FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces

Jun Han, Jun Tao, *Member, IEEE*, and Chaoli Wang, *Senior Member, IEEE*

Abstract—For effective flow visualization, identifying representative flow lines or surfaces is an important problem which has been studied. However, no work can solve the problem for both lines and surfaces. In this paper, we present FlowNet, a single deep learning framework for clustering and selection of streamlines and stream surfaces. Given a collection of streamlines or stream surfaces generated from a flow field data set, our approach converts them into binary volumes and then employs an autoencoder to learn their respective latent feature descriptors. These descriptors are used to reconstruct binary volumes for error estimation and network training. Once converged, the feature descriptors can well represent flow lines or surfaces in the latent space. We perform dimensionality reduction of these feature descriptors and cluster the projection results accordingly. This leads to a visual interface for exploring the collection of flow lines or surfaces via clustering, filtering, and selection of representatives. Intuitive user interactions are provided for visual reasoning of the collection with ease. We validate and explain our deep learning framework from multiple perspectives, demonstrate the effectiveness of FlowNet using several flow field data sets of different characteristics, and compare our approach against state-of-the-art streamline and stream surface selection algorithms.

Index Terms—Flow visualization, streamlines, stream surfaces, deep learning, autoencoder, feature descriptor, clustering, selection.

1 Introduction

Understanding large and complex 3D flow fields is critically important in many aero- and hydro-dynamical systems that dominate various physical and natural phenomena in the world. Applications that study these dynamic systems, such as computational fluid dynamics, automotive and aircraft design, weather forecast and climate modeling, and simulation of natural disasters (e.g., earthquakes, hurricanes, tornados), generate large amounts of vector field data that need to be analyzed and visualized. Most fluids (air, water, etc.) are transparent, and thus their flow patterns are invisible to us. Flow visualization is used to make the flow patterns visible so that we can visually acquire *qualitative* and *quantitative* flow information. In this paper, we place our focus on *integration-based flow visualization* as it is most widely used in practice. Specifically, we study integral flow lines (streamlines, pathlines) and flow surfaces (stream surfaces, path surfaces).

Many challenges exist when it comes to generating representative flow lines or surfaces as well as visually exploring a large collection of flow lines or surfaces. Although seeding and selection of streamlines have been well studied, the same problem for stream surfaces is clearly underexplored. All existing approaches for effective line and surface seeding and selection *explicitly* make use of handcrafted features (e.g., entropy, curvature, torsion, saliency, critical points, separation lines, vortex cores) in their solutions. We instead, take a drastically different approach that automatically learns features from the input lines or surfaces and encodes them *implicitly* in a latent space. This is achieved by borrowing techniques from deep learning that has made a significant impact on many fields including those that are closely related to scientific visualization, such as computer vision and computer graphics.

We aim to automatically extract latent features from raw streamline or stream surface data, which can be achieved using an autoencoder. Note that generative adversarial network is capable of generating novel data from a given data set that look at least superficially authentic to human observers, which does not directly match our purpose. We choose the sparse and stacked version of the autoencoder framework.

We introduce FlowNet, a single deep learning approach for streamline and stream surface clustering, filtering, and selection. The key lies in the design of an autoencoder that automatically learns line or surface feature descriptors. We show that by carefully designing the network architecture and loss function, the features learned can be used to well reconstruct the lines or surfaces with minimum errors. To visually explore the features, we perform dimensionality reduction and apply different clustering algorithms. We further develop a visual interface along with intuitive and convenient interactions to enable users to effectively explore the underlying set of streamlines and stream surfaces.

The contributions of this paper are as follows. Our work is the first one that applies deep learning techniques for feature learning of streamlines and stream surfaces. Unlike previous works, which need separated solutions for handling streamlines and stream surfaces, our method can take either streamlines or surfaces for feature learning using a single framework. For either lines or surfaces, there is no need to change the network architecture (as the network input remains in the same form) or feature definition (as

• J. Han, J. Tao, and C. Wang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {jhan5, jtao1, chaoli.wang}@nd.edu.

the feature descriptors are learned implicitly). We integrate feature learning, projection, and exploration into a single framework for visual analysis. Unlike previous methods which are almost exclusively fully-automatic, the visual interface enables users to customize their representative flow line or surface selection results. We validate FlowNet from different perspectives, demonstrate its effectiveness using multiple data sets, and compare our deep learning approach against existing streamline and stream surface selection algorithms.

2 Related Work

Flow line and surface selection. In flow visualization, selecting representative flow lines has become a useful alternative of seed placement. For *view-dependent* streamline selection and filtering, Marchesin et al. [21] measured the contribution of each streamline to the understanding of the vector field and selected those streamlines that have a higher contribution and lower probability leading to visual clutter. Ma et al. [20] presented an importance-driven approach that ensures coherent streamline update when the view changes gradually. For *view-independent* streamline clustering and selection, Yu et al. [43] clustered streamlines hierarchically and formed streamline bundles as representatives that succinctly capture flow features and patterns at varying levels of detail. Tao et al. [31] selected streamlines by considering their contributions to all sample viewpoints. Both streamline selection and viewpoint selection can be achieved using a unified information-theoretic framework which builds two interrelated information channels between a pool of streamlines and a set of sample viewpoints.

Lu et al. [19] advocated a distribution-based approach and utilized dynamic time warping to define the similarity between streamlines for clustering and query. Oeltze et al. [24] evaluated three different kinds of clustering techniques (k-means clustering, agglomerative hierarchical clustering, and spectral clustering) in terms of clutter reduction when visualizing streamlines traced from simulated blood flow.

Only a few works address the issue of flow surface selection. Martinez Esturo et al. [10] favored stream surfaces where the flow is aligned with principal curvature directions. Simulated annealing is used to select a globally optimal stream surface based on a set of stream surface quality measures. Schulze et al. [29] extended the above work to select a set of globally optimal stream surfaces in an iterative manner. All selected surfaces are mutually distant to convey different flow features while reducing visual occlusion and clutter.

All these existing methods leverage handcrafted features that explicitly define the feature representation. In this work, we explore a very different way that implicitly encodes features of lines or surfaces in a latent space. This eliminates the need for users to enumerate individual features and provides the opportunity to self-learn the feature representation, although no intuitive explanation of each dimension can be conveyed. In this sense, our work is similar to Hong et al. [15] which extracts pathline features through latent Dirichlet allocation (LDA) and groups pathlines through fuzzy clustering. The difference is that we employ a neural net for flow feature extraction and they utilized LDA for flow topic extraction.

Deep learning+VIS. Deep learning has been successfully applied to many applications such as computer vision, speech recognition, natural language processing, and bioinformatics, achieving results comparable or even superior to human experts [13]. In visualization, there is a burgeoning of works that integrate deep learning with visualization, especially visual analytics. Almost all of these works attempted to “open the black box” of various neural network models by designing effective user interfaces for interactive exploration, visual understanding, and analytical reasoning [14]. However, little work has been done that applies deep learning techniques to solve scientific visualization problems, in our case, flow line and surface clustering and selection. The challenges mainly lie in the missing of a consistent representation for scientific data (e.g., flow surfaces) as input to deep neural models, and the lack of sufficient high-quality labeled data for achieving acceptable learning performance.

3D shape analysis using CNN. Recently, computer vision and computer graphics researchers have investigated the use of *convolutional neural networks* (CNN) for 3D shape analysis [36]. These methods can be classified into *manifold-based* [4], *multiview-based* [1], [25], [30], and *voxel-based* [27], [39] methods. Manifold-based methods perform CNN operations over geometric features defined on a 3D mesh manifold, which is typically a genus zero or higher genus surface. This does not work for flow surfaces (which are non-closed) and flow lines (which can be treated as degenerated cases of flow surfaces where the seeding curve reduces to a seeing point). Multiview-based methods represent a 3D shape with a set of images rendered from different views and take the image stack as the input of a 2D CNN. However, a flow surface could be severely self-occluded, which renders a multiview-based solution impractical. Voxel-based methods model a 3D shape as a function sampled on voxels and define a 3D CNN over voxels for shape analysis. We choose this approach for our FlowNet design. Even though this method is currently limited to a resolution around 64^3 due to the high memory and computational costs [27], it still works in our application as no precise line or surface is required for computing the loss function and evaluating the reconstruction quality. Therefore, we can downsample the flow lines or surfaces to a resolution that is amenable to GPU computation.

3 FlowNet

Given a large set of streamlines or stream surfaces generated from a flow data set, we aim to identify a subset that best captures the underlying flow features and patterns. Instead of identifying the representatives directly, we opt to partition the input set into clusters and then select one from each cluster to form the representatives. A key question is how to learn the *feature descriptor* for a line or surface. We propose FlowNet design based on an *autoencoder* [2] that learns the feature representation using a deep neural net. We first voxelize and downsample each *object* (line or surface) into a 3D binary volume of an appropriate resolution, which will be the input to the autoencoder. The autoencoder trains the neural net and learns feature descriptors automatically. Instead of relying on labeled data for *supervised learning*, FlowNet applies the autoencoder for *unsupervised learning*,

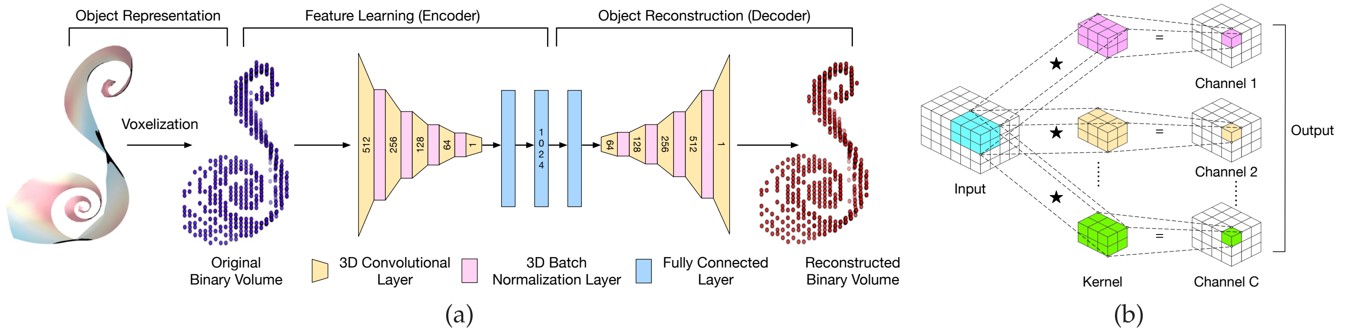


Fig. 1: (a) FlowNet for object feature learning. The input to FlowNet is the voxelized object representation. The network includes convolutional (CONV), batch normalization (BN), and fully-connected (FC) layers. (b) An example of a 3D CONV operation.

or more precisely, *self-supervised learning*. This eliminates the need to produce labeled data for training. Once the network converges, we apply t-SNE [34] to the feature descriptors for dimensionality reduction. We perform interactive clustering using DBSCAN [9] to identify the representatives. Finally, we design a visual interface for users to intuitively explore the line or surface collection and perform visual analysis and analytical reasoning.

3.1 Feature Descriptor Learning

Object voxelization. We define two representations of an object: *sequence* and *voxel* representations. The sequence representation of an object is a 1D vector, $\mathbf{s} = \{x_1, y_1, z_1, \cdots, x_n, y_n, z_n\}$, where (x_i, y_i, z_i) is a point on the object and n is the number of points. The voxel representation of an object is a volume $\mathbf{V}$ with size $L \times W \times H$. For streamlines, each object is a streamline represented by a sequence of points. For stream surfaces, each object is a stream surface where the sequence representation stores, line by line, the corresponding points following the streamline or timeline direction. We apply the rounding strategy so that each point on an object is mapped to its nearest voxel. If the voxel $\mathbf{V}[l_i, w_j, h_k]$ is occupied by the object, then the value of this voxel is 1 otherwise the value is 0.

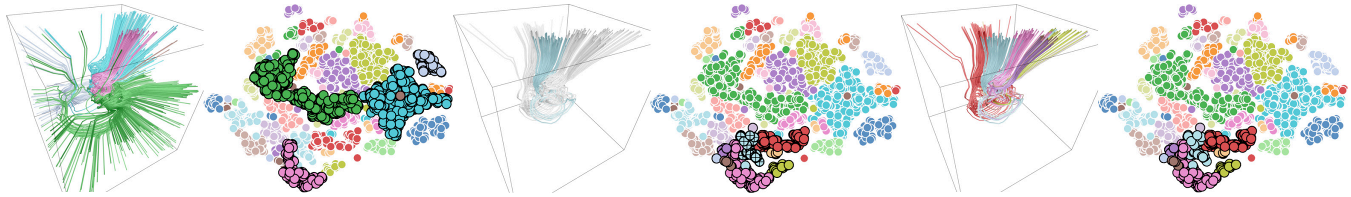
Object voxelization transfers the sequence representation of an object into its voxel representation. The rule is that $\mathbf{V}[x_i, y_i, z_i] = 1$ for $i = 1$ to n and the remaining voxels are filled with 0. Due to the GPU memory constraint, given the original volume $\mathbf{V}$ of size $L \times W \times H$, we downsample it into a volume $\mathbf{V}'$ of size $L' \times W' \times H'$. We first calculate the downsamping ratio of each dimension: $x_r = L/L'$, $y_r = W/W'$, $z_r = H/H'$. Then we set $\mathbf{V}'[x'_i, y'_i, z'_i] = 1$ for $i = 1$ to n, where $x'_i = x_i/x_r$, $y'_i = y_i/y_r$, $z'_i = z_i/z_r$. The remaining voxels are set to 0.

CNN and autoencoder. As a class of deep, feed-forward artificial neural networks, a CNN performs the computation by *neurons*, which are organized into *layers* of different types: *convolutional* (CONV) and *fully-connected* (FC). The CONV layer detects local and non-linear combinations of features from the previous layer to capture important information from the raw data. The FC layer serves as further learning (from general to specific) of the input observation, combining local features into global features.

An autoencoder consists of two parts: *encoder* and *decoder*. The encoder takes an object as input and maps it to a feature descriptor. The decoder takes the feature descriptor as input and reconstructs the object. The basic autoencoder only consists of FC layers for unsupervised learning, which cannot ensure that the network learns a concise data representation and could impact its performance in reconstructing complex data such as 3D models. We can improve the reconstruction results by introducing CONV into the autoencoder. This is because there are always linear combinations of neurons in the FC layers while CONV layers allow local and non-linear combinations of neurons, which enables the network to learn a complex data representation. Another advantage of using CONV layers is that it reduces the parameters to be learned through parameter sharing.

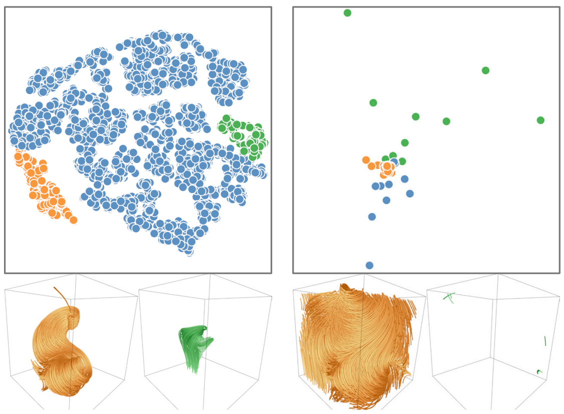
FlowNet architecture. As sketched in Figure 1 (a), our FlowNet design includes two stages: *feature learning* (encoder) and *object reconstruction* (decoder). The first stage learns object features by non-linearly mapping each object representation to a feature descriptor (we use 1024 dimensions). Inspired by the work of Girdhar et al. [11], we design a CNN for feature learning and object reconstruction. Since there is no padding, we set the stride to 1 in all CONV layers. Moreover, we apply the *batch normalization* (BN) layer [16] to prevent the network from overfitting and gradient vanishing, while speeding up network training. The FC layers at the end enable the network to learn global features from local ones. The second stage is the inverse of the first. It reconstructs the object based on the feature descriptor learned. A loss function is used to indicate the error between the reconstructed and original binary volume representations. FlowNet will adjust the parameters iteratively so that a more accurate feature representation can be learned.

Specifically, FlowNet takes a $L \times W \times H$ voxel representation of an object as input. The encoder consists of four CONV layers with BN added in between, one CONV layer without BN, followed by two FC layers. With five CONV layers and four BN layers, the decoder takes this embedded feature and maps it to a $L \times W \times H$ voxel grid. We apply the rectified linear unit (ReLU) [23] at the hidden layers and the sigmoid function at the output layer. Compared to other activation functions (e.g., tanh and sigmoid), ReLU can effectively avoid two major challenges in network training: gradient vanishing and explosion. Gradient vanishing happens when the gradient is close to zero in some hidden layers which prevents updating the parameters to their pre-



(a) (b) (c)

Fig. 2: A sequence of cluster interactions in the t-SNE view and the linked volume view using the supernova data set: (a) choosing multiple clusters simultaneously, (b) expanding from one selected cluster highlighted at the bottom of the t-SNE view in (a) to its neighboring clusters, and (c) looping through each of these clusters (the focal cluster is highlighted with additional + signs). In (c), the remaining streamlines in the neighborhood are drawn in gray as the context.



(a) (b)

Fig. 3: Comparison of t-SNE projections of (a) feature descriptors and (b) binary volumes using the five critical points data set. Two point groups (orange and green) are selected via brushing and linking and their corresponding streamlines are shown. The unselected points are in blue.

vious layers. Gradient explosion happens when the gradient is close to infinity which keeps the network from learning the structure of data. We train FlowNet with a binary cross-entropy loss on the final voxel output against the original voxel input. This loss qualifies the difference between the probability distributions of the true and predicted data. Other loss functions (e.g., mean squared error) will lead to a slower convergence of the network because they are prone to gradient vanishing [12]. The loss function of one training sample is defined as

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[p_n \log \hat{p_n} + (1-p_n)\log(1-\hat{p_n})\right], \quad (1)$$

where p_n is the target probability (1 or 0) of a voxel being filled, $\hat{p_n}$ is the predicted probability obtained through FlowNet, and $N = L \times W \times H$. The total loss is the sum of losses for all training samples.

3.2 Dimensionality Reduction and Object Clustering

Exploring the feature descriptors generated from FlowNet for clustering and selection requires mapping these feature descriptors to a low-dimensional (e.g., 2D) space and then grouping similar objects via clustering. The input to dimensionality reduction is the distance matrix recording the Euclidean distances between feature descriptors where each feature vector has been individually normalized [16] using L1-norm. After experimenting with three popular dimensionality reduction methods: t-SNE [34], MDS [18], and Isomap [33], and three widely used clustering algorithms: DBSCAN [9], k-means, and agglomerative clustering, we choose the combination of t-SNE and DBSCAN. Readers are referred to Section 1 in the Appendix for the details. All results presented for the rest of the paper use this combination.

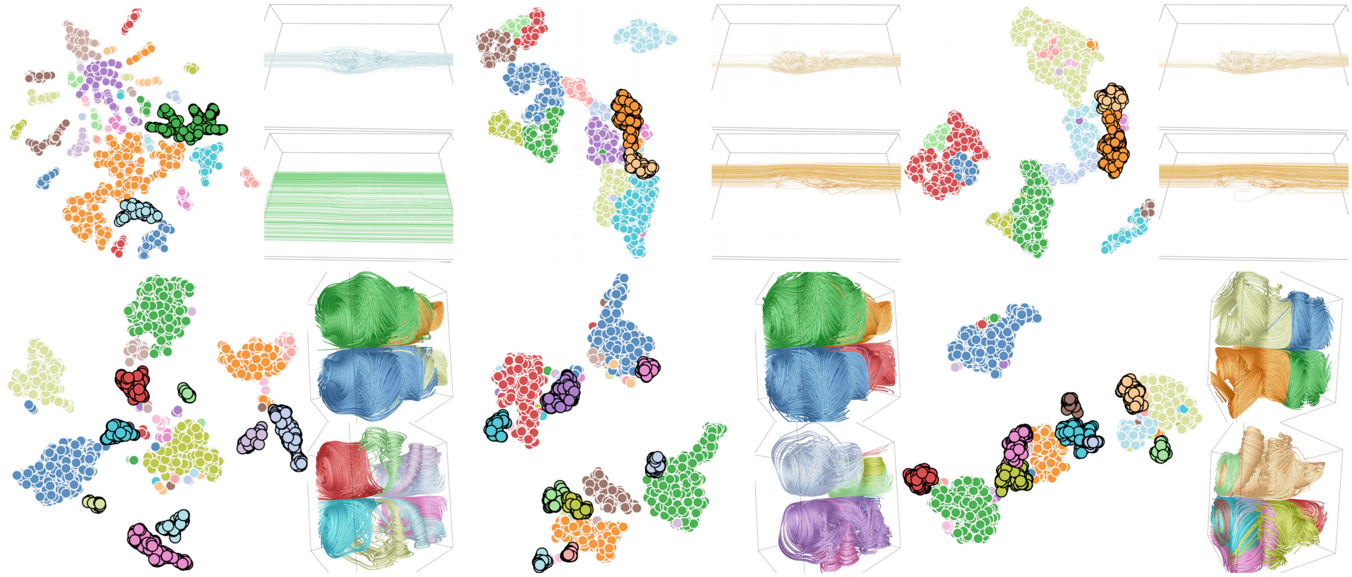
3.3 Interface and Interaction

Our FlowNet interface consists of two views: the *volume view* and *projection view*. Both views are connected via brushing and linking: when users interact with one view, the other view will be automatically updated. The volume view displays the line or surface objects in the original 3D spatial domain and the projection view displays the objects as points in the abstract 2D space. We provide the following functions to explore these objects and their features descriptors:

Clustering. Our interface allows users to interactively tune the parameters of DBSCAN (e.g., the maximum distance between two feature descriptors and the minimum number of samples in one cluster) to generate the desired clustering results. To distinguish each cluster, we draw neighboring clusters using different colors. The selected cluster is highlighted with a black boundary and the volume view displays the corresponding objects. Users can also select multiple clusters simultaneously in the projection view and examine the relationships among them in the volume view. An example is shown in Figure 2 (a).

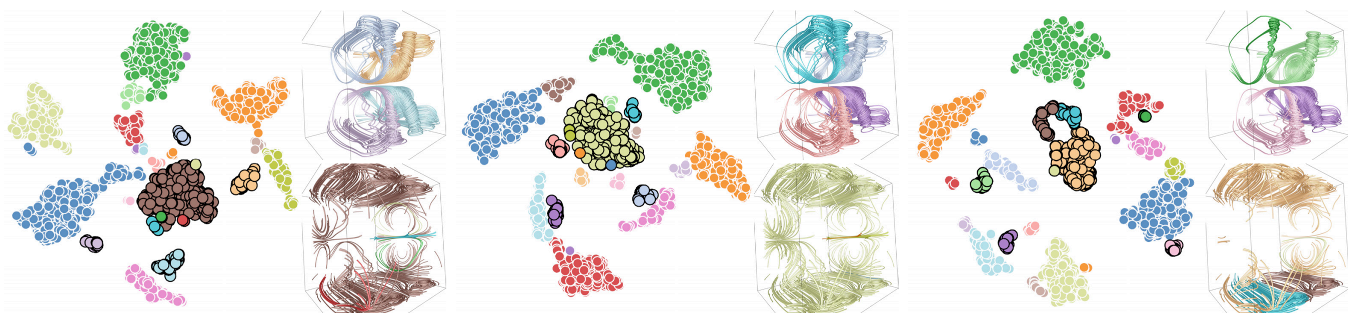
Representatives. To identify one representative from each cluster, we calculate the cluster's center as the data point where the sum of the Euclidean distances from this point to all the other points in the same cluster is the minimum. Users can interactively set the number of representatives. The volume view displays these representative objects, and the projection view shows the clustering result with the centers highlighted. An example is shown in Figure 9.

Neighborhood. By computing the distance between the centers of two clusters as their inter-cluster distance, we allow users to "expand" one selected cluster to its neighborhood and conveniently explore the neighboring clusters. An example is shown in Figure 2 (b). To verify the similarities and differences among these neighboring clusters, users can examine these clusters one by one ordered by their distances to the selected cluster. Such an example is shown in Figure 2 (c).



(a) FlowNet feature Euclidean distance (b) streamline MCP distance (c) streamline Hausdorff distance

Fig. 4: Comparison of different distance measures. Top row: the car flow data set showing 35 clusters. Two streamline clusters are shown. Bottom row: the two swirls data set showing 36 clusters. Four largest streamline clusters are shown at the top-right. Eight selected neighboring streamline clusters are highlighted in the t-SNE view and shown at the bottom-right.



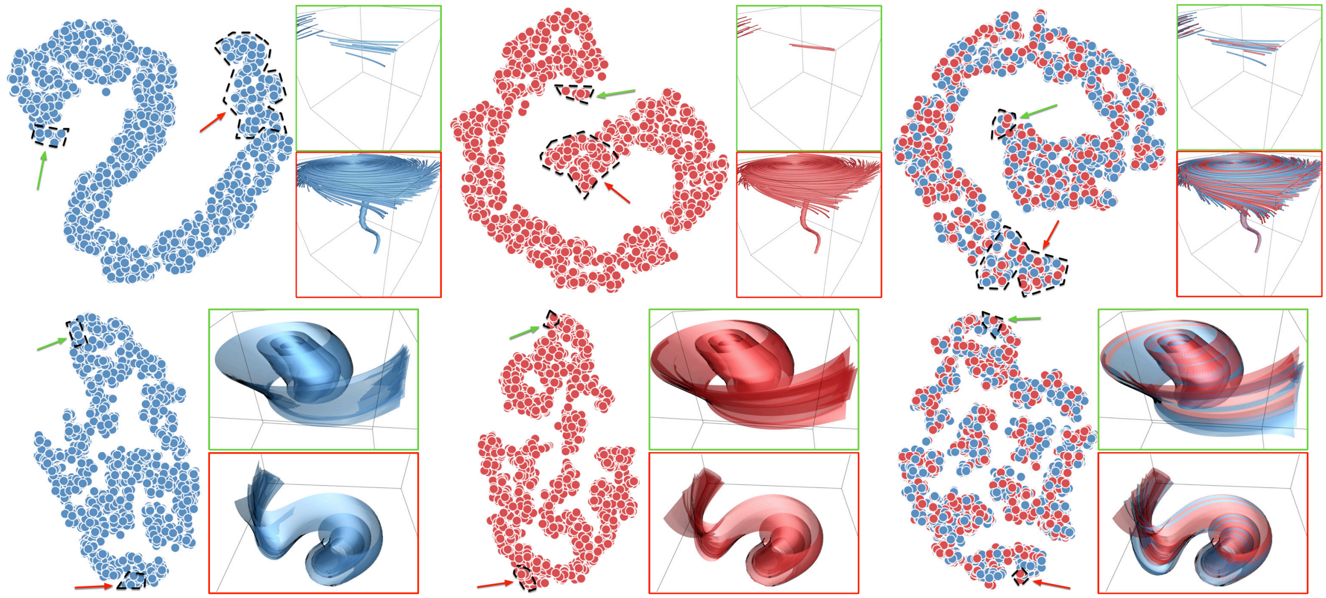
(a) binary cross-entropy, F_1 score = 0.88 (b) MSE, F_1 score = 0.75 (c) Dice loss [22], F_1 score = 0.84

Fig. 5: Comparison of feature descriptors under different loss functions using the two swirls data set. All generate 25 clusters through DBSCAN. The selected streamline clusters are highlighted and shown in the t-SNE view.

3.4 Validation

Feature descriptor. To justify the need of deriving feature descriptors from streamlines, we compare the results generated using feature descriptors with FlowNet employed vs. direct use of binary volumes without FlowNet employed. The distance between two binary volumes is their summed, voxel-wise Euclidean distance. We project the streamline binary volumes with t-SNE for the five critical points data set, as shown in Figure 3 (b). The brushing and linking result shows that projecting binary volumes directly does not help to reveal useful potential structures, for example, the streamlines around the critical regions that capture the main flow features, while projecting feature descriptors with t-SNE reveals these main features, as shown in Figure 3 (a). This comparison indicates that by extracting feature descriptors using the autoencoder, FlowNet can preserve structure exhibited by the streamline set rather than manufacturing structure by coincidence.

Distance measure. To verify the effectiveness of using the Euclidean distance between the corresponding feature descriptors for dimensionality reduction, we compare our distance measure against the mean of the closest point (MCP) distances between streamlines [43] and Hausdorff distance between streamlines [28] previously used to measure streamline similarity. We use t-SNE to project the data points and DBSCAN to group these points using the car flow and two swirls data sets, as shown in Figure 4. The result of the car flow data set shows that our distance measure can well separate the streamlines passing *through* the car (refer to the cyan cluster) from those passing *by* the car (refer to the green cluster), as shown in (a). Further brushing and linking shows that the surrounding small clusters correspond to streamlines located at the volume boundary. The other two distance measures, however, separate the streamlines passing through the car into different clusters, as shown in (b) and (c), which is not desirable. The result of the two swirls data set shows that all these three distance measures yield the four biggest streamline clusters which reveal the major structure of the two swirling patterns. However, our distance measure can better separate

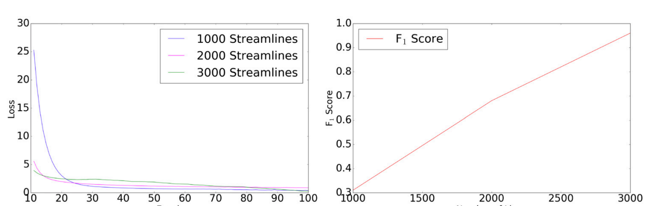


(a) training set only (b) test set only (c) training set + test set

Fig. 6: Evaluation of FlowNet using the tornado streamline and stream surface data via brushing and linking. Two selected point groups and their corresponding streamlines or surfaces are shown. The training and test sets are in blue and red, respectively.

contextual streamlines from those streamlines in the biggest clusters. In addition, these contextual streamline clusters in (a) exhibit a better symmetry compared with those in (b) or (c).

Loss function measure. To verify the effectiveness of using binary cross-entropy for FlowNet training, we compare feature descriptors under three different loss functions: binary cross-entropy, mean squared error (MSE), and Dice loss [22]. We apply t-SNE projection and DBSCAN clustering for the two swirls data set, as shown in Figure 5. The clustering result shows that all these three loss functions can separate out the four biggest clusters. However, compared to Dice loss, binary cross-entropy and MSE can discover the streamlines located at the volume boundary and therefore, better separate contextual streamlines. Moreover, using binary cross-entropy yields the highest F_1 score. Therefore, we choose binary cross-entropy as the loss function.



(a) binary cross-entropy loss (b) F_1 score

Fig. 7: Performance curves under different numbers of training streamlines for the five critical points data set.

Underfitting and overfitting. Two central challenges in machine learning are *underfitting* and *overfitting* [13]. Underfitting occurs when the model is not able to fit the training set. Overfitting occurs when the model fits the training set perfectly but fails to fit the test set. In Table 1, we report for each data set, the sizes of training and test sets and their corresponding F_1 scores. Note that the test sets are randomly generated, in the same fashion as the training sets (refer to Section 4.1).

F_1 score is defined as

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = \frac{2}{\frac{\text{FNs}+\text{TPs}}{\text{TPs}} + \frac{\text{FPs}+\text{TPs}}{\text{TPs}}}, \tag{2}$$

where TPs, FPs, and FNs stand for true positives, false positives, and false negatives, respectively. In our context, given a voxel in the binary volume, it is a true positive/false positive/false negative if the value of ground truth is $1/0/0$ and the possibility predicted by FlowNet is greater/greater/less than 0.5. Ranging between 0 and 1, F_1 score defines the similarity between the original and the predicted objects. If F_1 score is closer to $1/0$, it indicates that the predicted object is more/less similar to the original object.

The F_1 scores reported for training show that FlowNet is not underfitting. To visually demonstrate that FlowNet is not overfitting, we qualitatively compare the t-SNE views of the training set and test set via separate projections, as shown in Figure 6 (a) and (b). Note that the embedding is with respect to the data points, not the underlying space. Although the orientations or spreads are different (which is due to the randomness of the t-SNE algorithm), both views share the similar global structure and local characteristics. Through brushing and linking, we can further verify that the trained model works as expected for the test set. We also experiment with projecting the training and test sets in the same view, as shown in Figure 6 (c). The interspersed points from both sets confirm that the trained model can map new, previously unseen inputs to appropriate feature vectors.

4 Results and Discussion

4.1 Data Sets and Network Training

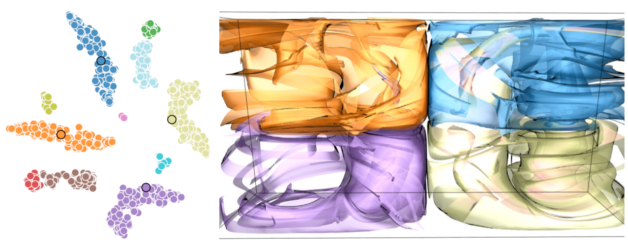
Data sets. We experimented with the list of data sets

TABLE 1: Left: the dimension of each data set and respective kernel size used. Middle: the training and test set sizes and respective F_1 scores for streamlines. Right: the training and test set sizes and respective F_1 scores for stream surfaces.

Data Set	Original Dimension	Downsampled Dimension	Kernel Size	Training # Lines	Training F_1 Score	Testing # Lines	Testing F_1 Score	Training # Surfaces	Training F_1 Score	Testing # Surfaces	Testing F_1 Score
ABC	$51 \times 51 \times 51$	$51 \times 51 \times 51$	$3 \times 3 \times 3$	3,000	0.91	3,000	0.82	2,000	0.84	2,000	0.71
Bénard flow	$128 \times 32 \times 64$	$64 \times 16 \times 32$	$4 \times 1 \times 2$	3,000	0.87	3,000	0.80	2,000	0.84	2,000	0.79
car flow	$368 \times 234 \times 60$	$92 \times 59 \times 15$	$6 \times 4 \times 1$	3,000	0.81	3,000	0.69				
computer room	$417 \times 345 \times 60$	$105 \times 87 \times 15$	$8 \times 6 \times 2$	3,000	0.74	3,000	0.68	2,000	0.83	2,000	0.59
crayfish	$322 \times 162 \times 119$	$81 \times 40 \times 30$	$4 \times 2 \times 2$	3,000	0.86	3,000	0.78				
five critical pts	$51 \times 51 \times 51$	$51 \times 51 \times 51$	$3 \times 3 \times 3$	3,000	0.96	3,000	0.72	2,000	0.72	2,000	0.57
solar plume	$126 \times 126 \times 512$	$32 \times 32 \times 128$	$2 \times 2 \times 8$	4,000	0.83	4,000	0.76	1,000	0.84	1,000	0.57
square cylinder	$192 \times 64 \times 48$	$96 \times 32 \times 24$	$8 \times 3 \times 2$	3,000	0.84	3,000	0.72	2,000	0.91	2,000	0.86
supernova	$100 \times 100 \times 100$	$50 \times 50 \times 50$	$2 \times 2 \times 2$	3,000	0.86	3,000	0.75				
tornado	$64 \times 64 \times 64$	$50 \times 50 \times 50$	$3 \times 3 \times 3$	3,000	0.91	3,000	0.76	2,000	0.88	2,000	0.78
two swirls	$64 \times 64 \times 64$	$32 \times 32 \times 32$	$4 \times 4 \times 4$	3,000	0.88	3,000	0.75	2,000	0.91	2,000	0.81

TABLE 2: F_1 scores under different training samples.

Data Set	computer room	five critical pts	supernova	two swirls
# Lines	1,000	1,000	1,000	1,000
F_1 Score	0.36	0.31	0.38	0.35
# Lines	2,000	2,000	2,000	2,000
F_1 Score	0.62	0.68	0.69	0.67



(a) (b)

Fig. 8: Representative stream surface selection with t-SNE projection and DBSCAN clustering using the Bénard flow data set.

shown in Table 1. From top to bottom, these data sets are: the Arnold-Beltrami-Childress (ABC) incompressible flow which is an exact solution of Euler's equation [7], the liquid flow between two parallel planes [38], the air flow around a car [21], the air flow in a computer room [37], the heat flow around a cooking crayfish [21], a synthesized flow field consisting of five critical points [42], the compressible downflow solar plume [26], the flow around a confined square cylinder [35], the flow of core-collapse supernovae [3], a procedurally generated tornado [5], and swirls resulting from wake vortices [21].

FlowNet training. We implemented FlowNet in PyTorch using an NVIDIA TITAN Xp 1080 GPU for network training. In the training process, we initialized all layers of FlowNet from scratch using $\mathcal{N}(0, 0.01)$ and applied the Adam optimizer [17] with learning rate 10^{-6} to update the parameters. We used the minibatch size of 1 and trained FlowNet with 100 epochs. For training efficiency, we follow a simple scheme to decide the training sample size. For streamlines, we initialize the training sample size with $1,000$ and train FlowNet. Then, we check the F_1 score after training FlowNet with 100 epochs. If the score is acceptable (e.g., larger than 0.7 based on our empirical experience), the training sample size is determined. Otherwise additional $1,000$ streamlines will be added to the training pool to retrain FlowNet. For stream surfaces, we initialize the training sample size with $1,000$, and if the training F_1 score is less than 0.7, additional 500 stream surfaces will be added to the training pool to retrain FlowNet. Based on this scheme, the size of the training set and the corresponding kernel size are listed in Table 1. The testing F_1 scores for the computer room, five critical points, and solar plume surface data are relatively low (less than 0.6), mainly due to the complex flow features exhibited by these data sets.

In Figure 7, we report the binary cross-entropy losses and F_1 scores under different training samples for the five critical points data set. In (a), we can see that the binary cross-entropy loss converges fast as the number of training samples increases. In (b), we can see that the F_1 score significantly improves when using more streamlines in the training process. This indicates that the performance of FlowNet is highly related to the number of training samples used. Adding more streamlines to the training pool, FlowNet converges faster and F_1 score also improves. However, both benefits are at the expense of longer training time. Balancing between performance and training time, we choose an increment of $1,000$ streamlines for the training. In Table 2, we report F_1 scores under different training samples for different data sets. It is clear that using $1,000$ and $2,000$ streamlines cannot achieve a good performance (e.g., F_1 score is larger than 0.7), while the performance is acceptable if $3,000$ samples are used for training, as shown in Table 1. Based on this experiment, we conclude that using $3,000$ streamlines is enough to train FlowNet. For the solar plume data set, we use $4,000$ streamlines for training, leading to the F_1 score of 0.83 ($3,000$ streamlines only give the F_1 score of 0.67).

All streamlines are traced from seeds randomly placed in the domain. All stream surfaces are traced from random seeding curves following the binormal directions. Each seeding curve is generated by tracing in the binormal field with a random starting point and length [32]. For stream surfaces, we do not experiment with the car flow, crayfish, and supernova data sets, since the flow patterns in these data sets are either laminar or too complex to be effectively captured by the randomly-placed surfaces. The training time is mainly determined by the sizes of the kernel and training set. Our experiments show that the Bénard flow stream surface data require the lowest training time (15 minutes per epoch) while the computer room streamline data require the highest training time (90 minutes per epoch). So training 100 epochs would take anywhere from one to seven days.

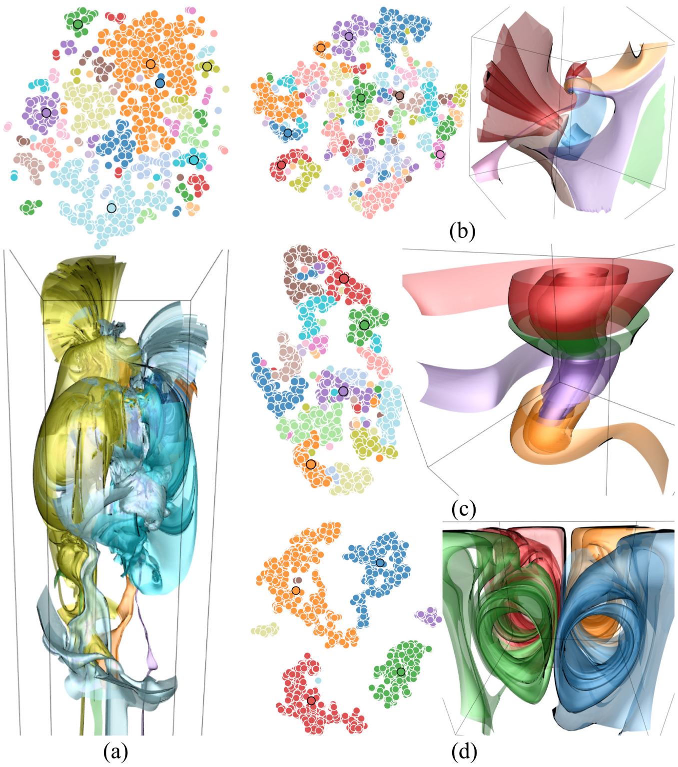


Fig. 9: Representative stream surfaces of the solar plume, five critical points, tornado, and two swirls data sets. (a) to (d) show 7, 7, 4, and 4 representative surfaces, respectively.

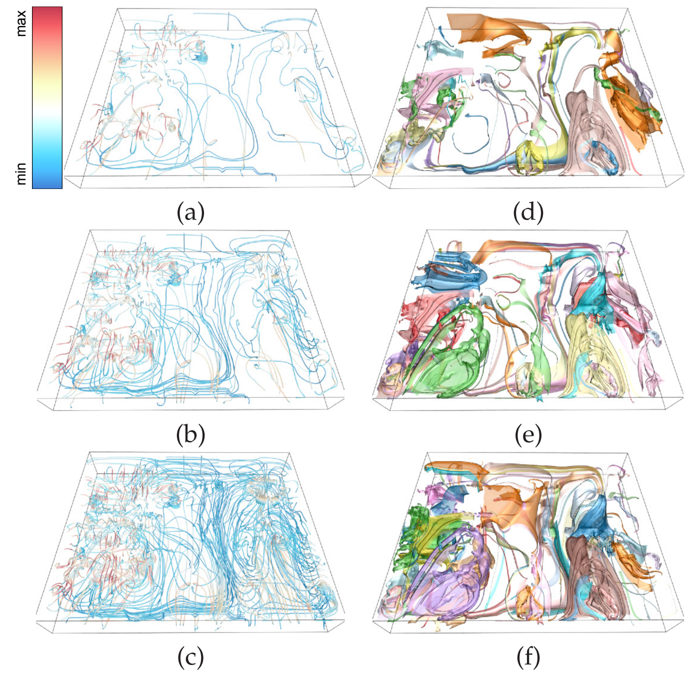
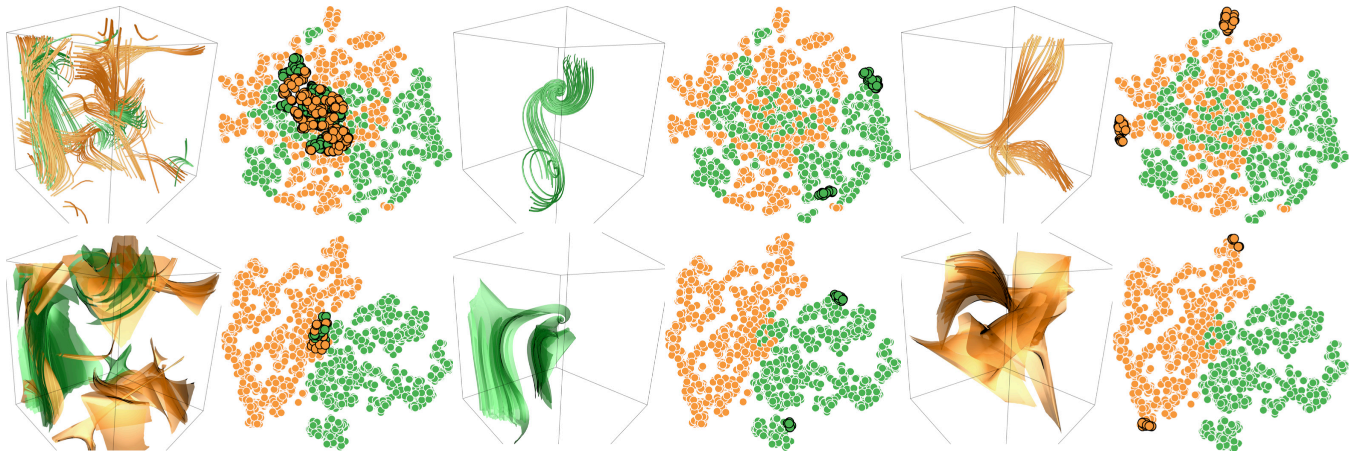


Fig. 10: Representative streamlines and stream surfaces of the computer room data set. (a) to (c) show 70, 150, and 300 streamlines, respectively. (d) to (f) show 40, 60, and 80 stream surfaces, respectively. Velocity magnitudes are mapped to streamline colors.

4.2 Results and Feature Understanding

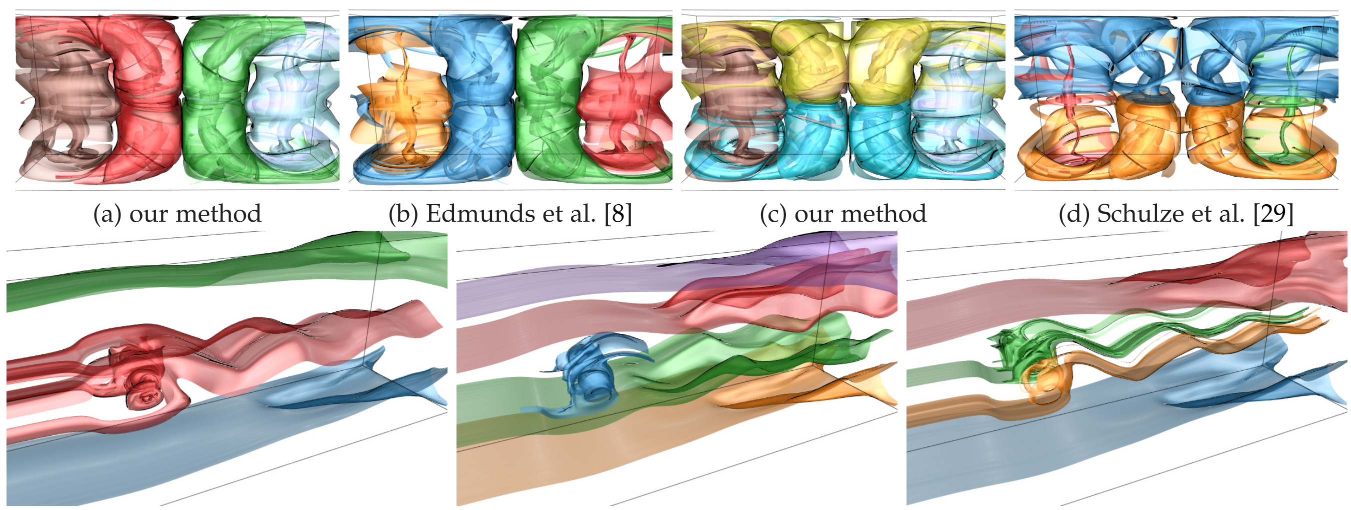
Clustering and selection results. In Figure 2, Figures 4 to 6, Figures 8 to 10, and Figure 2 in the Appendix, we show the results of clustering and selection of streamlines and stream surfaces. The brushing and linking results in Figure 3 (a), Figure 6, and Figure 1 (a) in the Appendix show the good correspondence of neighboring points in the t-SNE view and neighboring lines or surfaces in the spatial view. This indicates that FlowNet feature vectors are a faithful representation of the underlying streamlines and stream surfaces in terms of their shapes and locations. It also shows that the t-SNE projection well preserves neighborhood information. The clustering results in Figures 2, 4, 5, and Figure 2 in the Appendix show that meaningful clustering and flexible exploration can be achieved using our method. In Figures 8 to 10, we show representative streamline and stream surface results. Unlike representative streamlines which are normally in the range of tens to hundreds, representative stream surfaces are typically within ten or up to tens. Therefore, we select representative streamlines automatically, while giving the option of a two-step process for representative stream surface selection. With this option, we first generate a certain number of representative stream surfaces and then let users pick a subset that strikes a balance between surface representativeness and domain coverage. In Figure 8, we show a set of representative surfaces following this two-step process. Four surfaces are selected from the t-SNE view showing 11 clusters. These four surfaces are the centroids of the largest four clusters. Figure 12 (a) and (c) show two other sets of representative surfaces based on the t-SNE view shown in Figure 8 (a). With this process, users are able to generate customized representative surface results. In Figure 10, the representative streamlines and surfaces show that we can generate a good representation of the flow fields with representative streamlines and surfaces at varying levels of detail. The appropriate number of representatives is determined empirically as users can make the adjustment interactively to generate desirable results.

Separation of cross-dataset features. We also experiment with the joint training of streamline and stream surface data drawn from different data sets. We select two data sets and use half of the training samples from each data set for joint training. The results are shown in Figure 11. We can see from the t-SNE view that the two data sets are largely separated in the projection as these two data sets contain very dissimilar flow features and patterns. The first row of Figure 11 shows that the overlapped points in the t-SNE view correspond to similar streamlines around the volume boundary while the separated points correspond to streamlines of distinct spatial locations and flow patterns. The stream surface results in the second row also confirm similar findings, although there is a less number of similar surfaces. This is mainly because stream surfaces are one dimension higher than streamlines. Using random seeding, it is less likely to generate similar stream surfaces from these two data sets. This experiment shows the potential of FlowNet in separating cross-dataset features and the possibility to generalize FlowNet to handle multiple data sets.



(a) (b) (c)

Fig. 11: Joint training of the five critical points (green) and ABC (orange) streamline and stream surface data. (a) highlights where the two data sets overlap in the t-SNE view. (b) and (c) show an example of where the two data sets are separated in the t-SNE view.



(a) our method (b) Edmunds et al. [8] (c) our method (d) Schulze et al. [29]

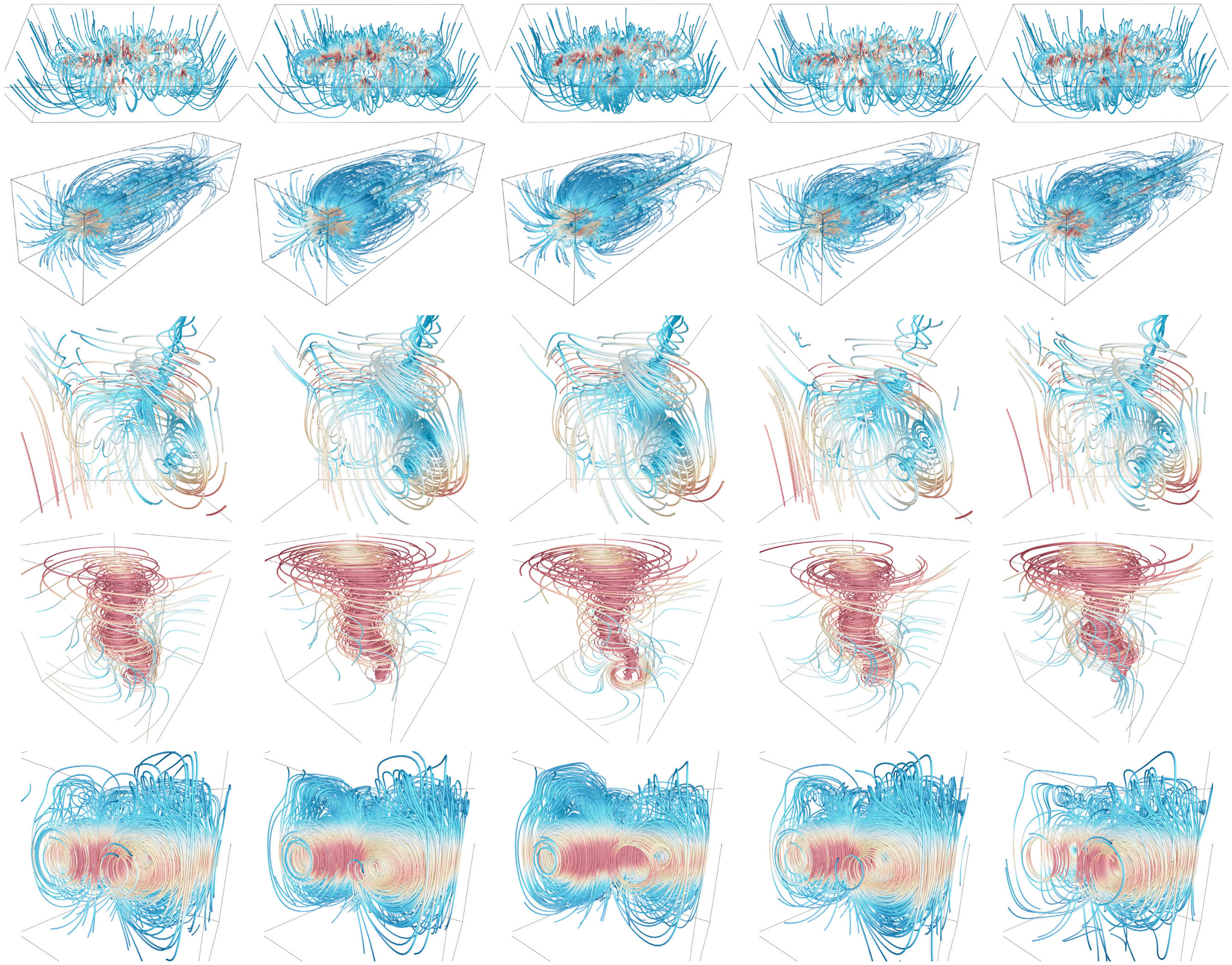
(e) our method (f) Edmunds et al. [8] (g) Schulze et al. [29]

Fig. 12: Top to bottom: comparison of surface selection results of the Bénard flow and square cylinder data sets using different methods. (a) to (d) show four stream surfaces each. (e) to (g) show three, five, and four stream surfaces, respectively.

4.3 Comparison against Existing Methods

Stream surface selection. In Figure 12, we compare our stream surface selection results against those generated from the feature-centered automatic surface seeding by Edmunds et al. [8] and the global selection of stream surfaces by Schulze et al. [29]. Both methods being compared are fully automatic, while our method provides the two-step process to users so that they can handpick representative stream surfaces. With this flexibility, for the Bénard flow data set, we are able to generate representative stream surface results similar to those generated by Edmunds et al. [8] (see (a) and (b)) and Schulze et al. [29] (see (c) and (d)). For the square cylinder data set, although the numbers of representative stream surfaces are not the same, the important surface features on the left are well captured by all three methods.

Streamline selection. In Figure 13, we compare our streamline selection results against those generated from the dual information channel based method of Tao et al. [31] and the entropy-based method of Xu et al. [41]. For fairness, each method produces the same number of streamlines. The information channel is built between the set of streamlines and a set of sample viewpoints. We generate the selection results using three criteria: $p(s)$ (streamline probability), $I(s;V)$ (streamline information), and REP (streamline representativeness). Our method and each of the three criteria of Tao et al. [31] select representatives from the same pool of streamlines. The entropy-based method generates streamlines iteratively guided by the conditional entropy between the original vector field and the field reconstructed from selected streamlines. By comparing the results side by side, we can observe that our method strikes a good balance between streamline informativeness and domain coverage, achieving comparable results with respect to those of REP. For the solar plume data set, our method yields the best domain coverage. For the five critical points data set, $p(s)$ fails to select surrounding streamlines which correspond to



(a) our method (b) $p(s)$ Tao et al. [31] (c) $I(s;V)$ Tao et al. [31] (d) REP Tao et al. [31] (e) Xu et al. [41]

Fig. 13: Top to bottom: comparison of streamline selection results of the crayfish, solar plume, five critical points, tornado, and two swirls data sets using different methods. All methods show the same number of streamlines: 70, 100, 140, 60, and 80, respectively.

TABLE 3: Comparison of PSNR and AAD of reconstructed vector fields under different streamline selection methods. For each data set, the largest PSNR and the smallest AAD are highlighted in bold.

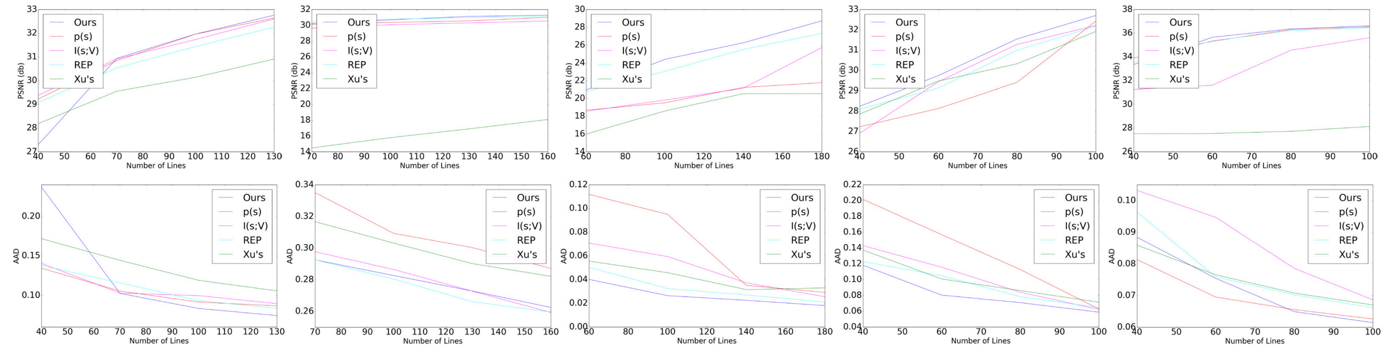
		PSNR (db)					AAD				
Data Set	# Lines	Ours	$p(s)$	$I(s;V)$	REP	Xu's	Ours	$p(s)$	$I(s;V)$	REP	Xu's
crayfish	70	**30.94**	30.84	30.91	30.02	28.97	**0.102**	0.105	0.103	0.116	0.144
solar plume	100	30.68	30.37	30.07	**30.75**	15.78	0.283	0.309	0.286	**0.280**	0.303
five critical pts	140	**26.25**	21.23	21.13	25.50	20.16	**0.023**	0.031	0.036	0.026	0.031
tornado	60	**29.74**	28.12	29.44	29.13	28.30	**0.080**	0.167	0.116	0.105	0.101
two swirls	80	**36.35**	36.30	34.55	36.21	27.72	**0.065**	0.066	0.079	0.070	0.071

the saddle pattern, while our method and REP best capture the source at the center of the volume. For the tornado data set, $I(s;V)$ gives the best result by revealing the swirling pattern surrounding the vortex core at the bottom. For the two swirls data set, our method strikes the best balance between domain coverage and feature highlighting. Overall, we feel that using FlowNet features generates competitive streamline selection results comparing to these state-of-the-art solutions.

To quantitatively evaluate the quality of selected streamlines, we use them to reconstruct the vector field $\mathbf{V}'$ through gradient vector flow [40]. We follow the work of Tao et al. [31] to initialize $\mathbf{V}'$ and iteratively refine $\mathbf{V}'$ using the generalized diffusion equations. After that, we compute the peak signal-to-noise ratio (PSNR) and average angle difference (AAD) of $\mathbf{V}'$ with respect to the original vector field $\mathbf{V}$. PSNR is defined as

$$\mathrm{PSNR}(\mathbf{V},\mathbf{V}') = 20\log_{10} I(\mathbf{V}) - 10\log_{10}\mathrm{MSE}(\mathbf{V},\mathbf{V}'), \quad (3)$$

where $I(\mathbf{V})$ is the difference between the maximum and minimum vector component values of $\mathbf{V}$, $\mathrm{MSE}(\mathbf{V},\mathbf{V}')$ is the mean squared error between $\mathbf{V}$ and $\mathbf{V}'$. For AAD, we calculate the angle difference between the original and



(a) crayfish (b) solar plume (c) five critical points (d) tornado (e) two swirls

Fig. 14: Comparison of PSNR (top row) and AAD (bottom row) of reconstructed vector fields under different streamline selection methods and different numbers of training streamlines.

reconstructed vectors for all voxels and then get the average. We normalize the error to $[0, 1]$ by dividing the AAD by π. A method is the best if it leads to the largest PSNR and the lowest AAD. In Table 3, under a given setting for the number of streamlines, we can see that our method achieves the highest PSNR and lowest AAD except for the solar plume data set. For that data set, REP achieves the best quality, while our method is the second best in terms of both PSNR and AAD. More extensive quality comparison results are given in Figure 14. We can see that in general, our method is the best under different streamline selection methods and different numbers of training streamlines. It is the clear winner for the five critical points and tornado data sets. For the crayfish data set, our method starts with the worst quality but ends with the best. For the solar plume data set, our method achieves the highest PSNRs very similar to REP, while loses to $I(s; V)$ and REP by a small margin in terms of AAD when the number of streamlines is larger than 140. For the two swirls data set, our method achieves the highest PNSRs very similar to $p(s)$ and REP, while the lowest AAD when the number of streamlines is larger than 80. We conclude that our method actually performs almost the best quantitatively compared to $p(s)$, $I(s; V)$, REP, and Xu's method.

5 Conclusions and Future Work

We have presented FlowNet, a novel approach for clustering and selection of streamlines and stream surfaces. Based on the encoder-decoder, FlowNet is able to learn latent features of streamlines and stream surfaces within a single framework in an unsupervised manner, which distinguishes itself from all previous works which have to solve them separately and explicitly utilize handcrafted features. These latent features encode the shape and location information of objects rather than the physical flow information. We then project the resulting feature vectors into a low-dimensional space, which lends itself to a visual mapping and interface for user interaction. Brushing and linking yields meaningful clustering and selection results. The line and surface clusters generated from FlowNet capture both spatial proximity and/or geometric similarity. We validate FlowNet using the network learned from the training set to examine the test set, and compare the results using FlowNet-trained features against those using other state-of-the-art methods.

To the best of our knowledge, our work is the first that applies deep learning to solve flow visualization problems. Our current work relies on downsampling binary streamline or stream surface volumes to make the network training possible in the GPU. In the future, we would like to explore a more efficient way that maps these 3D volumes to 2D images for network training. The 2D images could capture intrinsic volumetric information in a different space such as the spectrum space. This treatment allows us to handle much larger flow field data without sacrificing data resolution. Furthermore, we plan to explore how well FlowNet learned from streamlines or stream surfaces drawn from multiple data sets could capture cross-dataset latent features. Finally, we will develop a visual analytics interface that helps to make FlowNet explainable (for instance, identifying what about the learning process leads to clusters that can be explained with physical characteristics), which can also assist in network diagnosis and parameter tuning.

This promising direction offers other opportunities that are worth exploring. For example, could we design a deep neural net that learns the intricate relationships between the input and the output? In the context of flow visualization, the input could be a set of streamlines and the output could be representative stream surfaces, or the input could be a seeding curve and the output could be the quality of the corresponding stream surface. We would like to explore these opportunities in the future.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CNS-1629914, and DUE-1833129, and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki. GIFT: A real-time and scalable 3D shape search engine. In *CVPR*, pages 5023–5032, 2016.

[2] Y. Bengio. Learning deep architectures for AI. *Foundations Trends Mach. Learn.*, 2(1):1–127, 2009.

[3] J. M. Blondin and A. Mezzacappa. Pulsar spins from an instability in the accretion shock of supernovae. *Nature*, 445(7123):58–60, 2007.

[4] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Proc. Mag.*, 34(4):18–42, 2017.

[5] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *IEEE Vis*, pages 261–267, 1993.

[6] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, pages 705–712, 2003.

[7] T. Dombre, U. Frisch, J. M. Greene, M. Hénon, A. Mehr, and A. M. Soward. Chaotic streamlines in the ABC flows. *J Fluid Mech.*, 167:353–391, 1986.

[8] M. Edmunds, R. S. Laramee, R. Malki, I. Masters, N. Croft, G. Chen, and E. Zhang. Automatic stream surface seeding: A feature centered approach. *Comput. Graph. Forum*, 31(3):1095–1104, 2012.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[10] J. M. Esturo, M. Schulze, C. Rössl, and H. Theisel. Global selection of stream surfaces. *Comput. Graph. Forum*, 32(2):113–122, 2013.

[11] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, pages 484–499, 2016.

[12] P. Golik, P. Doetsch, and H. Ney. Cross-entropy vs. squared error training: A theoretical and experimental comparison. In *INTERSPEECH*, pages 1756–1760, 2013.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[14] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE TVCG*, 2018. Accepted.

[15] F. Hong, C. Lai, H. Guo, E. Shen, X. Yuan, and S. Li. FLDA: Latent Dirichlet allocation based unsteady flow analysis. *IEEE TVCG*, 20(12):2545–2554, 2014.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[19] K. Lu, A. Chaudhuri, T.-Y. Lee, H.-W. Shen, and P. C. Wong. Exploring vector fields with distribution-based streamline analysis. In *PacificVis*, pages 257–264, 2013.

[20] J. Ma, C. Wang, and C.-K. Shene. Coherent view-dependent streamline selection for importance-driven flow visualization. In *IS&T/SPIE VDA*, pages 865407–1–865407–15, 2013.

[21] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3D vector fields. *IEEE TVCG*, 16(6):1578–1586, 2010.

[22] F. Milletari, N. Navab, and S.-A. Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *IEEE 3D Vision*, pages 565–571, 2016.

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pages 807–814, 2010.

[24] S. Oeltze, D. J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim. Blood flow clustering and applications in virtual stenting of intracranial aneurysms. *IEEE TVCG*, 20(5):686–701, 2014.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, pages 77–85, 2017.

[26] M. P. Rast. The scales of granulation, mesogranulation, and supergranulation. *The Astrophysical Journal*, 597(2):1200–1210, 2003.

[27] G. Riegler, A. O. Ulusoy, and A. Geiger. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*, pages 6620–6629, 2017.

[28] C. Rössl and H. Theisel. Streamline embedding for 3D vector field exploration. *IEEE TVCG*, 18(3):407–420, 2012.

[29] M. Schulze, J. Martinez Esturo, T. Günther, C. Rössl, H.-P. Seidel, T. Weinkauf, and H. Theisel. Sets of globally optimal stream surfaces for flow visualization. *Comput. Graph. Forum*, 33(3):1–10, 2014.

[30] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, pages 945–953, 2015.

[31] J. Tao, J. Ma, C. Wang, and C.-K. Shene. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE TVCG*, 19(3):393–406, 2013.

[32] J. Tao and C. Wang. Semi-automatic generation of stream surfaces via sketching. *IEEE TVCG*, 24(9):2622–2635, 2018.

[33] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[34] L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-SNE. *J Mach. Learn. Res.*, 9:2579–2605, 2008.

[35] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE TVCG*, 14(6):1396–1403, 2008.

[36] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM TOG*, 36(4):72–1–72–11, 2017.

[37] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *PacificVis*, pages 129–136, 2010.

[38] D. Weiskopf, T. Schafhitzel, and T. Ertl. Real-time advection and volumetric illumination for the visualization of 3D unsteady flow. In *VisSym*, pages 13–20, 2005.

[39] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shape modeling. In *CVPR*, pages 1912–1920, 2015.

[40] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *CVPR*, pages 66–71, 1997.

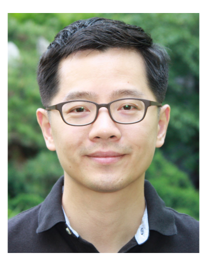
[41] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *IEEE TVCG*, 16(6):1216–1224, 2010.

[42] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *IEEE Vis*, pages 471–478, 2005.

[43] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen. Hierarchical streamline bundles. *IEEE TVCG*, 18(8):1353–1367, 2012.

Jun Han is a PhD student at University of Notre Dame. He received a BS degree in software engineering and a MS degree in computer software and theory in 2014 and 2017, respectively. Both degrees are from Xidian University. His current research focuses on applying deep learning techniques to solve data visualization problems.

Jun Tao is a postdoctoral researcher at University of Notre Dame. He received a Ph.D. degree in computer science from Michigan Technological University in 2015. Dr. Tao's major research interest is scientific visualization, especially on applying information theory, optimization techniques, and topological analysis to flow visualization and multivariate data exploration. He will join Sun Yat-sen University as an associate professor in October 2018.

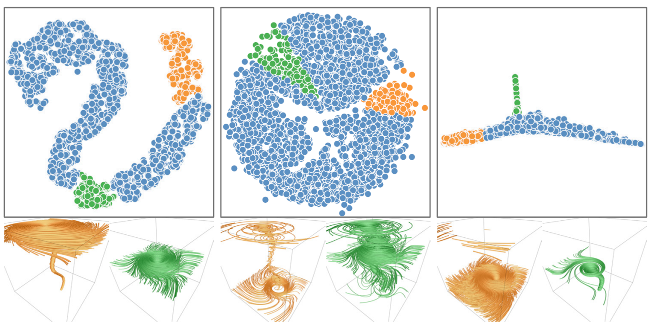


Chaoli Wang is an associate professor of computer science and engineering at University of Notre Dame. He received a Ph.D. degree in computer and information science from The Ohio State University in 2006. Dr. Wang's main research interest is data visualization, in particular on the topics of time-varying multivariate data visualization, flow visualization, as well as information-theoretic algorithms, graph-based techniques, and deep learning solutions for big data analytics.

APPENDIX

1 CHOOSING DIMENSIONALITY REDUCTION AND OBJECT CLUSTERING METHODS

Dimensionality reduction. To project feature descriptors to 2D, we experiment with three popular dimensionality reduction methods: t-SNE [34], MDS [18], and Isomap [33]. Among them, t-SNE is a *neighborhood-preserving* method while MDS and Isomap are *distance-preserving* methods. t-SNE is expected to perform the best because it preserves the local neighborhood of the data by minimizing the Kullback-Leibler divergence between the distribution over the high-dimensional data and the distribution over their low-dimensional projections. MDS performs low-dimensional embedding based on the pairwise Euclidean distance between data points. Isomap uses the geodesic distance induced by a neighborhood graph embedded in MDS. In addition, when the dimensionality is high (in our case, 1024), distance-preserving methods may not work as well as neighborhood-preserving methods [6], [34].

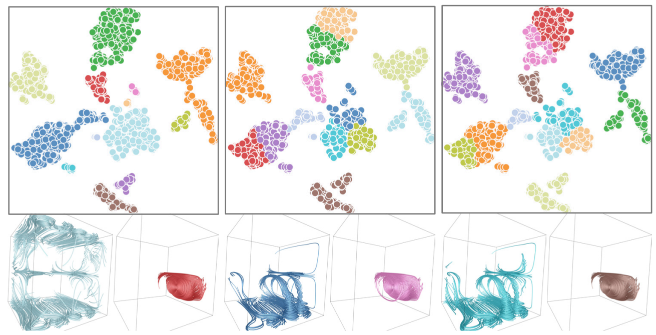


(a) t-SNE (b) MDS (c) Isomap

Fig. 1: Comparison of different dimensionality reduction methods via brushing and linking using the tornado data set. Two selected point groups and their corresponding streamlines are in orange and green. The unselected points are in blue.

In Figure 1, we visualize these feature descriptors using the three methods in order to identify the most desirable one. For each projection, we brush two groups of points and show their corresponding streamlines. We find that t-SNE best preserves the similarity structure (i.e., similar streamlines should be projected to neighboring points). In (a), we can see a clear 1D 'S'-shape layout of the feature descriptors. Brushing and liking shows that this 1D direction corresponds to streamlines from top to bottom of the tornado. This is not the case for (b) and (c). In (b), the MDS view shows a nearly uniform distribution of points. Both the orange and green points include streamlines at the top of the tornado, but these points are separated on different sides in the projection. Compared with (a), streamlines within the same local neighborhood in (b) are less similar. In (c), the orange cluster shown in the Isomap view corresponds to streamlines at both ends, which is not desirable.

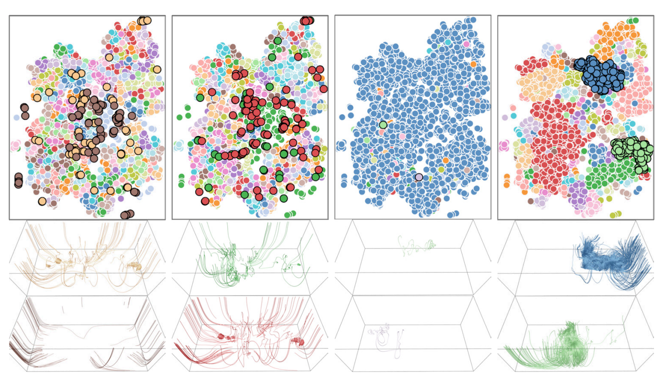
Object clustering. After projecting these feature descriptors to 2D, we explore three types (i.e., *density-based*, *partition-based*, and *hierarchy-based*) of clustering algorithms to group similar objects. Based on the t-SNE projection results, density-based algorithms could be the best choice. Partition-based algorithms aim to divide the data points into clusters with similar sizes but this guideline may not be ideal for all kinds of data sets. Hierarchy-based algorithms build a tree to store all data points and group them through the parent-child relationship but the hierarchical structure may not always be the desirable representation for all kinds of data sets. Density-based algorithms group the data points through projection density, meaning that data points located around a small local neighborhood will be grouped together, which is exactly what the t-SNE projection shows.



(a) DBSCAN (b) k-means (c) agglomerative

Fig. 2: Comparison of different clustering algorithms along with t-SNE projection using the two swirls data set. Each produces 13 clusters and two streamline clusters are shown.

In Figure 2, we show the clustering results using three widely used clustering algorithms: DBSCAN (density-based), k-means (partition-based), and agglomerative clustering (hierarchy-based), to identify the best one. For all three clustering algorithms, the distance between two data points is computed as their Euclidean distance in the projection view. We use Ward's minimum variance method for agglomerative clustering. All three cases use the same t-SNE projection and yield 13 clusters. Among them, we find that DBSCAN achieves the best clustering results. In (a), we can observe that the cyan cluster corresponds well to boundary streamlines. This is not the case for (b) and (c) as this cluster is separated into smaller ones and both boundary and inner streamlines are grouped into the same cluster.



(a) FS AM (b) FS KM (c) FS DS (d) TS DS

Fig. 3: Comparison of different clustering algorithms along with t-SNE projection using the crayfish data set. FS = feature space. TS = t-SNE space. AM = agglomerative. KM = k-means. DS = DBSCAN. Each produces 156 clusters and two streamline clusters are shown.

We also conduct a comparative study to justify clustering feature vectors in the projection space (in our case, the t-SNE space) rather than clustering directly in the feature space. From Figure 3, we can see that the results of clustering directly in the feature space are not satisfactory, regardless of which clustering method is employed: DBSCAN generates the worst results while k-means and agglomerative clustering generate similar results. However, all of them cannot group similar streamlines together. One possible explanation is that in the feature space, data are rather sparse, which is problematic for any method that requires statistical significance. Moreover, these data are dissimilar in many different ways, which prevents the traditional clustering algorithm from working efficiently. By clustering feature vectors in the t-SNE space, we keep all important information and decompose co-related factors, which ensures that the clustering algorithm can work well. In addition, projection-space clustering guarantees that points close to each other are clustered together, which clearly is not the case if clustering is performed in the feature space.

Therefore, we choose the combination of t-SNE and DBSCAN for dimensionality reduction and object clustering, and perform clustering of feature vectors in the t-SNE space. Unless otherwise stated, all results presented in the paper use this combination.