pages 359–370, 1994.

[2] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[3] A. Brun, H.-J. Park, H. Knutsson, and C.-F. Westin. Coloring of dt-mri fiber traces using laplacian eigenmaps. In *EUROCAST '03: Computer Aided Systems Theory*, pages 518–529, 2003.

[4] S. Cohen and L. Guibas. The earth mover's distance under transformation sets. In *ICCV '99: Proceedings of the International Conference on Computer Vision*, volume 2, pages 1076–, 1999.

[5] I. Corouge, S. Gouttard, and G. Gerig. Towards a shape model of white matter fiber bundles using diffusion tensor mri. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro*, pages 344 – 347 Vol. 1, 2004.

[6] D. Degani, A. Seginer, and Y. Levy. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28(8):1347–1352, 1990.

[7] B. Heckel, G. Weber, B. Hamann, and K. Joy. Construction of vector field hierarchies. In *Vis '99: Proceedings of IEEE Visualization*, pages 19 –25, 505, 1999.

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

[9] E. Kreyszig. *Differential geometry*. Dover Publications, 1991.

[10] S. Kullback. *Information Theory and Statistics (Dover Books on Mathematics)*. Dover Publications, 1997.

[11] M. Maddah, W. E. L. Grimson, S. K. Warfield, and W. M. Wells. A unified framework for clustering and quantitative analysis of white matter fiber tracts. *Medical Image Analysis*, 12(2):191 – 202, 2008.

[12] B. Moberts, A. Vilanova, and J. van Wijk. Evaluation of fiber clustering methods for diffusion tensor imaging. In *Vis '05: Proceedings of IEEE Visualization*, pages 65 – 72, 2005.

[13] L. M. Portela. *Identification and Characterization of Vortices in the Turbulent Boundary Layer. Volume I*. PhD thesis, Stanford University, 1997.

[14] C. Rossl and H. Theisel. Streamline embedding for 3d vector field exploration. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):407 –420, 2012.

[15] Y. Rubner, C. Tomasi, and L. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the International Conference on Computer Vision*, pages 59 –66, 1998.

[16] 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.

[17] D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):pp. 605–610, 1979.

[18] K. Shi, H. Theisel, H.-C. Hege, and H.-P. Seidel. Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3d timedependent flow fields. In *Proceedings of Topology-Based Methods in Visualization*, 2007.

[19] A. Telea and J. Van Wijk. Simplified representation of vector fields. In *Vis '99: Proceedings of IEEE Visualization*, pages 35 –507, 1999.

[20] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[21] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *PacificVis '10: Proceedings of IEEE Pacific Visualization Symposium*, pages 129 –136, 2010.

[22] H. Yu, C. Wang, C. Shene, and J. Chen. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1353–1367, 2012.