# Streamline Similarity Analysis Using Bag-of-Features

Yifei Li[a], Chaoli Wang[a], and Ching-Kuang Shene[a]

[a]Department of Computer Science, Michigan Technological University, Houghton, MI 49931

## ABSTRACT

Streamline similarity comparison has become an active research topic recently. We present a novel streamline similarity comparison method inspired by the bag-of-features idea from computer vision. Our approach computes a feature vector, spatially sensitive bag-of-features, for each streamline as its signature. This feature vector not only encodes the statistical distribution of combined features (e.g., curvature and torsion), it also contains the information on the spatial relationship among different features. This allows us to measure the similarity between two streamlines in an efficient and accurate way: the similarity between two streamlines is defined as the weighted Manhattan distance between their feature vectors. Compared with previous distribution based streamline similarity metrics, our method is easier to understand and implement, yet producing even better results. We demonstrate the utility of our approach by considering two common tasks in flow field exploration: streamline similarity query and streamline clustering.

**Keywords:** flow visualization, streamline similarity, streamline clustering, bag-of-features

## 1. INTRODUCTION

For more than two decades, flow visualization has been a central topic in scientific visualization and a variety of techniques, including glyph-based, texture-based, integration-based, topology-based, and partition-based techniques have been presented. In this paper, we focus on integration-based flow visualization as it is most widely used in practice. It works by placing particles or seeds in a vector field and advecting them over time. The traces or *field-lines* that the particles follow, i.e., *streamlines* for steady flow and *pathlines* for unsteady flow, depict the underlying vector data. Visual clutter and occlusion is a major issue when thousands of streamlines are rendered to depict a flow field. It makes it difficult for users to explore the interesting underlying features. For example, it is often the case that users want to see all the streamlines of some specific shape (e.g., spirals). Therefore, it is important to be able to categorize all the streamlines based on shape similarity such that different features can be explored easily.

Previously, most methods measure the similarity between two streamlines by computing the distance between their points. These point-wise distance based similarity metric include closest point distance, mean of closes point distance,[1] Hausdorff distance,[2] and end point distance. A major issue with these metrics is that they are not invariant to rotation, translation and scaling. In other words, the similarity between two streamlines may become different after we change their orientations, positions or sizes. For example, when similar streamlines need to be grouped together, the point-wise distance based similarity metric fail to group similar streamlines which are far from each other. Recently, similarity measures based on feature distribution were proposed to overcome this issue, and our work falls into this category.

Inspired by the idea of *bag-of-features* from computer vision, we first construct *spatially sensitive bag-of-features* (i.e., feature vector) for each streamline, and then streamline similarity can be calculated in terms of the distance among feature vectors. A bag-of-features can be considered as a distribution of various features. The major problem with distribution based methods is that two different streamlines may have similar feature distribution. For example, in Figure 1, the curve on the right is made by cutting the left curve into pieces and shuffling the pieces, therefore, they have very similar curvature distribution. However, these two curves do not look similar to each at all. The spatially sensitive bag-of-features described later in this paper not only reflects

Further author information:(Send correspondence to Yifei Li)
Yifei Li: E-mail: yifli@mtu.edu
Chaoli Wang: E-mail: chaoliw@mtu.edu
Ching-Kuang Shene: E-mail: shene@mtu.edu

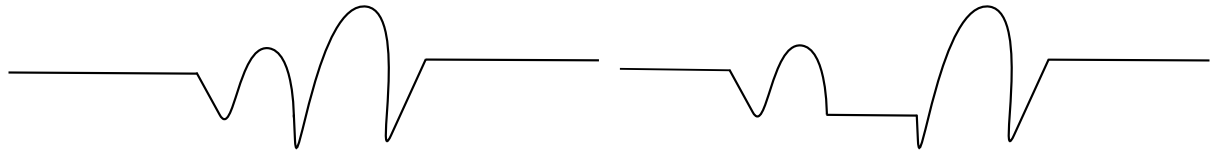



Figure 1. The curve in (b) is obtained by rearranging different segments of the curve in (a). Although they look different, their curvature distributions are almost similar due to the way they are formed.

feature distribution, but also encodes spatial relations among features. We demonstrate the effectiveness of our approach by considering two scenarios: (1) streamline query - find streamlines similar to a query streamline from a flow field (2) streamline clustering - group similar streamlines together.

To summarize, the main contribution of this paper is as follows:

- We proposed a novel approach for comparing streamline similarity. In this approach, each streamline is described by a feature vector which is invariant to affine transformation and encodes spatial relations among features.
- Our algorithm is straightforward to implement, which only requires summing up the contributions of different "expressions" (Section 3.1).
- Our approach is general enough such that new features can be easily integrated into the framework.

## 2. RELATED WORK

### 2.1 Bag of Features

Feature-based methods for comparing image similarity are widely used in computer vision. The construction of feature-based representation of an image usually consists of two steps: *feature detection* and *feature description*. During feature detection, a set of feature points which carry a significant amount of information is picked. For example, SIFT[3] consider the local extrema of the discrete image Laplacian at different scales, and define feature points to be the local extrema which appear at multiple scales. During feature description, SIFT assigns a 128-dimensional descriptor vector constructed as local histograms of image gradient orientations around each feature point. After these two steps, similar images can be found by matching their feature descriptors.

In order to reduce the representation size, *vocabulary* can be constructed by performing vector quantization in descriptor space. Descriptors can be replaced by the indices in vocabulary representing visual "words". Aggregating all the indices into a histogram by counting the frequency of appearance of each word, the *bag of features* is constructed. This way, two images can be compared by their bags of features. Common techniques to measure the difference between two bags of features include Euclidean distance or weighted Euclidean distance. Intuitively, the weights can be chosen such that frequent words play a less important role than infrequent ones. For instance, if we try to compare the similarity between two documents on streamline visualization, the difference between the number of times "the" appear in each document should be less weighted than "vortex", even though "the" appear more frequently than "vortex". It was shown by that weighted distance is better than non-weighted distance.[4]

### 2.2 Streamline Similarity Measure

Many existing works on streamline similarity measure and fiber bundle clustering are based on point-wise distances between streamlines or fibers. For streamlines, Chen et al.[5] compared randomly-seeded candidate streamlines based on their Euclidean distance as well as their shape and orientation. This similarity-guided approach produces streamlines that accentuate regions of interest without explicit feature detection and extraction. In DTI fiber bundle clustering, the proximity of two fibers can be defined as the distance between two selected points with one from each fiber.[6] Examples include the closest point measure, the Hausdorff distance, and the

Fréchet distance. The proximity of two fibers can also be measured using the mean distance of all points of the fibers along their arc length. Examples include the average of point-by-point distances between corresponding pairs, the mean of closest point distances,[7] the thresholded average distance,[8] the weighted normalized sum of minimum distance,[9] and the distance measured in a transformed feature space. Moberts et al.[10] evaluated different combinations of clustering methods (single-link, complete-link, weighted average, and shared nearest neighbor) and similarity measures (closest point distance, mean of closest point distances, Hausdorff distance, and end point distance) and reported that the use of hierarchical single-link clustering combined with the mean of closest point distances gives the best results.

The most significant issue with point-wise distance based similarity measures is that similar streamlines far away from each other will not be considered as similar. In order to overcome this problem, similarity measures based on feature distribution were proposed recently. McLoughlin et al.[11] computed streamline signatures based on curvature, torsion and tortuosity and then performed similarity comparisons using chi-squared test on those signatures. However, as pointed out by Lu et al.,[12] two dissimilar streamlines may have similar feature distributions. Therefore, Lu et al.[12] proposed to take into account the spatial relations among features. In order to do so, they first segmented each streamline and computed a 1D histogram for each segment. The distance between two streamlines is then computed by finding a mapping between two sets of segments such that the coal distance between pairs of segments is minimized using Dynamic Time Warping (DTW).[13]

Lu et al.[12] recursively segmented each streamline until either a segment is too short or a segment cannot be split into two segments which are dissimilar enough. In order to determine whether two resulting segments after the split are similar or not, they compared their feature distribution difference against some threshold. However, this sounds like a chicken-and-egg problem because streamline similarity comparison is exactly the problem they tried to solve. They used an approximate version of earth mover's distance (EMD)[14] to measure the similarity between two streamline segments in order to reduce the computation cost. Their approach also requires the tuning of two parameters in order to determine when the segmentation should stop, although they claimed their similarity metric is not very sensitive to over-segmentation. In comparison, our approach describes each streamline using a feature descriptor which already encodes the spatial relations among features. Therefore, the steps of segmentation and finding a mapping between two sets of segments are unnecessary. This makes our algorithm easier to understand and implement.

Finally, Wei et al.[15] constructed a feature vector for a streamline by concatenating the feature metrics at sampled points. However, the task of sampling a space curve is difficult because important points may be missed. Wei et al.[15] solved this problem by using accumulated curvature. This resulted in variable-sized feature vectors for different streamlines. While this approach served their purpose well because they actually focused on matching part of a streamline against some user-sketched pattern, we are not able to adopt their method for similarity comparison between two streamlines. Even if two streamlines are described with their feature vectors of the same length, the direction matters when it comes to compute accumulated curvature. For example, two identical streamlines may considered to be different if curvature is accumulated from different end points.

## 3. PROPOSED METHOD

Our approach is inspired by the recent success of using bag of features to retrieve similar 3D models to a query model.[16] In the following, we first discuss how spatial relations among features are encoded in the bag of features of each streamline in Section 3.1. Next, in Section 3.2, we explain the specific features we choose to include in bags of features.

### 3.1 Spatially Sensitive Bag-of-Features

Given feature descriptors (e.g., tuples containing curvature and torsion) computed at points on streamlines, the first step is to quantize the descriptor space in order to obtain a compact representation called vocabulary. A vocabulary $P = \{p_1, p_2, \cdots, p_V\}$ of size $V$ is a set of representative vectors in the descriptor space. The dimension of each vector is the number of features we compute at each point on a streamline. For example, the dimension of each representative vector is two if only curvature and torsion are computed at each point. In our implementation, the quantization is done using the efficient k-means algorithm by Kanungo et al.,[17] and the size of the vocabulary is set empirically to 16. We found that increasing the size of vocabulary not necessarily

improves the results because we would get many duplicates in the vocabulary due to a small number of features (Section 3.2) we used. Each $p_i$ in the vocabulary is also called a "word".

Given a vocabulary, a feature distribution for a streamline point $p$ is the following $V \times 1$ vector:

$$\theta(x) = (\theta_1(x), \theta_2(x), \cdots, \theta_V(x))^T \tag{1}$$

where $\theta_i(x) = 1$ and $i$ is the index of the word in the vocabulary which best describes the features at point $x$, and $\theta_j(x) = 0$ for $j \neq i$. To obtain the bag-of-features for a streamline, we simply sum up the feature distributions over the entire streamline:

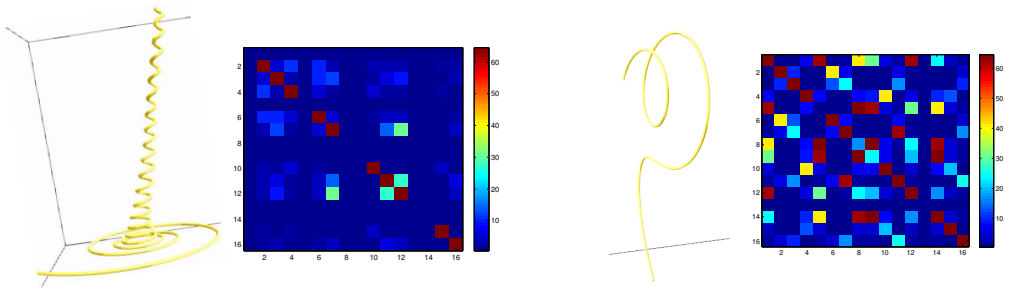
$$\text{BoF}(X) = \sum_{x \in X} \theta(x) \tag{2}$$

The similarity between two streamlines can be defined as the distance between their bags-of-features in $\mathbb{R}^V$.

However, the above definition of bag-of-features only considers the distribution of words in the vocabulary and loses the spatial relationship among them. The spatially sensitive bag-of-features considers combination of words, also called expression, to address this problem. It is defined as follows:

$$\text{SS-BoF}(X) = \sum_{x \in X} \sum_{y \in X \wedge y \neq x} \theta(x)\theta^T(y) \frac{1}{l/d(x,y)} \tag{3}$$

where $l$ is the total length of a streamline and $d(x, y)$ is the arc-length between two points $x$ and $y$ on that streamline. The resulting spatially sensitive bag-of-features is a $V \times V$ symmetric matrix which represents the frequency of expressions. Intuitively, two words close to each other can be considered as an expression, and thus have a large contribution to the frequency of the expression. On the contrary, two words far from each other are less likely to form an expression, thus making a smaller contribution to the frequency of the expression. The spatially sensitive bag-of-features can be easily represented by a $V(1+V)/2$ dimension vector by converting the upper/lower triangular matrix into a vector. Notice that we use $l/d(x, y)$ instead of just $d(x, y)$ as the denominator to offset scaling effects. Otherwise, a scaled version of a streamline will have a very different spatially sensitive bag-of-features than the original streamline since the values in the matrix will also be scaled. Figure 2 shows two streamlines along with the symmetric matrices representing their spatially sensitive bag-of-features. As we can see, the two symmetric matrices indeed look different for the two streamlines of dissimilar shapes.



(a) (b)

Figure 2. Two different streamlines along with the symmetric matrices representing their spatially sensitive bag-of-features.

Frequent expressions are less likely to be informative. Down-weighting common expressions is an effective technique used in text search engines, and has been successfully used in object retrieval in video[4] and three-dimensional shape retrieval.[16] Similarly, we define *inverse document frequency* of expression $i$ as the logarithm of the inverse fraction of streamlines in a flow field in which this expression appears:

$$w_i = \log\left(\frac{D}{\sum_{j=1}^{D} \delta(f_i(X_j) > 0)}\right) \tag{4}$$

where $D$ is the total number of streamlines, $\delta$ an indicator function, and $f_i(X_j)$ counts the number of occurrences of expression $i$ in streamline $j$. The smaller $w_i$ is, the more common the expression $i$ is in all the streamlines, and so it is less likely to be able to discriminate between streamlines.

Finally, we use the weighted Manhattan distance to measure the similarity between two streamlines:

$$d(X,Y) = \sum_{i=1}^{V(1+V)/2} w_i |X_i - Y_i| \tag{5}$$

where $X$ and $Y$ refers to the spatially sensitive bag-of-features of the two streamlines.

### 3.2 Streamline Feature Selection

Spatially sensitive bag-of-features allows us to consider the combination of any suitable feature metrics. Since we are interested in finding out streamlines with similar shapes, the first two metrics we consider are curvature and torsion. According to the fundamental theory of curves,[18] any regular curve has its shape completely determined by its curvature and torsion. Curvature and torsion are examples of *local* geometric properties. Our experiment showed that similarity comparison results are greatly improved by considering the other two *global* geometric properties: tortuosity and velocity direction entropy. Tortuosity was first proposed by McLoughlin et al.,[11] which has a low computation cost.

#### 3.2.1 Curvature

Curvature is the rate of change in the tangent vector with respect to arc-length. The curvature at a point $x$ can be computed using the following formula as described by Farin:[19]

$$\kappa(x) = \frac{\|\dot{L(x)} \times \ddot{L(x)}\|}{\|\dot{L(x)}\|^3} \tag{6}$$

where $L(x)$ is the position of a point $x$ on a curve, $\dot{L(x)}$ (i.e., velocity) and $\ddot{L(x)}$ are the first and the second derivative of $L(x)$.

Since the velocity vector information is only available at each grid point, we first compute the curvature at each grid point and then use tri-linear interpolation to derive the curvature value at an arbitrary point. The second derivative of $L(x)$ is approximated using central difference:

$$\ddot{L(x)} = \frac{\dot{L(x+h)} - \dot{L(x-h)}}{2h} \tag{7}$$

where $h$ is one which the grid resolution.

#### 3.2.2 Torsion

Torsion describes how much a curve squirms out of its osculating plane. It can be computed as follows:[19]

$$\tau(x) = \frac{\det[\dot{L(x)}, \ddot{L(x)}, \dddot{L(x)}]}{\|\dot{L(x)} \times \ddot{L(x)}\|^2} \tag{8}$$

where det means the determinant of a matrix.

Similar to curvature computation, we compute the torsion at each grid point first and then use tri-linear interpolation to calculate the torsion at an arbitrary point.

### 3.2.3 Tortuosity

Intuitively, tortuosity measures how much a curve deviates from a straight line. It is calculated as the ratio of the length of curve compared to the shortest distance between its start and end points. The length of a curve can be computed by summing up the length of straight line segments consisting of the curve. The shortest distance between a curve's start and end points is the Euclidean distance between the two points. For example, a straight line has the lowest tortuosity value of one. For each point on a streamline, we precompute the arc-length $s(x)$ between this point and the streamline's start point, the tortuosity at a point $x$ is then defined as:

$$\text{tortuosity}(x) = \frac{s(x)}{||p(x) - p(0)||} \tag{9}$$

where $p(0)$ is the start point position.

### 3.2.4 Velocity Direction Entropy

Since entropy is a measure of randomness, velocity direction entropy is a good indicator of turbulence. For each point $x$ on a streamline, we consider a small neighborhood $N$ around that point and compute velocity direction entropy for that region. In our experiment, we empirically choose $N$ such that it includes ten points on each side of point $x$. Another possibility is to calculate several entropy values using different sizes of neighborhood. In order to compute the entropy, we first partition a unit sphere into 50 regions with equal area using an algorithm by Leopardi.[20] In other words, we quantize the three-dimensional vector space into 50 bins. Leopardi[20] also gave an algorithm on how to find out which bin an arbitrary vector falls into. This way, velocity direction entropy can be easily computed using the formula:

$$\text{entropy}(x) = -\sum_{i=1}^{50} p_i \log(p_i) \tag{10}$$

where $p_i$ is the fraction of the velocity vectors in the neighborhood $N$ which fall into bin $i$.

## 4. RESULTS AND DISCUSSION

We used the seven flow data sets listed in Table 1 for experiment. The five critical points data set is a synthesized flow field consisting of two spirals, two saddles and one source. The tornado data set is from a simulation of a tornado event. The supernova data set is from a simulation of the explosion of stars. The car flow data set is from the simulation of the air flow around a car. The crayfish data set is from a simulation of the heat flow around a cooking crayfish. The solar plume data set is from a simulation of down-flowing solar plumes for studying the heat, momentum and magnetic field of the sun. Finally, the computer room data set is from a simulation of air flows inside a computer room. In the following, we present the machine configuration and timing results, followed by streamline similarity query and and streamline clustering results. We refer readers to the accompanying video for the best evaluation of our approach.

| data set | dimension | initial # lines | average # points per line | feature evaluation time | ss-bof evaluation time |
|---|---|---|---|---|---|
| five critical pts | $51 \times 51 \times 51$ | 500 | 60 | 0.372s | 0.245s |
| tornado | $64 \times 64 \times 64$ | 500 | 300 | 0.833s | 0.785s |
| supernova | $100 \times 100 \times 100$ | 500 | 106 | 0.553s | 0.323s |
| car flow | $368 \times 234 \times 60$ | 500 | 338 | 0.846s | 0.711s |
| crayfish | $322 \times 162 \times 119$ | 800 | 354 | 1.327s | 1.422s |
| solar plume | $126 \times 126 \times 512$ | 600 | 492 | 1.674s | 1.799s |
| computer room | $417 \times 345 \times 60$ | 800 | 227 | 1.28s | 1.076s |

Table 1. The timing results of seven flow data sets for feature and spatially sensitive bag-of-features computation.

### 4.1 Configuration and Timing

We used a hybrid CPU-GPU solution in our computation with the following hardware configuration: Intel Core i7 quad-core CPU running at 3.20GHz, 24GB main memory and an nVidia GeForce GTX 580 graphics card. We implemented streamline tracing and feature calculation in the GPU using CUDA. The input velocity field was loaded into the texture memory on GPU such that velocity derivatives can be computed efficiently. The spatially sensitive bag-of-features for all the streamlines are computed in parallel using OpenMP. As we can see from Table 1, the pre-processing is finished very quickly even for large data sets such as solar plume and computer room.

### 4.2 Streamline Similarity Query

Due to the large number of streamlines that are displayed, visual cluttering and occlusion become a challenge for flow field exploration. In order to overcome this problem, we developed a user interface where users can pick a streamline of interest and request to display streamlines similar to the query streamline. When a query streamline is picked, the similarity between that streamline and each of the remaining streamlines is calculated. After that, users can use a slider to adjust the number of similar streamlines to be displayed. Since the spatially sensitive bag-of-features for each streamline is precomputed, the query can be done in real time. We compare our streamline similarity query results with the ones obtained by using the similarity metric proposed by Lu et al.[12] For the purpose of fair comparison, we implemented their algorithm by considering the features described in Section 3.2 instead of the features (curvature, torsion and curl) used in their original paper. They determined the number of bins in the 1D feature histogram of each streamline segment using Scott's choice.[21] However, we set the number of bins of the feature histogram to be the same as the size of our vocabulary due to the same reason.

Figure 3 and 4 illustrate the comparison between the two methods. In these two figures, the left column shows the query streamlines, the middle column shows our results, and the right column shows the results obtained from the other method. Notice that only the 20% most similar streamlines are displayed. Although most of the similar streamlines found by the two methods are the same, our method has a better overall performance. For the five critical point and solar plume data sets, our method found the streamlines which are more close to the query streamline. The critical point in the middle of the five critical point data set is also revealed by our method. For the tornado data set, the streamlines found by our method have a swirling intensity more similar to that of the query streamline. For the following three data sets: supernova, car flow and crayfish, it is hard to say which method is better because the streamlines found by both methods look similar enough to the query streamline except that they may be at different locations in the flow field. The reason why some straight streamlines show up as similar streamlines in the car flow data set is that there does not exist a sufficient number of streamlines similar to the query streamline given the current threshold on the percentage of similar streamlines to display. For a similar reason, in the crayfish example, swirling streamlines with straight parts are found to be similar to the query streamline (although the query streamline does not contain any straight part). Finally, for the computer room data set, both methods successfully extracted the swirls in the data set, our method then chose some short streamlines as the remaining similar streamlines, whereas the other method chose some longer streamlines.

The reason why our method performs better is that the performance of the other method is actually affected by its segmentation step. As explained in Section 2.2, the termination of the segmentation process is determined by two parameters: the number of points in a segment and the difference between the feature distributions of two possible segments. Based on our experiment, it is hard to find reasonable values for these two parameters that work for different data sets. Even though we tried to manually adjust these two parameters for each data set, the same values did not work very well for all the streamlines in a data set at the same time. In one extreme, a streamline could be a segment by itself. If this happens, their similarity metric will reduce to the approximate version of EMD distance. This is the reason, for example, why some streamlines with swirly tails are considered to be similar to the query streamline in the five critical point data set.

### 4.3 Streamline Clustering

We further tested the effectiveness of our similarity metric by considering streamline clustering. The clustering algorithm we used is affinity propagation,[22] which can automatically determine the best number of clusters by

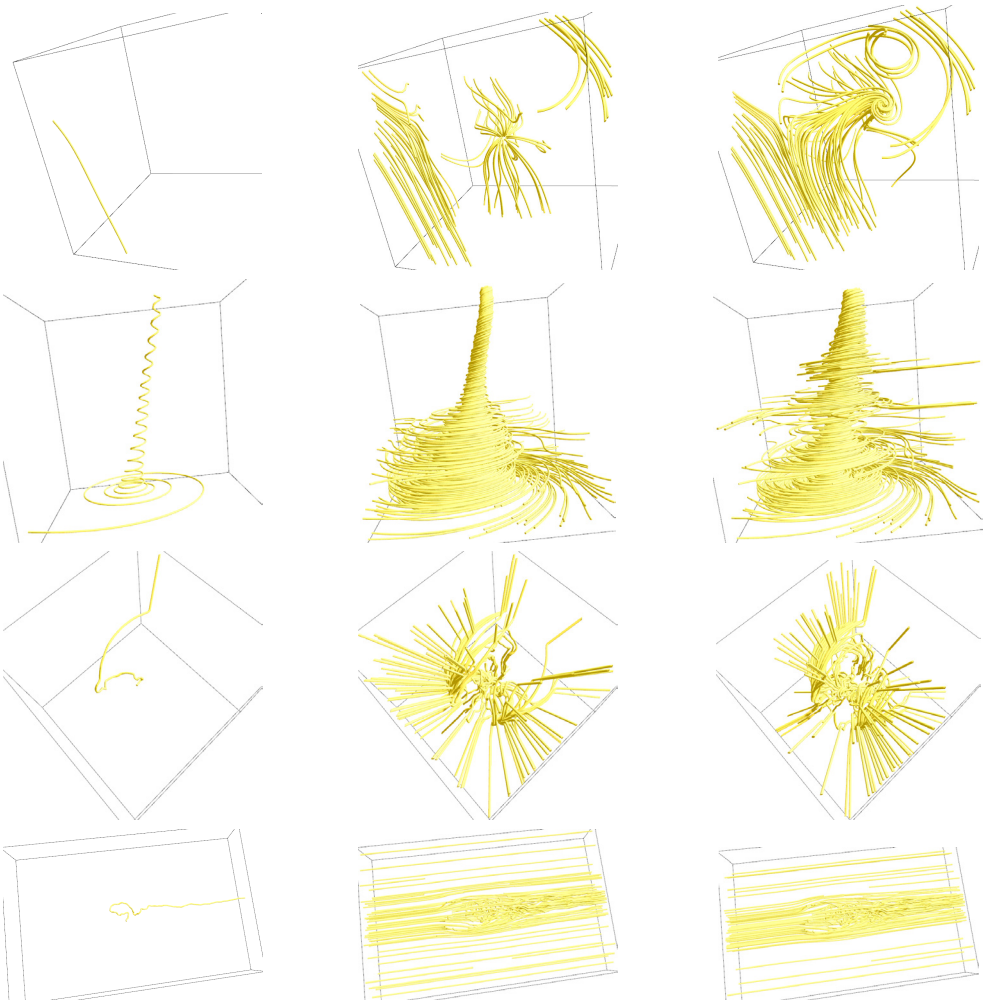



Figure 3. Streamline similarity query results: five critical points (first row), tornado (second row), supernova (third row) and car flow (fourth row). From left to right, query streamline (left column), top 20% similar streamlines found by our method (middle column), and top 20% similar streamlines found using the method by Lu et al.[12]

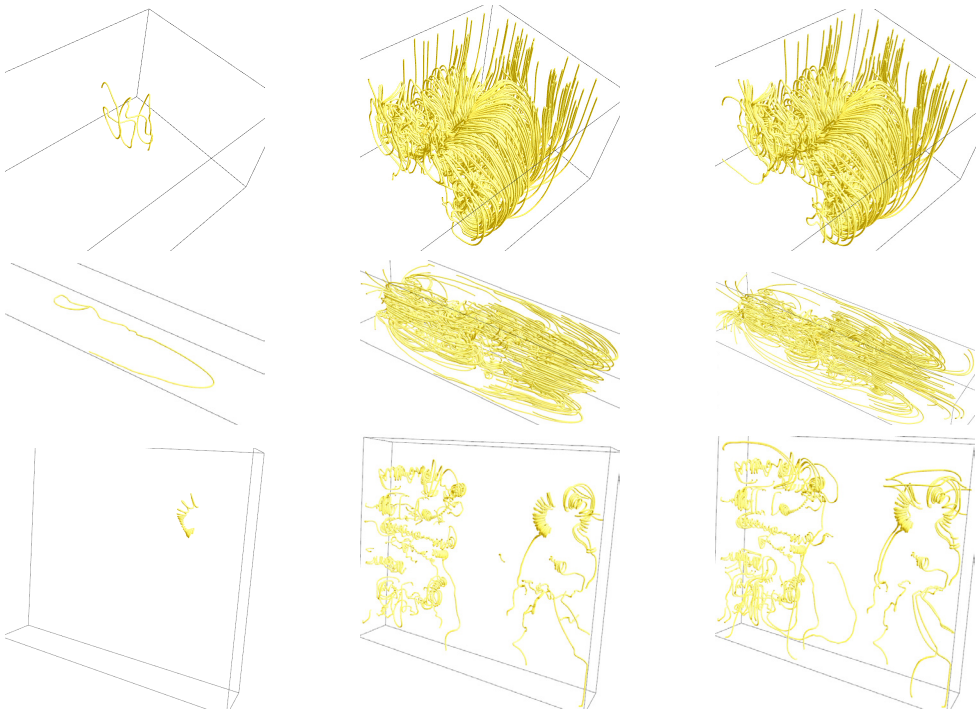



Figure 4. Streamline similarity query results: crayfish (first row), solar plume (second row) and computer room (third row). From left to right, query streamline (left column), top 20% similar streamlines found by our method (middle column), and top 20% similar streamlines found using the method by Lu et al.[12]

message passing. The number of clusters is affected by the value of *preference*. Low preferences leads to small numbers of clusters, whereas high preferences lead to large numbers of clusters. The input to affinity propagation algorithm is a usually a symmetric similarity matrix where each matrix element at $(i, j)$ is the similarity between data points $i$ and $j$. In our experiment, the similarity matrix is computed using our similarity metric and that by Lu et al.[12] respectively for the purpose of comparison.

We carried out the comparison on solar plume and tornado data sets. The results are illustrated in Figure 5 and 6. In these two figures, the clustering results generated by our method are shown on the top, whereas the ones generated by the other method are shown at the bottom. Since affinity propagation does not allow users to specify the number of clusters, we tried to keep the number of clusters as close as possible in our comparison. As we can see, our similarity metric leads to better clustering results in that streamlines with similar shapes are grouped more tightly together. Notice that in the first cluster (purple) of the tornado data set, the long spiral does not look like the remaining streamlines in the same cluster, neither does it look similar to the streamlines in the other clusters. The clustering algorithm failed to make a separate cluster for the spiral because its spatially sensitive bag-of-features is closer to that of the other streamlines in the same cluster. This example revealed a limitation of weighted Manhattan distance: it is not necessarily a natural reflection of how we human beings perceive streamline similarity. We will discuss how we plan to address this problem in Section 5.

Finally, we want to mention that hierarchical clustering results could be easily obtained by starting with a low preference value and then successively increasing the value according to affinity propagation FAQ.[23] Although the resulting partitions will not form a tree, this is often a desirable property of hierarchical clustering, which standard methods such as agglomerative clustering do not have.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an novel method of computing a streamline's feature vector for similarity comparison. This feature vector, also known as spatially sensitive bag-of-features, encodes both the statistical

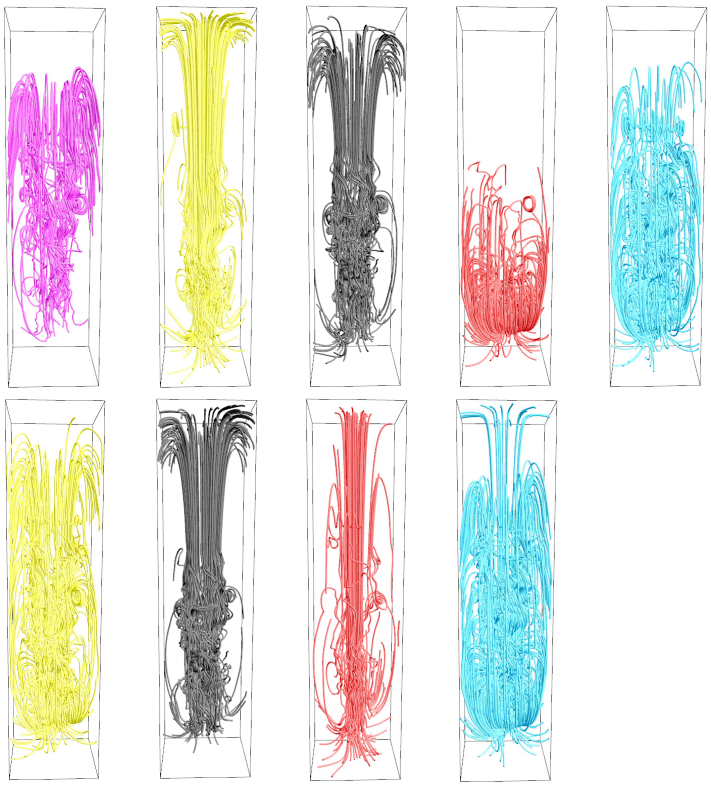


Figure 5. Clustering results for solar plume data set using our method (top) and Lu et al.[12](bottom).

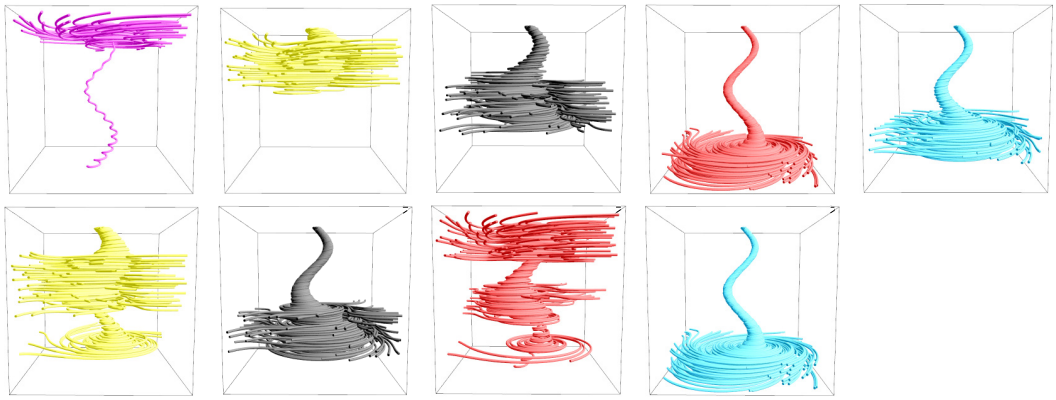


Figure 6. Clustering results for tornado data set using our method (top) and Lu et al.[12](bottom).

distribution of combined features and their spatial relationship. This allows us to evaluate the similarity between two streamlines efficiently by using weighted Manhattan distance. Currently, we use four types of features to derive a feature vector: curvature, torsion, tortuosity and velocity direction entropy. However, our approach allows other types of features to be incorporated into the computation easily. Compared with other distribution-based similarity metrics, our approach is simpler to implement and yields better results.

In the future, we would like to improve the way of how similarity is calculated. Currently, we use weighted Manhattan distance as the similarity measure. However, the weighted Manhattan distance may not be the most natural way to evaluate the similarity if we need to take into account human perception. In other words, it is possible that two streamlines are considered to be similar by a human being, whereas the weighted Manhattan distance between their feature vectors is large (or vice versa). In order to solve this problem, we would like to use machine learning techniques such as support vector machine[24] (SVM) to train the computer to compare two streamlines. More specifically, we ask users to rate the similarity (e.g., in a scale from one to ten) between pairs of streamlines. After that, SVM regression can be used to evaluate the difference between two feature vectors based on the training data. The training process could be made incremental/decremental by leveraging some online SVM regression algorithm.[25]

## REFERENCES


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

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

[3] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* **60**(2), 91–110 (2004).

[4] Sivic, J. and Zisserman, A., "Video google: A text retrieval approach to object matching in videos," in [*Ninth IEEE International Conference on Computer Vision*], 1470–1477 (2003).

[5] Chen, Y., Cohen, J. D., and Krolik, J. H., "Similarity-guided streamline placement with error evaluation," *IEEE Transactions on Visualization and Computer Graphics* **13**(6), 1448–1455 (2007).

[6] Zhang, S., Correia, S., and Laidlaw, D. H., "Identifying white-matter fiber bundles in DTI data using an automated proximity-based fiber-clustering method," *IEEE Transactions on Visualization and Computer Graphics* **14**(5), 1044–1053 (2008).

[7] Corouge, I., Gouttard, S., and Gerig, G., "Towards a shape model of white matter fiber bundles using diffusion tensor MRI," in [*Proceedings of International Symposium on Biomedical Imaging*], 344–347 (2004).

[8] Chen, W., Zhang, S., Correia, S., and Ebert, D. S., "Abstractive representation and exploration of hierarchically clustered diffusion tensor fiber tracts," *Computer Graphics Forum* **27**(3), 1071–1078 (2008).

[9] Jianu, R., Ç. Demiralp, and Laidlaw, D. H., "Exploring 3D DTI fiber tracts with linked 2D representations," *IEEE Transactions on Visualization and Computer Graphics* **15**(6), 1449–1456 (2009).

[10] Moberts, B., Vilanova, A., and van Wijk, J. J., "Evaluation of fiber clustering methods for diffusion tensor imaging," in [*Proceedings of IEEE Visualization Conference*], 65–72 (2005).

[11] McLoughlin, T., Jones, M. W., Laramee, R. S., Malki, R., Masters, I., and Hansen, C. D., "Similarity measures for enhancing interactive streamline seeding," *IEEE Transactions on Visualization and Computer Graphics* (2012).

[12] Lu, K., Chaudhuri, A., Lee, T.-Y., Shen, H., and Wong, P. C., "Exploring vector fields with distribution-based streamlilne analysis," in [*IEEE Pacific Visualization 2013*], (2013).

[13] Berndt, D. J. and Clifford, J., "Using dynamic time warping to find patterns in time series.," in [*KDD workshop*], **10**(16), 359–370 (1994).

[14] Rubner, Y., Tomasi, C., and Guibas, L. J., "A metric for distributions with applications to image databases," in [*Sixth International Conference on Computer Vision*], 59–66 (1998).

[15] Wei, J., Wang, C., Yu, H., and Ma, K.-L., "A sketch-based interface for classifying and visualizing vector fields," in [*PacificVis'10*], 129–136 (2010).

[16] Bronstein, A. M., Bronstein, M. M., Guibas, L. J., and Ovsjanikov, M., "Shape google: Geometric words and expressions for invariant shape retrieval," *ACM Transactions on Graphics* **30**(1), 1 (2011).

[17] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y., "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(7), 881–892 (2002).

[18] Do Carmo, M. P. and Do Carmo, M. P., [*Differential geometry of curves and surfaces*], vol. 2, Prentice-Hall Englewood Cliffs (1976).

[19] Farin, G., [*Curves and surfaces for CAGD: a practical guide*], Morgan Kaufmann Publishers Inc., 5th ed. (2002).

[20] Leopardi, P., "A partition of the unit sphere into regions of equal area and small diameter," *Electronic Transactions on Numerical Analysis* **25**(12), 309–327 (2006).

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

[22] Frey, B. J. and Dueck, D., "Clustering by passing messages between data points," *science* **315**(5814), 972–976 (2007).

[23] Frey, B. J., "Affinity propagation faq." `http://www.psi.toronto.edu/affinitypropagation/faq.html`. [Online; accessed 19-July-2013].

[24] Schölkopf, B. and Smola, A. J., [*Learning with kernels*], The MIT Press (2002).

[25] Cauwenberghs, G. and Poggio, T., "Incremental and decremental support vector machine learning," *Advances in neural information processing systems* , 409–415 (2001).