Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

SurfNet: Learning Surface Representations via Graph Convolutional Network

Jun Han and Chaoli Wang
University of Notre Dame

Abstract

For scientific visualization applications, understanding the structure of a single surface (e.g., stream surface, isosurface) and selecting representative surfaces play a crucial role. In response, we propose SurfNet, a graph-based deep learning approach for representing a surface locally at the node level and globally at the surface level. By treating surfaces as graphs, we leverage a graph convolutional network to learn node embedding on a surface. To make the learned embedding effective, we consider various pieces of information (e.g., position, normal, velocity) for network input and investigate multiple losses. Furthermore, we apply dimensionality reduction to transform the learned embeddings into 2D space for understanding and exploration. To demonstrate the effectiveness of SurfNet, we evaluate the embeddings in node clustering (node-level) and surface selection (surface-level) tasks. We compare SurfNet against state-of-the-art node embedding approaches and surface selection methods. We also demonstrate the superiority of SurfNet by comparing it against a spectral-based mesh segmentation approach. The results show that SurfNet can learn better representations at the node and surface levels with less training time and fewer training samples while generating comparable or better clustering and selection results.

CCS Concepts

• ***Computing methodologies*** → *Neural networks; Unsupervised learning;* • ***Human-centered computing*** → *Scientific visualization;*

1. Introduction

Analyzing various kinds of surfaces, such as stream surfaces and isosurfaces produced from 3D vector and scalar fields, is critically important for many applications. In this context, *nodes* represent sample tracing points forming a stream surface or triangle vertices constituting an isosurface. Grouping nodes on stream surfaces can help us, for example, better understand the airflow at the wingtip of an aircraft that traces the wake vortex. Partitioning nodes on isosurfaces can help us, for example, quickly detect different parts of a human body. Meanwhile, identifying representative surfaces can yield a compact representation of the underlying data and lead to the efficient and effective analysis and visualization. Therefore, we propose a node embedding solution for studying stream surfaces and isosurfaces generated from scientific visualization. Our goal is to investigate the relationship among nodes on a single surface and the correlation among multiple surfaces.

Achieving node clustering and surface selection needs to answer three key questions. First, *how to represent surfaces so that both node- and surface-level information can be extracted*? Although multiview-based [SMKLM15, QSN*16] and voxel-based representations [WSK*15, HTW20] have been proposed to formulate a surface, these representations can only extract *surface-level* information (such as the overall shape) rather than *node-level* information (such as individual sources or sinks) since no node information is built. Second, *how to group nodes on a surface without any given labels*? Unlike 3D objects, where many node labels are supported, it is impractical to manually label every node on the surface since this process is extremely time-consuming and expensive. Third, *how to derive surface embeddings from a set of node embeddings*? Although several existing approaches [YYM*18, HHF*19] can directly generate surface embeddings, they require labeling each graph, which is not suitable in our scenario as we need to handle a large set of surfaces (e.g., 1,000).

To respond, we apply a geometric representation [BBL*17] (i.e., meshes) for surfaces since it can preserve geometric information of the nodes. We introduce SurfNet, a deep learning approach for embedding nodes on surfaces. These learned node embeddings support node clustering, surface clustering, filtering, and selection. The crux of SurfNet is the design of a *graph convolutional network* (GCN) and a novel loss that can automatically learn the hidden embedding of every node on a single surface in an *unsupervised* fashion. Specifically, we train SurfNet to learn an end-to-end function that maps a node with only its information (e.g., position, normal) to a node embedding with its neighborhood information. The trained model allows us to explore the relationship among different nodes on a single surface or the relationship among different surfaces. Our approach consists of three major steps:

- **Data sample generation.** Given a vector or scalar field data

set, we produce stream surfaces using random seeding curves following the binormal directions [TW18] or isosurfaces with uniformly sampled isovalues. The resulting surface samples are used for training and testing SurfNet.

- **SurfNet training.** Given the surfaces, we first map them to undirected unweighted graphs and initialize node information in every graph using their positions, normals, or velocities. Then we feed these graphs into SurfNet that maps these node embeddings to new node embeddings, which can capture the graph's *spatial* and *geometric* structures. Moreover, we optimize SurfNet with a loss function that measures node similarity.
- **Interactive exploration.** With the trained SurfNet, we develop a visual interface supporting interactive surface exploration and analysis from two perspectives. First, we can analyze a single surface's structure through automatically partitioning node embeddings generated by SurfNet. Second, we can select representative surfaces and measure surface similarity via grouping surface embeddings aggregated from node embeddings.

We demonstrate the effectiveness of SurfNet with stream surfaces and isosurfaces generated from different vector and scalar data sets. We qualitatively compare the clustering results of SurfNet against other node embedding methods and the representative selection results against other surface selection methods. We also quantitatively show that we can use the representatives selected by SurfNet to reconstruct the original vector and scalar field data with better quality. Furthermore, we study the impact of different hyperparameters of SurfNet (e.g., loss function design and node information initialization) and make recommendations. In summary, our contributions are as follows:

- We propose SurfNet that embeds node and surface information for scientific visualization applications.
- We investigate several unsupervised loss functions and propose a geodesic loss for network optimization.
- We conduct an extensive study to understand the learned representations on node clustering and surface selection tasks.

2. Related Work

Deep learning for scientific visualization. With the tremendous success of deep learning techniques in computer vision, natural language processing, and robotics, the scientific visualization community has begun to leverage deep neural networks (DNNs) to solve various volume and flow visualization problems.

For volume visualization, Han and Wang [HW20b] introduced TSR-TVD that combines the recurrent neural network and generative adversarial network (GAN) to upscales time-varying volumetric data in the *temporal* dimension and designed SSR-TVD [HW20a] that upscales time-varying data in the *spatial* dimension. Han et al. [HZ*22] further presented STNet, an end-to-end solution for *spatiotemporal* upscaling. Weiss et al. [WCTW21] proposed a convolutional neural network (CNN) in the image space, mapping low-resolution isosurfaces to high-resolution ones. Cheng et al. [CCJ*19] presented an interactive system based on deep learning for detecting and discovering complicated volumetric structures. Berger et al. [BLL19] generated various rendering images based on different viewpoints and transfer functions through a GAN. Similar work on DNN-supported volume visualization was done by Hong et al. [HLY19]. He et al. [HWG*20] proposed InSituNet that gathers rendering images from ensemble simulations for offline training and enables interactive exploration in a post hoc manner. Han et al. [HZX*21] presented V2V, a variable-to-variable translation framework that translates a variable sequence to another one for multivariate time-varying data. Tkachev et al. [TFE21a, TFE21b] developed machine learning solutions for identifying spatiotemporal similarities for volume visualization. Weiss et al. [WITW20] generated a sparse adaptive sampling structure from a given low-resolution isosurface or direct volume rendering image for reconstructing a high-resolution one. Shi et al. [SXW*22] designed GNN-Surrogate, a hierarchical and adaptive graph neural network for generating unstructured grid data.

For flow visualization, Hong et al. [HZY18] leveraged long short-term memory to estimate data access patterns and reduce I/O latency for parallel particle tracing. Han et al. [HTZ*19] and Gu et al. [GHCW21] proposed vector field reconstruction algorithms that take a set of representative streamlines to synthesize high-quality vector fields. Han et al. [HTW20] designed FlowNet that generates the latent representations of streamlines or stream surfaces and utilizes them for clustering and selection tasks. Guo et al. [GYH*20] presented SSR-VFD, a machine learning solution to upscale a 3D vector field by 64 or 512 times. Han and Wang [HW22] developed TSR-VFD that generates temporal super-resolution for unsteady vector fields. Jakob et al. [JGG21] trained CNNs on a large 2D fluid flow data set to provide a benchmark for the post-analysis of Lagrangian fluid flow. Gu et al. [GHCW22] designed Scalar2Vec that translates scalar fields to velocity vector fields.

Our work is similar to FlowNet [HTW20]; however, there are several differences. First, instead of converting surfaces into binary volumes as input, we directly use the original surface mesh as input. This mesh representation can capture spatial, and more importantly, geometric information. Second, we leverage GCN instead of CNN to speed up training and achieve better performance. Third, we aim to select representative surfaces and group nodes on a single surface for understanding different node patterns.

Geometric deep learning. Geometric deep learning can be classified into two categories: spectral and spectral-free methods [BBL*17]. Spectral methods compute eigenvectors of graph Laplacian and operate on the graph's spectrum. For example, Monti et al. [MBM*17] designed Gaussian mixture model convolution (GMMConv) to graphs and meshes for learning task-specific representations. Yi et al. [YSGG17] introduced SynSpecCNN that parameterizes kernels in the spectral domain spanned by graph Laplacian eigenbases for keypoint detection and part clustering. Ranjan et al. [RBSB18] proposed a spectral convolutional network that can capture non-linear variations of shapes for 3D face generation. Liu et al. [LJQ19] utilized convolution operating on one-ring neighbors of each node in mesh for the classification of impairment and disease. Shu et al. [SQX*16] applied autoencoders to transform low-level mesh features into high-level ones and grouped the high-level features to co-segment 3D shapes.

Spectral-free methods implicitly learn to aggregate information on the graph. For instance, Litany et al. [LBBM18] proposed a variational graph convolutional autoencoder that learns hidden representations of meshes to complete partial shapes. Smith et al. [SFRM19] designed GEOMetrics, a framework based on graph

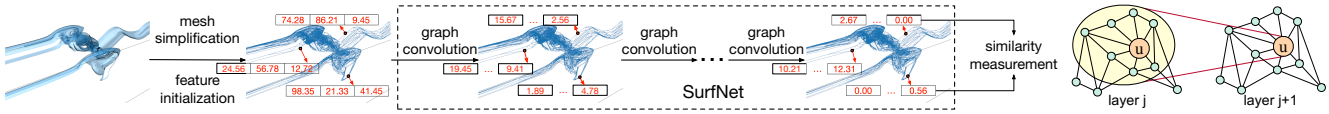


Figure 1: *(a) SurfNet for node embedding learning. The input to SurfNet is the simplified surface. Several graph convolutions are leveraged to learn node embeddings. Finally, a similarity measure is applied to optimize SurfNet. (b) An example of embedding propagation in GCN. The latent embedding for node u at layer $j+1$ is aggregated from its previous embedding and immediate neighbors at layer j.*

convolution, for reconstructing mesh objects. Yao et al. [YYSZ20] applied GCNs to learn motions from meshes for face reenactment. Kostrikov et al. [KJP*18] built Surface Networks, leveraging Dirac operator on meshes, for temporal prediction of mesh deformations and mesh generation. Wang et al. [WSL*19] proposed EdgeConv, a convolutional operation acting on point clouds, to handle high-level classification and clustering tasks. Wang et al. [WZL*18] introduced Pixel2Mesh that generates meshes from single RGB images based on GCNs. Hanocka et al. [HHF*19] established MeshCNN that utilizes geodesic connections among edges to process meshes for mesh classification and clustering.

Our work is a spectral-free method. Compared with spectral methods utilizing linear operations, spectral-free methods apply polynomial estimation on the mesh, capturing more complicated patterns. SurfNet is different from the above works in two ways. First, instead of requiring a large number of annotations at either the mesh level [HHF*19, MBM*17, LJQ19] or the node level [HHF*19, YSGG17, WSL*19], we propose a novel unsupervised loss that helps SurfNet automatically group nodes. Second, we present an unsupervised node aggregation operation that generates mesh embedding in our context from a set of node embeddings.

3. SurfNet

Given a large set of stream surfaces or isosurfaces generated from a vector or scalar field data set, we propose two aims: clustering and selection. For clustering, we aim to partition every stream surface or isosurface into several parts so that these parts exhibit different patterns. For selection, a subset of surfaces best covering the underlying features and patterns needs to be identified. Instead of identifying these surfaces directly, we group the input set into clusters and select the representatives from these clusters. A key question is how to generate node embeddings for a surface in an unsupervised manner. We propose SurfNet, a GCN that learns node embeddings through graph convolution. An overview of SurfNet is sketched in Figure 1 (a). We first simplify the surfaces using a mesh simplification algorithm [GH97], an iterative approach to simplify surfaces and maintain surface errors through a given threshold for computation reduction. Then, every simplified surface is treated as an undirected unweighted graph, which will be the input to SurfNet. We initialize the node using their positional information. SurfNet learns node embeddings automatically by aggregating their neighborhoods. To optimize SurfNet, we compute a node similarity matrix, and based on this matrix, SurfNet will update learnable parameters through gradient descent. Once SurfNet converges, these node embeddings represent each node's information or the entire surface's information. We then apply t-SNE [vdMH08] to node embeddings for projecting to a low-dimensional space and leverage DBSCAN [EKSX96] to identify the clustering or the representatives based on interactive clustering (t-SNE+DBSCAN has shown to produce the most meaningful clustering results [HTW20]). Finally, users can explore the surface(s) and perform visual analysis and analytical reasoning through a visual interface.

3.1. Notation

Formally, we define a surface (mesh) as a graph $G = (V, E)$, where V is the set of all nodes (e.g., sample tracing points along both streamline and timeline directions on a stream surface or triangle vertices on an isosurface) and E is the set of all edges (e.g., edges connecting sample tracing points on a stream surface or triangle edges on an isosurface). Each node $u \in V$ has a set of associated embedding values, denoted as $\mathbf{F}_u$. $\mathbf{F}_u$ could denote various pieces of information (e.g., position, normal, velocity) of the corresponding node u. $\mathcal{N}(u)$ denotes the neighborhood of u. In addition, $\mathbf{G} = \{G_1, G_2, \ldots, G_n\}$ is a set of graphs and $\mathbf{F}^{G_i}$ represents the embedding descriptor of graph G_i.

3.2. Node Embedding

Given a surface, SurfNet aims to generate a set of node embeddings, and each node embedding presents rich spatial and geometric information of the corresponding node on the surface. To this end, SurfNet needs to satisfy the two major properties as suggested by Bai et al. [BDQ*19]:

- **Inductivity.** SurfNet should learn a unified mapping function so that it can be directly applied to any unseen surface to generate the corresponding node embeddings.
- **Permutation invariance.** A different adjacency matrix can represent the same surface by permuting the order of nodes, and SurfNet should be insensitive to such permutations.

Across different node embedding models, neighborhood aggregation methods based on GCN are permutation-invariant and inductive. This is because the core operation, graph convolution, updates a node's representation by aggregating the node and its neighbors' embedding. Since the aggregation function treats a node's neighbors as a set, the order does not affect the final embedding result.

Node aggregation. To merge information on a node u, we choose a topology-based aggregator [YHC*18], as sketched in Figure 1 (b). The aggregator works as follows. For each node u on a surface, all nodes v, where $v \in \mathcal{N}(u)$, are first identified. Then, a multi-perceptron layer is leveraged to transform the node representation $\mathbf{F}_v$ into a new node representation through an aggregation function (e.g., averaging or summation). This aggregation function merges representations $\mathbf{F}_u$ and $\mathbf{F}_v$, where $v \in \mathcal{N}(u)$. After that, we apply an activation function to the transformed embedding to increase the capability of capturing nonlinear behaviors. Finally, a

representation of u consisting of the node and its local neighborhood's information is produced.

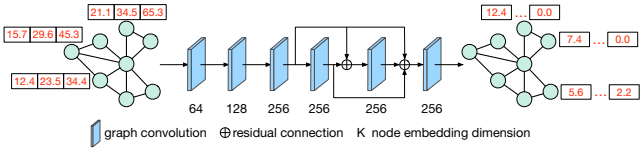


Figure 2: *The architecture of SurfNet. SurfNet contains six graph convolution layers and the last four graph convolutions are bridged by residual connection.*

Network architecture. As sketched in Figure 2, following Ying et al. [YHC*18], after applying one graph convolution to a surface, we update the embedding representation of each surface node u. We employ several graph convolutions to aggregate enough information from u's neighbors. Specifically, SurfNet contains six graph convolution layers. These six layers embed the embeddings into 64, 128, 256, 256, 256, and 256 dimensions, respectively. Moreover, we leverage residual connection [HZRS16] to bridge the information from the third to fifth graph convolution layers, mitigating the impact of vanishing gradients and accelerating convergence [XZJK21]. We apply the rectified linear unit (ReLU) [NH10] as the activation function after each graph convolution layer in the following computation. Note that the initial representations (i.e., the input to the first layer) could be the original information of the surface nodes (e.g., position, normal). The detailed parameters are listed in Table 1. Note that we increase the embedding dimension in network design so that the learned embedding have enough information to represent nodes.

Table 1: *Network parameter details of SurfNet.*

operation	# input neurons	# output neurons
graph Conv	3	64
graph Conv	64	128
graph Conv	128	256
graph Conv	256	256
graph Conv	256	256
graph Conv	256	256

3.3. Loss Function

To optimize SurfNet, we investigate several loss functions for measuring node similarity in the embedding space.

- Optimization based adjacency matrix [ASN*13]:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{u \in V_i} \sum_{v \in V_i} (\mathbf{A}_i(u,v) - \mathbf{F}_u^T \mathbf{F}_v)^2, \tag{1}$$

  where n is the number of graph samples (i.e., $\{G_1, G_2, \ldots, G_n\}$) used for training, V_i is the node set of G_i, and $\mathbf{A}_i$ is the adjacency matrix of G_i.
- Optimization based on random walk [HYL17]:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{u \in V_i} \left[-\log(\sigma(\mathbf{F}_u^T \mathbf{F}_v)) - \sum_{k \in P_u} \log(\sigma(\mathbf{F}_u^T \mathbf{F}_k)) \right], \tag{2}$$

  where v is a node that is near u on a fix-length random walk (e.g., 10), σ is the sigmoid function, and P_u is a negative sampling node set of node u.
- Optimization based on multidimensional scaling (MDS): Following Corso et al. [CYP*21], which utilizes MDS to estimate the error between the sequences' edit distance and the distance between the learned embeddings, we apply MDS as a self-supervised loss as follows

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{u \in V_i} \sum_{v \in V_i} (\mathcal{D}(u,v) - d(\mathbf{F}_u, \mathbf{F}_v))^2, \tag{3}$$

  where $d(\cdot,\cdot)$ denotes the distance measure (such as L_2 norm) in the embedding space and $\mathcal{D}(u,v)$ is a distance metric between nodes u and v (e.g., position, normal, or velocity similarity). When we use geodesic distance to estimate the learned node embeddings, the loss becomes ISOMAP.

Loss analysis. To investigate the effectiveness of various losses, we train SurfNet with different objective functions (i.e., adjacency matrix, random walk, Euclidean, normal, velocity, and geodesic distances) using the five critical points data set. The results are given in Figure 3. Node clustering results are not satisfactory using the adjacency matrix, random walk, normal, velocity, and curature distances. For example, these loss functions cannot isolate the spirals on this surface. Euclidean distance can detect the spiral at the top of the surface (i.e., the yellow part) but fails to discover the other one. The geodesic distance can separate the two spirals at both ends (the yellow and blue parts). Therefore, in the paper, we choose geodesic distance as the objective function for SurfNet optimization.

We further reason these losses during network optimization. As illustrated with an example in Figure 4, Euclidean distance deems red and yellow nodes closer than red and purple nodes. This contrasts the observation that red and purple nodes should be more similar since they come from the same surface branch, while red and yellow nodes are less similar as they reside in two different branches. For adjacency matrix and random walk, both determine that the red node is *equally* distant from the green and yellow ones. The red node is disconnected from the yellow or green node in the adjacency matrix. The red node cannot reach the yellow or green node through multiple random walks. Only the loss function based on the geodesic distance reflects the correct similarity order.

3.4. Surface Embedding

For surface G_i, given a set of node embeddings $\mathbf{F} = \{\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_{|V_i|}\}$, we compute the corresponding surface embedding as

$$\mathbf{F}^{G_i} = \frac{1}{|V_i|} \sum_{u \in V_i} \mathbf{F}_u, \tag{4}$$

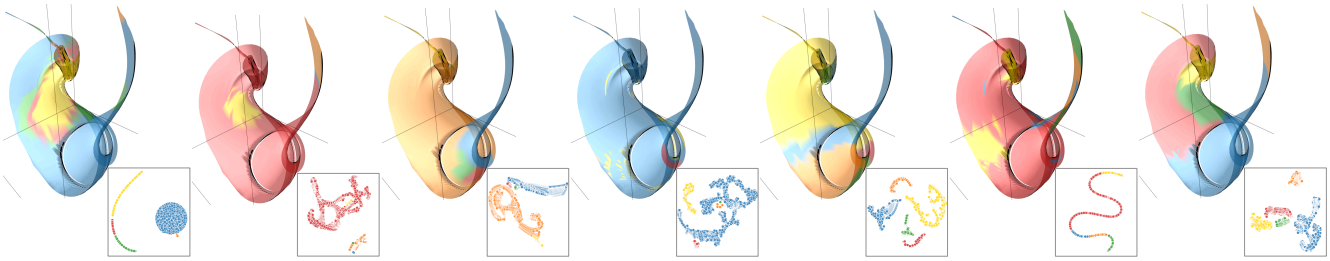
where $|V_i|$ is the number of nodes in G_i.

Similar to Tkachev et al. [TFE21b], the distance between surfaces G_i and G_j is defined as

$$\mathcal{D}(G_i, G_j) = ||\mathbf{F}^{G_i} - \mathbf{F}^{G_j}||_2, \tag{5}$$

where $||\cdot||_2$ is L_2 norm.

To demonstrate the effectiveness of this surface distance metric, we compare this metric with two traditional surface distance metrics, i.e., chamfer distance [BTBW77] (CD), Jensen-Shannon



(a) adj. matrix (b) random walk (c) Euclidean (d) normal (e) velocity (f) curvature (h) geodesic

Figure 3: *SurfNet node clustering results of a stream surface under different loss functions using the five critical point data set.*

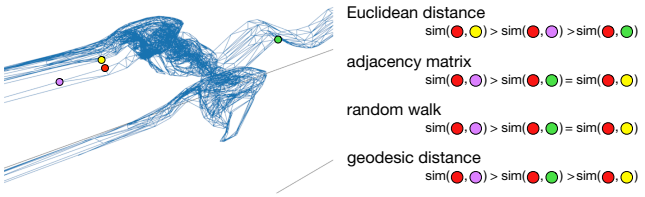


Figure 4: *Comparison of different loss functions. The correct similarity order is sim(●,●)>sim(●,●)>sim(●,○). This is because red, purple, and green nodes are on the same surface branch while red and yellow nodes are on two different branches. In addition, red and purple nodes are closer on the surface than red and green nodes.*

divergence [Lin91] (JSD), and Jaccard distance [FML15] (JD). A comparison is shown in Figure 5. In the projection space generated by each distance metric, we brush two parts and display the corresponding stream surfaces. For CD, JSD, and JD, similar surfaces are not placed in close locations (i.e., the purple surfaces). For SurfNet, we can observe that the surfaces can be classified into four groups in general and similar surfaces can be placed together.

3.5. Interface and Interaction

The screenshot in Figure 6 shows that our SurfNet interface includes two views: *surface view* and *projection view*. Brushing and linking are used to connect both views. Surface(s) in the original 3D space is shown in the surface view, and the projected points in the 2D space are displayed in the projection view. Note that for node clustering, each point in the projection view represents a *node* on the surface (*node embedding*). For representative selection, each point represents a *surface* (*surface embedding*). The following interactive functions are supported to explore the clustering and selection results.

- **Clustering.** The hyperparameters of DBSCAN (i.e., the maximum distance between two node embeddings in the 2D space and the minimum number of samples in one cluster) can be tuned by users to produce different clustering results. Neighboring clusters are drawn using different colors for differentiation. A black boundary is added to the selected cluster for highlighting, and the corresponding surface parts or surfaces are displayed in the surface view. Multiple clusters in the projection view can be chosen simultaneously by users. The relationships among them are examined in the surface view. The unselected ones are colored with light gray.
- **Representatives.** To select the representative from one cluster, we follow FlowNet [HTW20] and define a cluster's center as the point where the average distance is the shortest between this point and all the other points in this cluster. The number of representatives can be manually changed based on user preferences. The selected representatives are displayed in the surface view, and the corresponding points are shown in the projection view.
- **Neighborhood.** Following FlowNet [HTW20] , we define the distance between two clusters as the distance between their centers. Users can select one cluster and expand to its neighborhood to explore the neighboring clusters. Users can also go through these clusters according to their distances to the selected one. They can check the similarities and differences among the clusters in the neighborhood.

Table 2: *Dimension and training epochs of vector data sets.*

data set	dimension ($x \times y \times z$)	epochs
Bénard flow	$128 \times 32 \times 64$	200
five critical points	$51 \times 51 \times 51$	200
solar plume	$126 \times 126 \times 512$	200
square cylinder	$192 \times 64 \times 48$	200
tornado	$64 \times 64 \times 64$	200
two swirls	$64 \times 64 \times 64$	200

Table 3: *Dimension and training epochs of scalar data sets.*

data set	variable	dimension ($x \times y \times z$)	epochs
combustion	HR, MF, YOH	$240 \times 360 \times 30$	100
ionization	H, He, PD	$600 \times 248 \times 248$	100
bonsai	Intensity	$204 \times 204 \times 204$	100
brain	Intensity	$128 \times 128 \times 72$	200
lobster	Intensity	$301 \times 324 \times 56$	300

4. Results and Discussion

4.1. Data Sets and Network Training

We experimented with vector and scalar data sets shown in Tables 2 and 3, respectively. We used two multivariate data sets (combustion and ionization), and the rest of the data sets have a single variable. We implemented SurfNet using PyTorch and DGL [WZY*19]. The training and inference were run on an NVIDIA GTX 1080 Ti GPU. In terms of optimization, we initialized SurfNet parameters following the suggestion of He et al. [HZRS15] and employed the Adam optimizer [KB14] for parameter updates ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We set one training sample per mini-batch and the learning rate to 10^{-4}. For stream surfaces, we generated $2,000$ surfaces. We

(a) selected (b) chamfer dist. (c) Jensen-Shannon div. (d) Jaccard dist. (e) SurfNet

Figure 5: *Evaluation of different surface distance metrics using the two swirls data set via brushing and linking. (a) shows the selected surfaces, while (b) to (e) show their corresponding projections where the unselected nodes are colored in gray.*

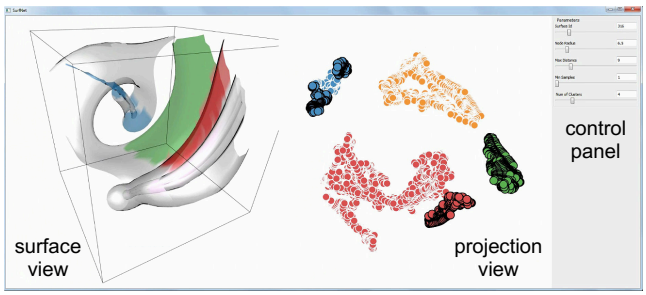


Figure 6: *The visual interface of SurfNet. In this example, the selected nodes are highlighted in both views. The unhighlighted surface parts are colored with light gray in the surface view.*

used 1,000 surfaces for training and the other 1,000 for inference. For isosurfaces, we uniformly selected 256 isovalues for training and sampled other isovalues for inference. We determined all these hyperparameters empirically. We use spatial location as initialized node information for SurfNet.

To evaluate SurfNet, we analyze different hyperparameter settings, including mesh simplification, network depth, training stability, embedding strategy, embedding dimension, node information initialization, and training samples. Please refer to the appendix for SurfNet analysis. The appendix also includes a comparison of dimensionality reduction and clustering methods.

Evaluation metric. For quantitative evaluation, following Han et al. [HTW20], we use the representative surfaces to reconstruct the vector field (for steam surfaces) and the scalar field (for isosurfaces) through gradient vector flow [XP97]. We then compute the peak signal-to-noise ratio (PSNR) between the reconstructed and original data.

4.2. Node Clustering

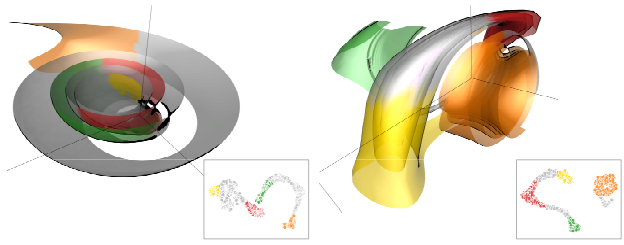
Baselines. For node clustering, we compare SurfNet against both spectral and spectral-free methods:

- GMMConv [MBM*17] is a spectral method. It learns d-dimensional node embedding through Gaussian mixture CNNs.
- EdgeConv [WSL*19] is a spectral-free method. It encodes each node into a d-dimensional vector by aggregating its nearest neighbors.

For a fair comparison, we apply the same settings (i.e., the loss function, number of training samples, optimizer, and epochs) for optimizing GMMConv, EdgeConv, and SurfNet.

Similar to other node embedding approaches [JZL*19, KW17, AEHPARA18, HLL*19], the 2D t-SNE projection cannot completely represent relationships among different nodes.

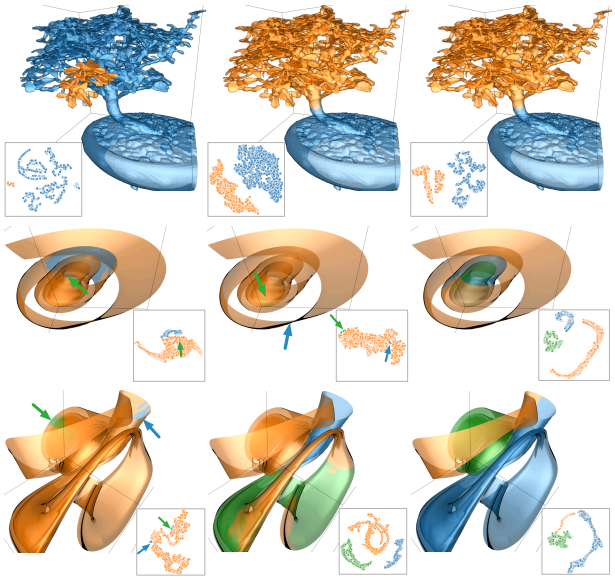
Node embedding validation. To verify the effectiveness of our node embedding, we apply t-SNE to project node embeddings to a 2D space and then perform brushing and linking. The results are shown in Figure 7. The t-SNE projection conveys the node's positional information and potentially the surface's structural information. For example, for the tornado (Figure 7 (a)), the orange points at the bottom-right corner of the projection view correspond to the tail of the surface, and the green and red points correspond to the spiral. Likewise, for the two swirls (Figure 7 (b)), the orange points in the projection view exhibit a swirling pattern, which corresponds to the orange part of the surface. These brushing and linking results demonstrate the meaningfulness of the learned embeddings, indicating that they can capture the nodes' neighborhoods and their positional information.



(a) tornado (b) two swirls

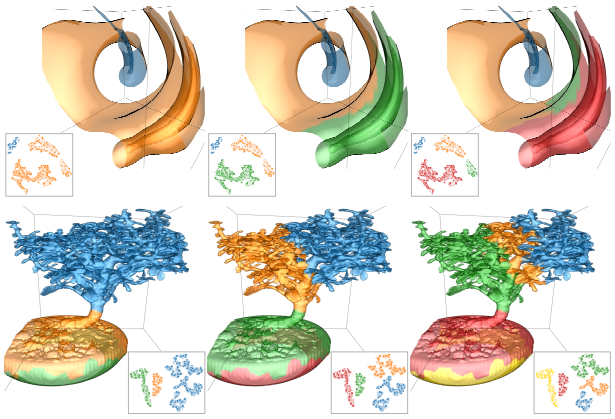
Figure 7: *Evaluation of node embeddings via brushing and linking. The unselected nodes are colored in gray.*

Node clustering comparison. In Figure 8, we qualitatively compare the node clustering results generated by GMMConv, EdgeConv, and SurfNet. The same number of clusters is used for the same data set. For the bonsai data set, GMMConv does not separate the bonsai from the basin, while both EdgeConv and SurfNet can correctly separate the two structures. The same conclusion can be drawn for the tornado data set, as only SurfNet partitions the tornado into three spirals of different curvature ranges. For the two swirls data set, GMMConv groups two spirals (refer to the green



(a) GMMConv (b) EdgeConv (c) SurfNet

Figure 8: *Node clustering results of a single surface. Top to bottom: bonsai (isosurface), tornado (stream surface), and two swirls (stream surface). Smaller surface clusters are highlighted with arrows of the same colors.*

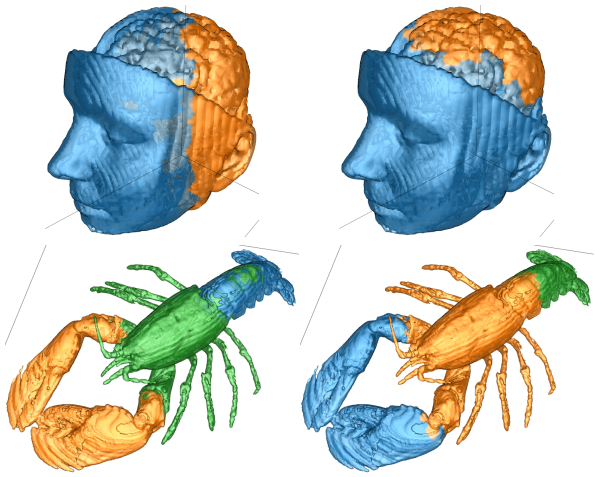


(a) (b) (c)

Figure 9: *SurfNet node clustering results of a stream surface (top) using the five critical points data set and an isosurface (bottom) using the bonsai data set. From (a) to (c), the numbers of clusters for stream surfaces are* 2, 3, *and* 4, *and the numbers of clusters for isosurfaces are* 3, 4, *and* 5.

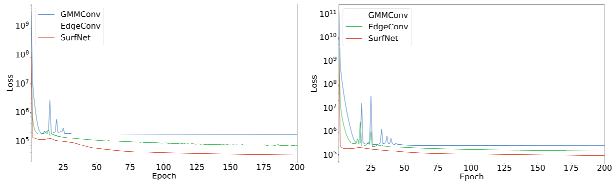
and orange parts) with overlap. EdgeConv does not detect the two spirals. SurfNet can recognize the surface's structure correctly, i.e., two spirals (refer to the blue and green parts) and one bridge (refer to the orange part). Overall, SurfNet outperforms GMMConv and EdgeConv by producing satisfactory node clustering results.

In Figure 9, we adjust the number of clusters to show the results with different scales (i.e., from coarse to fine) on a single stream surface and isosurface. All the results indicate that SurfNet can produce meaningful node clustering results under different numbers of



(a) Liu and Zhang [LZ04] (b) SurfNet

Figure 10: *Node clustering results of a single isosurface using the brain (top) and lobster (bottom) data sets.*

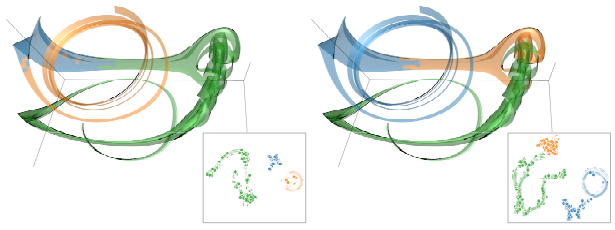


(a) five critical points (b) square cylinder

Figure 11: *Loss convergence among GMMConv, EdgeConv, and SurfNet.*

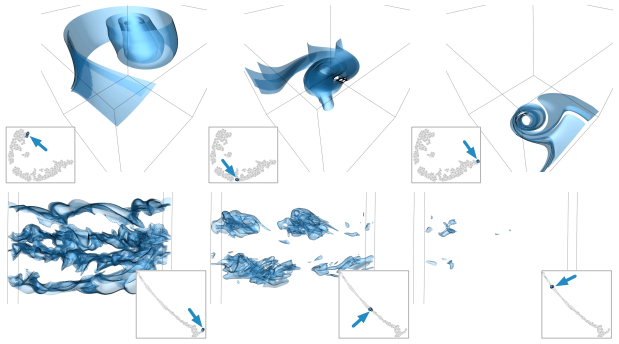
clusters. For example, the clustering shows the basin's top and bottom portions and various parts of the bonsai.

Comparison against mesh segmentation. We compare SurfNet against a spectral-based mesh segmentation approach of Liu and Zhang [LZ04]. Their method computes geodesic and angular distances between faces to build a distance matrix and applies spectral methods to select eigenvectors for segmentation. Since it is designed for closed meshes, we choose the brain and lobster data sets for comparison. The results are displayed in Figure 10. As we can



(a) joint training (b) separate training

Figure 12: *SurfNet node clustering results of a stream surface using joint training and separate training. We utilize the tornado and two swirls data set to jointly train SurfNet.*



(a) (b) (c)

Figure 13: *Evaluation of surface embeddings using the tornado (top) and combustion (YOH) (bottom) data sets via brushing and linking. Surfaces corresponding to unselected points are not displayed.*

see, Liu and Zhang [LZ04] cannot separate the face and forebrain of the brain data set and the body and tail of the lobster data set.

Baseline analysis. As shown in Figure 8, we observe that GMMConv does not produce acceptable node clustering results for all data sets, and EdgeConv only generates acceptable results for simple surfaces (e.g., bonsai). This is because GMMConv only applies linear operations on the mesh Laplacian, which leads to slow convergence and poor performance. To verify this, in Figure 11, we plot the loss curves among GMMConv, EdgeConv, and SurfNet using the five critical points and square cylinder data sets. We can see that SurfNet and EdgeConv converge faster and exhibit more stable training compared with GMMConv. EdgeConv has limited capability to detect complex node patterns. It does not produce satisfactory clustering results for the lobster and two swirls data sets.

Timing and model size analysis. In terms of training time, inference time, and model size, SurfNet, GMMConv, and EdgeConv do not exhibit significant differences, as shown in Table 4. The training time relies on the number of training samples and the complexity of surfaces (e.g., the number of nodes and edges on surfaces). As for the training time for isosurfaces, SurfNet only takes less than 10 seconds per epoch.

Performance degradation on boundary. As node clustering results show in Figures 8 and 10, the performance of node clustering degrades on the boundary of two structures (e.g., the face and forebrain of the brain data set). We give three explanations for such inaccurate boundary results. (1) *ambiguity*: The transition from one structure to another is a gradual change. This means that the shortest-path distance between two nodes located in two different patterns could be small, letting SurfNet treat the two nodes as similar. (2) *difficulty*: Detecting the boundary accurately in an unsupervised fashion is challenging since current deep learning frameworks require node annotations (e.g., experts label each node on the mesh) to delineate the boundary [HHF*19, WSL*19]. Even with these annotations, the boundary could still be inaccurate due to the uncertainty and error introduced by the annotations [ZPP*20]. (3) *amplification*: The similarity of nodes located in two structures could be close. After several graph convolutions, the similarity could be amplified due to the weight-sharing mechanism and node propagation and aggregation in GCN, which hinders SurfNet from detecting a clear boundary between two structures.

Cross data set evaluation. To evaluate the cross data set generalization of SurfNet, we perform joint training using the tornado and two swirls data sets. Both joint training and separate training take the same number of epochs for training. The surface clustering results are shown in Figure 12. For the two swirls data set, both ways of training can isolate the two swirls on the corresponding surfaces; however, separate training can also isolate the "bridge" connecting the two swirls. Therefore, we prefer using separate training.

4.3. Surface Selection

Baselines. For representative selection, we compare SurfNet against two surface selection methods:

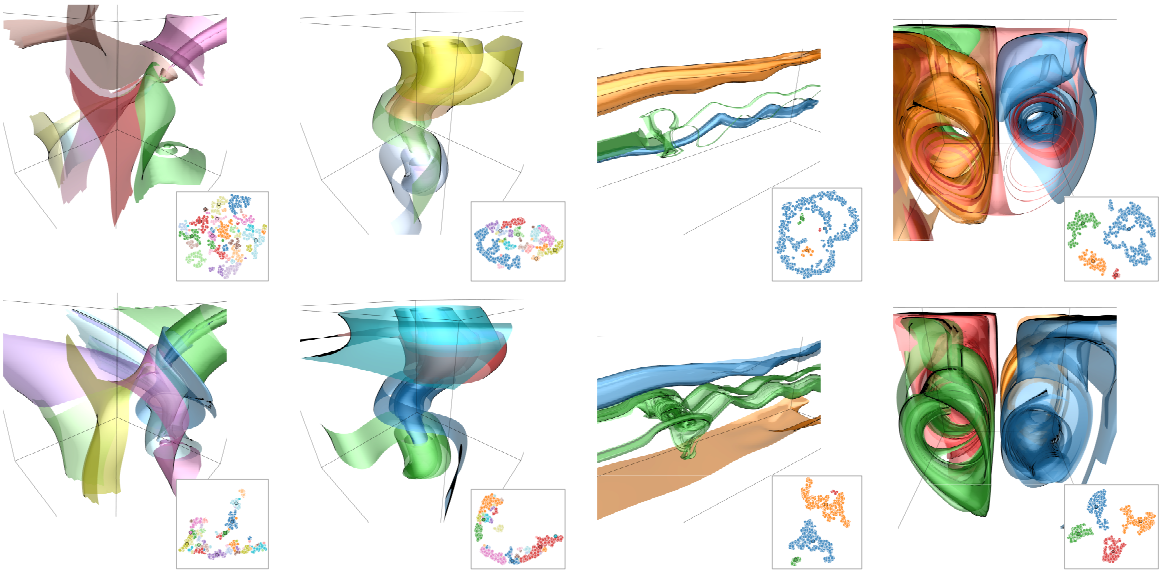
- FlowNet [HTW20] is a deep learning solution for representative stream surface selection. It encodes each surface into a latent embedding and leverages t-SNE for dimensionality reduction and DBSCAN for surface clustering.
- Isosurface similarity maps (ISM) [BM10] is constructed by computing the mutual information between isosurfaces. The representative isosurfaces are selected based on ISM.

Please refer to the accompanying videos for the frame-to-frame comparison of node clustering and representative selection results.

Surface embedding validation. To verify our surface embedding approach's effectiveness, we leverage t-SNE to project surface embeddings to a 2D space and then perform brushing and linking of these surfaces. The results are shown in Figure 13. The t-SNE projection conveys the surface's positional and shape information. For the tornado data set, the points on the left/right side of the projection view correspond to the surfaces shown at the top/bottom part of the surface view. For the combustion (YOH) data set, the points from right to left correspond to the isosurfaces with increasing isovalues. Thus, these brushing and linking results demonstrate the meaningfulness of these surface embeddings.

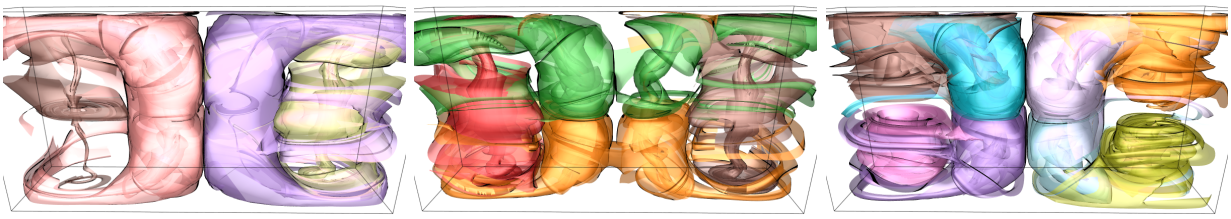
Representative selection. In Figure 14, we compare representative stream surface selection results between SurfNet and FlowNet [HTW20]. Both methods allow users to handpick representative stream surfaces via a two-step process. Both methods pick the same number of representatives from the same set of stream surfaces for fair comparisons. We can see that compared to FlowNet, SurfNet chooses a subset of surfaces that better covers the domain to reveal more interesting flow features and patterns.

In Table 4, we report the average training time per epoch, average inference time, and the model size of FlowNet and SurfNet. SurfNet only needs 17 to 50 seconds to train 1,000 samples per epoch, while FlowNet requires 200 to 1,000 seconds. As for the inference time, SurfNet is still faster than FlowNet. Besides, SurfNet only needs 0.93 MB to store model parameters, while FlowNet needs 710 MB on average. This is because SurfNet is independent of the surface resolution, while FlowNet depends on the resolution of the vector field data. Another advantage of SurfNet is that it requires fewer training samples (i.e., 1,000) for optimization since graph convolution operations are permutation invariant, and SurfNet utilizes both Euclidean and geodesic distances and a node's neighborhood information when producing embeddings. In



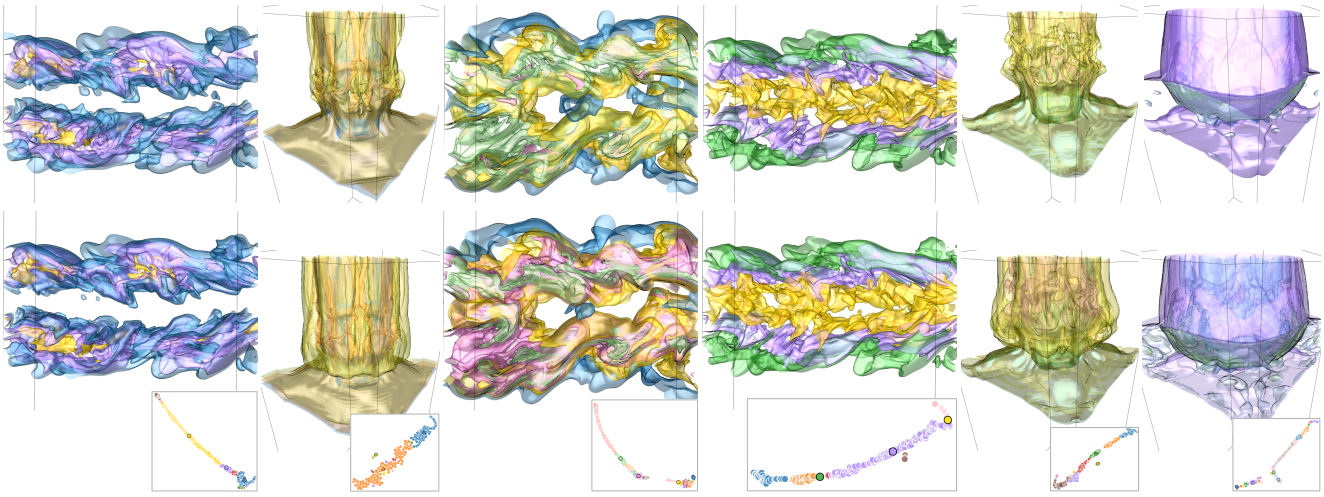
(a) five critical points (b) tornado (c) square cylinder (d) two swirls

Figure 14: *Representative stream surface selection results. Top row: FlowNet. Bottom row: SurfNet.*



(a) (b) (c)

Figure 15: *Customized SurfNet representative stream surface selection results using the Bénard flow data set. The numbers of representative surfaces are* 4, 4, *and* 8, *respectively, from (a) to (c).*



(a) YOH (b) H (c) YOH → HR (d) YOH → MF (e) H → He (f) H → PD

Figure 16: *Representative isosurface selection results. Top row: ISM. Bottom row: SurfNet. (a), (c), and (d): combustion. (b), (e), and (f): ionization. For SurfNet, (a) and (b) show same-variable inference results, while (c) to (f) show difference-variable inference results. The numbers of representative surfaces are* 3, 3, 4, 3, 3, *and* 4, *respectively, from (a) to (f).*

Table 4: *Average training time per epoch (in second) with* $1,000$ *surface samples for training, average inference time (in second), and model size (MB). Note that FlowNet is designed for learning surface embedding, not node embedding.*

data set	method	training	inference	model size
five critical points	FlowNet	333	1.87	541
	GMMConv	19.18	0.005	0.83
	EdgeConv	14.84	0.006	0.84
	SurfNet	17.40	0.009	0.93
solar plume	FlowNet	$1,001$	2.42	$1,228$
	GMMConv	61.14	0.028	0.83
	EdgeConv	54.56	0.030	0.84
	SurfNet	50.85	0.034	0.93
square cylinder	FlowNet	255	1.83	791
	GMMConv	18.79	0.005	0.83
	EdgeConv	17.42	0.006	0.84
	SurfNet	18.36	0.009	0.93
two swirls	FlowNet	210	1.56	281
	GMMConv	21.69	0.005	0.83
	EdgeConv	19.84	0.006	0.84
	SurfNet	20.69	0.009	0.93

contrast, FlowNet requires $2,000$ samples to train for most of the vector field data sets because convolutional operations are not permutation invariant, and FlowNet only considers neighborhood information when learning the surface embedding.

In Figure 15, we can see that SurfNet can flexibly represent the underlying vector field with selected stream surfaces at different levels of detail. Users can determine the suitable number of representatives empirically as they adjust interactively.

In Figure 16 (a) and (b), we compare our representative isosurface selection results against those selected by ISM [BM10]. It takes ISM three minutes to generate the representative set. For fair comparisons, both methods select the same number of representatives from the same pool of isosurfaces. We can see that both methods can capture important isosurfaces. Furthermore, in Figure 16 (c) to (f), we compare the representative isosurface selection results through different-variable inference. In this case, we use variable YOH of the combustion data set for training and apply the network to HR and MF for inference. For the ionization data set, we use variable H for training and He and PD for inference. We can observe that these representative isosurfaces also exhibit comparable results as ISM. In Table 5, we report PSNR values for different representative surface selection methods. Although the difference is marginal, SurfNet achieves higher PSNRs for surface selection.

Table 5: *PSNRs of reconstructed vector and scalar fields (the higher PSNRs are highlighted in bold).*

data set	method	# REPs	PSNR (dB)
solar plume	FlowNet	7	30.43
	SurfNet	7	**31.43**
tornado	FlowNet	4	22.49
	SurfNet	4	**24.41**
two swirls	FlowNet	4	27.55
	SurfNet	4	**28.12**
combustion (HR)	ISM	4	18.86
	SurfNet	4	**18.91**
ionization (PD)	ISM	4	18.86
	SurfNet	4	**19.02**

4.4. Discussion

Although SurfNet produces coarser results in 3D shape compared with the existing solutions [Sha08], SurfNet still has two advantages. (1) *Generalizability*: Once converged, SurfNet can group nodes on *unseen* surfaces, while traditional shape analysis approaches for meshes need to be rerun for producing new results. (2) *Diversity*: SurfNet can handle different kinds of surfaces while shape analysis algorithms are typically used to process *closed* meshes (e.g., 3D objects) but not *open* ones (e.g., stream surfaces). Also, SurfNet can handle node *clustering* and surface *selection*, while shape analysis only focuses on node-level rather than surface-level analysis. In addition, we believe it is a good start in unsupervised geometric learning for surface analysis and has great potential to improve in the future by incorporating domain knowledge into graph convolutional operations.

5. Conclusions and Future Work

We have presented SurfNet, a new solution for clustering and selecting stream surfaces and isosurfaces. Based on the GCN, SurfNet can learn permutation-invariant node embedding from surfaces in an unsupervised manner, drawing a big difference from other works that learn node embeddings from labels and use different frameworks to solve surface clustering and selection separately. These node embeddings encode their neighborhood information rather than purely physical information (e.g., position, normal). By projecting these embeddings into a 2D space, we provide an interactive visual interface for user exploration. Meaningful clustering and selection results are yielded under different clustering hyperparameters. These node embeddings produced from SurfNet preserve spatial proximity and geometric similarity.

We validate SurfNet on various vector and scalar field data sets of different patterns and compare the clustering and selection results derived from the learned node embeddings with those produced using other state-of-the-art methods. Compared with FlowNet, training SurfNet is 10 to 20 times faster per epoch and inferring is 70 to 170 times faster while the model's storage-saving is 300 to $1,300$ folds (Table 4). Qualitative results show that SurfNet generates better node clustering results than those generated by GMMConv and EdgeConv. Qualitative and quantitative results show that SurfNet suggests comparable or better representative selection results than those generated by FlowNet (stream surfaces) and ISM (isosurfaces).

To our best knowledge, SurfNet is the first that applies GCN to solve surface clustering and selection problems for scientific visualization. Our current work focuses on detecting surface patterns and grouping a set of surfaces. We will explore the possibility of GCN in two directions (1) predicting stream surfaces (lines) conditioned on a user-specified seeding curve (point) through GCN. If successful, this could enable us to predict the quality of the corresponding stream surfaces (lines) without actual tracing. (2) detecting delicate structures (e.g., separating the lobster's arms from its claws) with little human intervention. Currently, the clustering results can discover major surface structures but fail to discern minor ones. We will leverage semi- supervised learning to improve the node embedding quality by providing a small number of human annotations to SurfNet.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CNS-1629914, DUE-1833129, IIS-1955395, IIS-2101696, and OAC-2104158.

References

[AEHPARA18] ABU-EL-HAIJA S., PEROZZI B., AL-RFOU R., ALEMI A. A.: Watch your step: Learning node embeddings via graph attention. In *Proceedings of Advances in Neural Information Processing Systems* (2018), pp. 9198–9208. 6

[ASN*13] AHMED A., SHERVASHIDZE N., NARAYANAMURTHY S., JOSIFOVSKI V., SMOLA A. J.: Distributed large-scale natural graph factorization. In *Proceedings of ACM International Conference on World Wide Web* (2013), pp. 37–48. 4

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine 34*, 4 (2017), 18–42. 1, 2

[BDQ*19] BAI Y., DING H., QIAO Y., MARINOVIC A., GU K., CHEN T., SUN Y., WANG W.: Unsupervised inductive graph-level representation learning via graph-graph proximity. In *Proceedings of International Joint Conferences on Artificial Intelligence* (2019), pp. 1988–1994. 3

[BLL19] BERGER M., LI J., LEVINE J. A.: A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics 25*, 4 (2019), 1636–1650. 2

[BM10] BRUCKNER S., MÖLLER T.: Isosurface similarity maps. *Computer Graphics Forum 29*, 3 (2010), 773–782. 8, 10

[BTBW77] BARROW H. G., TENENBAUM J. M., BOLLES R. C., WOLF H. C.: Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of International Joint Conference on Artificial Intelligence* (1977), pp. 659–663. 4

[CCJ*19] CHENG H.-C., CARDONE A., JAIN S., KROKOS E., NARAYAN K., SUBRAMANIAM S., VARSHNEY A.: Deep-learning-assisted volume visualization. *IEEE Transactions on Visualization and Computer Graphics 25*, 2 (2019), 1378–1391. 2

[CYP*21] CORSO G., YING R., PÁNDY M., VELIČKOVIĆ P., LESKOVEC J., LIÒ P.: Neural distance embeddings for biological sequences. In *Proceedings of Advances in Neural Information Processing Systems* (2021). 4

[EKSX96] ESTER M., KRIEGEL H.-P., SANDER J., XU X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (1996), pp. 226–231. 3

[FML15] FOFONOV A., MOLCHANOV V., LINSEN L.: Visual analysis of multi-run spatio-temporal simulations using isocontour similarity for projected views. *IEEE Transactions on Visualization and Computer Graphics 22*, 8 (2015), 2037–2050. 5

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH Conference* (1997), pp. 209–216. 3

[GHCW21] GU P., HAN J., CHEN D. Z., WANG C.: Reconstructing unsteady flow data from representative streamlines via diffusion and deep learning based denoising. *IEEE Computer Graphics and Applications 41*, 6 (2021), 111–121. 2

[GHCW22] GU P., HAN J., CHEN D. Z., WANG C.: Scalar2Vec: Translating scalar fields to vector fields via deep learning. In *Proceedings of IEEE Pacific Visualization Symposium* (2022). Accepted. 2

[GYH*20] GUO L., YE S., HAN J., ZHENG H., GAO H., CHEN D. Z., WANG J.-X., WANG C.: SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In *Proceedings of IEEE Pacific Visualization Symposium* (2020), pp. 71–80. 2

[HHF*19] HANOCKA R., HERTZ A., FISH N., GIRYES R., FLEISHMAN S., COHEN-OR D.: MeshCNN: A network with an edge. *ACM Transactions on Graphics 38*, 4 (2019), 90:1–90:12. 1, 3, 8

[HLL*19] HUANG J., LI Z., LI N., LIU S., LI G.: AttPool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *Proceedings of IEEE International Conference on Computer Vision* (2019), pp. 6480–6489. 6

[HLY19] HONG F., LIU C., YUAN X.: DNN-VolVis: Interactive volume visualization supported by deep neural network. In *Proceedings of IEEE Pacific Visualization Symposium* (2019), pp. 282–291. 2

[HTW20] HAN J., TAO J., WANG C.: FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics 26*, 4 (2020), 1732–1744. 1, 2, 3, 5, 6, 8

[HTZ*19] HAN J., TAO J., ZHENG H., GUO H., CHEN D. Z., WANG C.: Flow field reduction via reconstructing vector data from 3D streamlines using deep learning. *IEEE Computer Graphics and Applications 39*, 4 (2019), 54–67. 2

[HW20a] HAN J., WANG C.: SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). Accepted. 2

[HW20b] HAN J., WANG C.: TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 205–215. 2

[HW22] HAN J., WANG C.: TSR-VFD: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics 103* (2022), 168–179. 2

[HWG*20] HE W., WANG J., GUO H., WANG K.-C., SHEN H.-W., RAJ M., NASHED Y. S. G., PETERKA T.: InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 23–33. 2

[HYL17] HAMILTON W., YING Z., LESKOVEC J.: Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems* (2017), pp. 1024–1034. 4

[HZ*22] HAN J., ZHENG H., , CHEN D. Z., WANG C.: STNet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. *IEEE Transactions on Visualization and Computer Graphics 28*, 1 (2022), 270–280. 2

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of IEEE International Conference on Computer Vision* (2015), pp. 1026–1034. 5

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778. 4

[HZX*21] HAN J., ZHENG H., XING Y., CHEN D. Z., WANG C.: V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1290–1300. 2

[HZY18] HONG F., ZHANG J., YUAN X.: Access pattern learning with long short-term memory for parallel particle tracing. In *Proceedings of IEEE Pacific Visualization Symposium* (2018), pp. 76–85. 2

[JGG21] JAKOB J., GROSS M., GÜNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1279–1289. 2

[JZL*19] JIANG B., ZHANG Z., LIN D., TANG J., LUO B.: Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 11313–11320. 6

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 5

[KJP*18] KOSTRIKOV I., JIANG Z., PANOZZO D., ZORIN D., BRUNA J.: Surface networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2540–2548. 3

[KW17] KIPF T. N., WELLING M.: Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference for Learning Representations* (2017). 6

[LBBM18] LITANY O., BRONSTEIN A., BRONSTEIN M., MAKADIA A.: Deformable shape completion with graph convolutional autoencoders. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 1886–1895. 2

[Lin91] LIN J.: Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory 37*, 1 (1991), 145–151. 5

[LJQ19] LIU C., JI H., QIU A.: Convolutional neural network on semi-regular triangulated meshes and its application to brain image data. *arXiv preprint arXiv:1903.08828* (2019). 2, 3

[LZ04] LIU R., ZHANG H.: Segmentation of 3D meshes through spectral clustering. In *Proceedings of Pacific Conference on Computer Graphics and Applications* (2004), pp. 298–305. 7, 8

[MBM*17] MONTI F., BOSCAINI D., MASCI J., RODOLA E., SVOBODA J., BRONSTEIN M. M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5115–5124. 2, 3, 6

[NH10] NAIR V., HINTON G. E.: Rectified linear units improve restricted Boltzmann machines. In *Proceedings of International Conference on Machine Learning* (2010), pp. 807–814. 4

[QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L. J.: Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5648–5656. 1

[RBSB18] RANJAN A., BOLKART T., SANYAL S., BLACK M. J.: Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of European Conference on Computer Vision* (2018), pp. 704–720. 2

[SFRM19] SMITH E., FUJIMOTO S., ROMERO A., MEGER D.: GEOMetrics: Exploiting geometric structure for graph-encoded objects. In *Proceedings of International Conference on Machine Learning* (2019), pp. 5866–5876. 2

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556. 10

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of IEEE International Conference on Computer Vision* (2015), pp. 945–953. 1

[SQX*16] SHU Z., QI C., XIN S., HU C., WANG L., ZHANG Y., LIU L.: Unsupervised 3D shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design 43* (2016), 39–52. 2

[SXW*22] SHI N., XU J., WURSTER S. W., GUO H., WOODRING J., VAN ROEKEL L. P., SHEN H.-W.: Gnn-Surrogate: A hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations. *IEEE Transactions on Visualization and Computer Graphics* (2022). Accepted. 2

[TFE21a] TKACHEV G., FREY S., ERTL T.: Local prediction models for spatiotemporal volume visualization. *IEEE Transactions on Visualization and Computer Graphics 27*, 7 (2021), 3091–3108. 2

[TFE21b] TKACHEV G., FREY S., ERTL T.: S4: Self-supervised learning of spatiotemporal similarity. *IEEE Transactions on Visualization and Computer Graphics* (2021). Accepted. 2, 4

[TW18] TAO J., WANG C.: Semi-automatic generation of stream surfaces via sketching. *IEEE Transactions on Visualization and Computer Graphics 24*, 9 (2018), 2622–2635. 2

[vdMH08] VAN DER MAATEN L. J. P., HINTON G. E.: Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 3

[WCTW21] WEISS S., CHU M., THUEREY N., WESTERMANN R.: Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics 27*, 6 (2021), 3064–3078. 2

[WITW20] WEISS S., IŞIK M., THIES J., WESTERMANN R.: Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). Accepted. 2

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1912–1920. 1

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics 38*, 5 (2019), 146:1–146:12. 3, 6, 8

[WZL*18] WANG N., ZHANG Y., LI Z., FU Y., LIU W., JIANG Y.-G.: Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proceedings of European Conference on Computer Vision* (2018), pp. 52–67. 3

[WZY*19] WANG M., ZHENG D., YE Z., GAN Q., LI M., SONG X., ZHOU J., MA C., YU L., GAI Y., XIAO T., HE T., KARYPIS G., LI J., ZHANG Z.: Deep Graph Library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019). 5

[XP97] XU C., PRINCE J. L.: Gradient vector flow: A new external force for snakes. In *Proceedings of IEEE International Conference on Computer Vision* (1997), pp. 66–71. 6

[XZJK21] XU K., ZHANG M., JEGELKA S., KAWAGUCHI K.: Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *Proceedings of International Conference on Machine Learning* (2021), pp. 11592–11602. 4

[YHC*18] YING R., HE R., CHEN K., EKSOMBATCHAI P., HAMILTON W. L., LESKOVEC J.: Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), pp. 974–983. 3, 4

[YSGG17] YI L., SU H., GUO X., GUIBAS L. J.: SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2282–2290. 2, 3

[YYM*18] YING Z., YOU J., MORRIS C., REN X., HAMILTON W., LESKOVEC J.: Hierarchical graph representation learning with differentiable pooling. In *Proceedings of Advances in Neural Information Processing Systems* (2018), pp. 4800–4810. 1

[YYSZ20] YAO G., YUAN Y., SHAO T., ZHOU K.: Mesh guided one-shot face reenactment using graph convolutional networks. In *Proceedings of ACM International Conference on Multimedia* (2020), pp. 1773–1781. 3

[ZPP*20] ZHENG H., PERRINE S. M. M., PITIRRI M. K., KAWASAKI K., WANG C., RICHTSMEIER J. T., CHEN D. Z.: Cartilage segmentation in high-resolution 3D micro-CT images via uncertainty-guided self-training with very sparse annotation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention* (2020), pp. 802–812. 8

Appendix

1. Graph Convolution

Algorithm 1 shows the graph convolution operation.

Algorithm 1: Graph convolution.

Input: node embedding $\mathbf{F}_{u,j}$ at layer j, a set of neighbor embeddings $\{\mathbf{F}_v | v \in \mathcal{N}(u)\}$, a set of neighbor weights $\mathbf{W}$ and bias b, an activation function $\gamma(\cdot)$, an aggregator $\alpha(\cdot)$, and a norm function $N(\cdot)$.

Output: node embedding $\mathbf{F}_{u,j+1}$ at layer $j+1$.

$\mathbf{F}_{u,j+1} \leftarrow \gamma(\{\mathbf{W}\mathbf{F}_{v,j} + b) | v \in \{\mathcal{N}(u), u\}\})$

$\mathbf{F}_{u,j+1} \leftarrow \alpha(\mathbf{F}_{u,j+1})$

$\mathbf{F}_{u,j+1} \leftarrow \mathbf{F}_{u,j+1} / N(\mathbf{F}_{u,j+1})$

2. Additional Results

In Figure 1, we show more results with different data sets using SurfNet for stream surface clustering. Each column shows node clustering results for different surfaces of the same data set. These results further confirm the reliability of SurfNet in clustering the nodes of stream surfaces. We found that for complex flows (e.g., solar plume), the clustering results generated by SurfNet include some minor errors, but the main surface structures can be detected correctly.

3. Dimensionality Reduction and Clustering

Dimensionality reduction. To transform the learned features into a 2D space, we experiment with four dimensionality reduction methods: t-SNE [vdMH08], UAMP [MHM18], MDS [Kru64], and Isomap [TdSL00]. Among them, t-SNE is a neighborhood-preserving method, UMAP is a global-preserving method, and MDS and Isomap are distance-preserving methods. The brushing and linking results are shown in Figure 2. Both MDS and Isomap cannot group similar nodes in the 2D projection space, while t-SNE and UMAP meet our expectations. In addition, in the t-SNE projection, the nodes are more separated than those in the UMAP projection. Therefore, we choose t-SNE as the dimensionality reduction algorithm to project the learned features.

Clustering. To group the points in the 2D projection space, we study three representative clustering algorithms: DBSCAN (density-based), k-means (partition-based), and agglomerative clustering (hierarchy-based). The clustering results are shown in Figure 3. DBSCAN performs best by detecting the lobster's claws, body, and tail while the other two clustering algorithms mix these structures.

Feature space vs. projection space. We also conduct a study to justify clustering the node embeddings in the projection space (i.e., the space generated by t-SNE) instead of the feature space. As shown in Figure 4, the results of clustering nodes directly in the feature space do not make sense, and only several isolated parts are grouped. The possible reason for this unsatisfactory result is that in the feature space, data are sparse, which is problematic for any method requiring statistical significance. Moreover, these data are dissimilar in many ways, preventing the traditional clustering algorithm from working efficiently. However, by clustering the node embeddings in the t-SNE space, we keep all important information and decompose co-related factors, ensuring that the clustering algorithm works well.

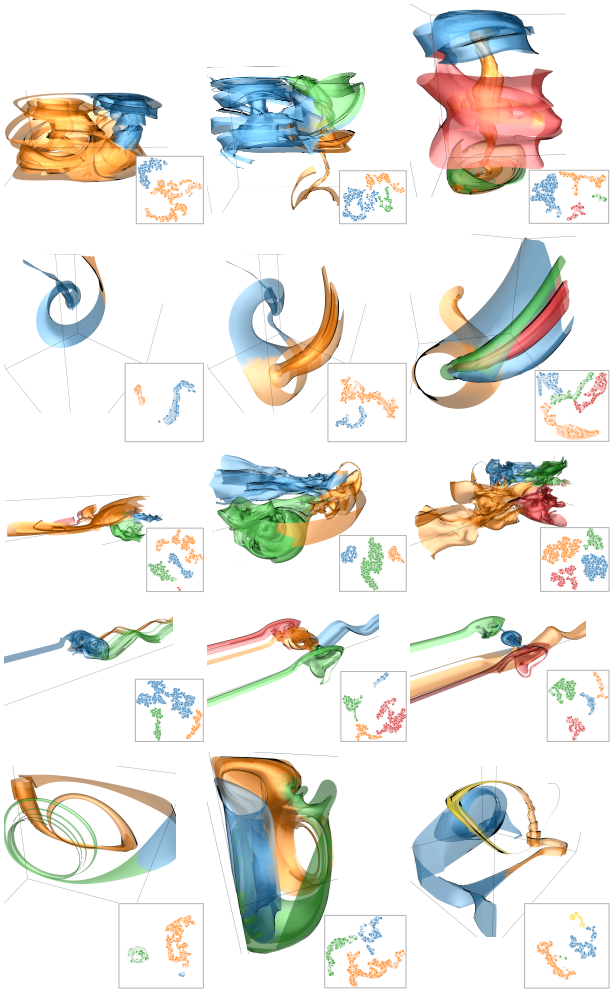


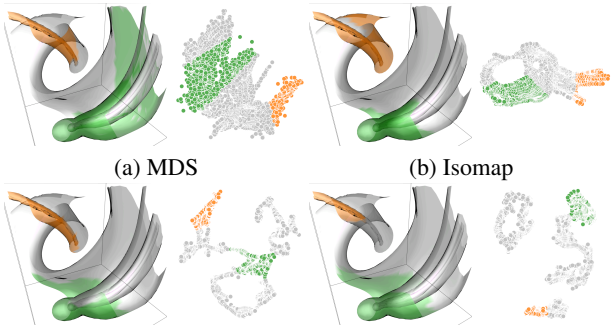
Figure 1: *SurfNet node clustering results of a stream surface. Top to bottom: Bénard flow, five critical points, solar plume, square cylinder, and two swirls.*

Therefore, in this paper, we choose the combination of t-SNE for dimensionality reduction and DBSCAN for node clustering and perform clustering of node embeddings in the t-SNE space.

4. SurfNet Analysis

To evaluate SurfNet, we analyze the following hyperparameter settings: mesh simplification, network depth, training stability, embedding strategy, embedding dimension, feature initialization, and training samples. We focus on node clustering results, from which surface embedding (representative selection) results are derived. For fair comparisons, all remaining parameters use the same settings (e.g., the number of clusters).

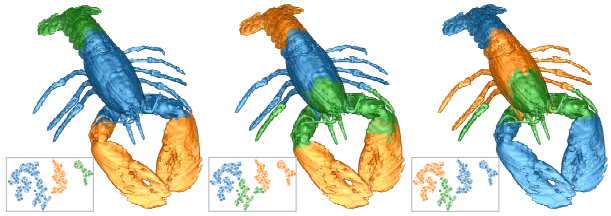
Mesh simplification. To study how mesh simplification impacts the quality of node clustering results, we apply different mesh simplification thresholds ε [GH97] (larger ε leads to more simplifica-



(a) MDS (b) Isomap

(c) UMAP (d) t-SNE

Figure 2: *Comparison of different dimensionality reduction methods via brushing and linking using the five critical points data set. The unselected nodes are colored in gray.*



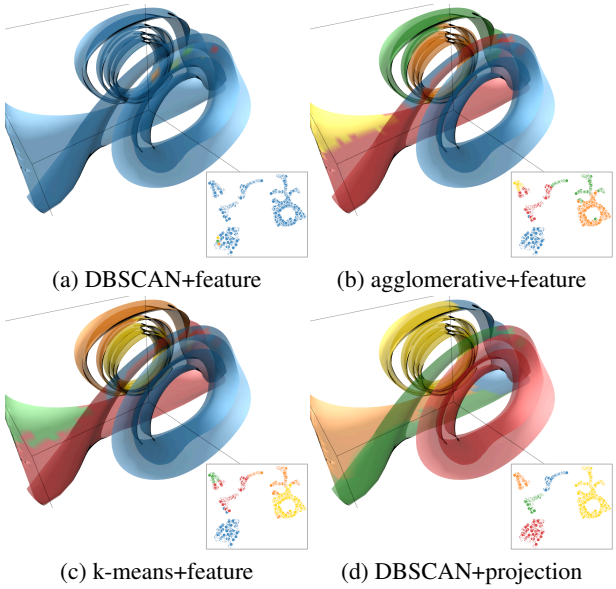
(a) DBSCAN (b) k-means (c) agglomerative

Figure 3: *Comparison of different clustering algorithms under t-SNE projection using the lobster data set. Each produces* 3 *clusters.*

tion) to simplify stream surfaces. We then use the simplified surfaces to train SurfNet using the two swirls data set. The results are shown in Figure 5. Under $\varepsilon = 1.0$, SurfNet cannot detect the bridge that connects the left and right spirals. In addition, some blue surface patches are mixed with the orange ones. Under $\varepsilon = 0.5$ and $\varepsilon = 0.2$, the clustering results have no significant differences. Both can separate the three spirals and the bridge well. Note that all the surfaces displayed in the figures are the original surface without simplification. Simplified surfaces are only used for SurfNet training. Hence, we suggest applying $\varepsilon = 0.5$ to simplify the surfaces.

Network depth. Given a mesh simplification threshold ε (i.e., 0.5), we evaluate how the network depth affects the performance of node embedding. We use 3, 6, and 9 layers to optimize SurfNet using the two swirls data sets. The node clustering results are displayed in Figure 6. Only using 3 layers cannot produce meaningful node embeddings since it groups one swirl and its bridge into one cluster. However, applying 6 and 9 layers can discover the two swirls (i.e., the blue and red parts) and two bridges (i.e., the orange and green parts). Therefore, under $\varepsilon = 0.5$, using SurfNet with 6 layers is sufficient to generate meaningful node embeddings.

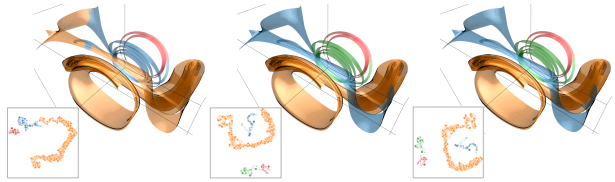
Training stability. To investigate the stability and sensitivity of the training process, we independently train SurfNet four times. The parameters are randomly initialized using He et al. [HZRS15]. The results are displayed in Figure 7. There is no significant difference among these results, and all results can discover the bonsai and its basin. Note that the t-SNE projections of the four results are not similar due to random initialization of the 2D points in the t-SNE algorithm. However, the relative positions of these points



(a) DBSCAN+feature (b) agglomerative+feature

(c) k-means+feature (d) DBSCAN+projection

Figure 4: *Comparison of different clustering algorithms under different spaces (feature vs. projection) using the two swirls data set.*

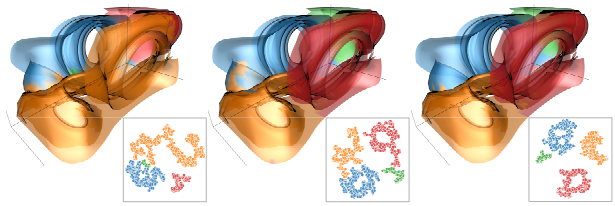


(a) $\varepsilon = 1.0$ (b) $\varepsilon = 0.5$ (c) $\varepsilon = 0.2$

Figure 5: *SurfNet node clustering results of a stream surface under different thresholds for mesh simplification using the two swirls data set.*

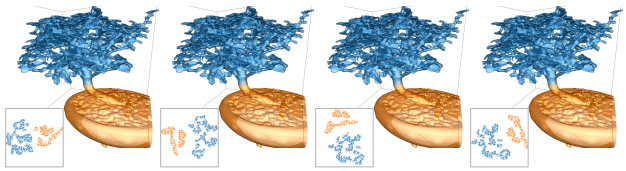
are consistent. Therefore, SurfNet is stable in terms of training and node clustering.

Embedding strategy. To confirm the effectiveness of generating node embeddings using SurfNet, we compare the clustering results generated using node embedding with SurfNet and directly using the shortest-path length as the distance measure, as shown in Figure 8. The clustering results of SurfNet outperform those of direct embedding as the number of clusters increases. One possible explanation is that the features generated by SurfNet include Euclidean



(a) 3 layers (b) 6 layers (c) 9 layers

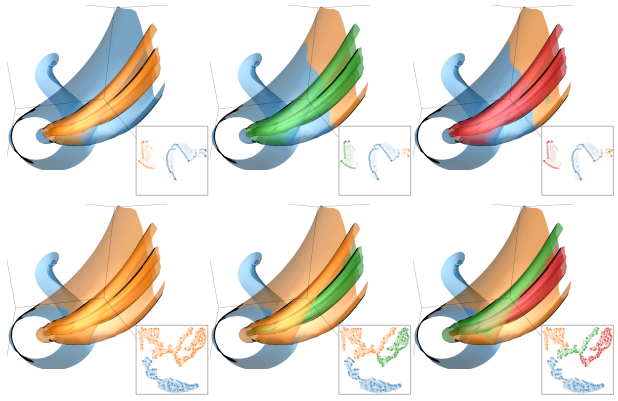
Figure 6: *SurfNet node clustering results of a stream surface under different network depths using the two swirls data set.*



(a) (b) (c) (d)

Figure 7: *SurfNet node clustering results of an isosurface using the bonsai data set. (a) to (d) show the node clustering results from SurfNet trained four times independently.*

information (i.e., node position) and geodesic information (i.e., loss optimization). The former can offer cues between nearby nodes, while the latter can help separate spatially close nodes with large geodesic distances [HBS*21]. Therefore, the embedding generated by SurfNet better captures the surface structure and leads to better clustering results.

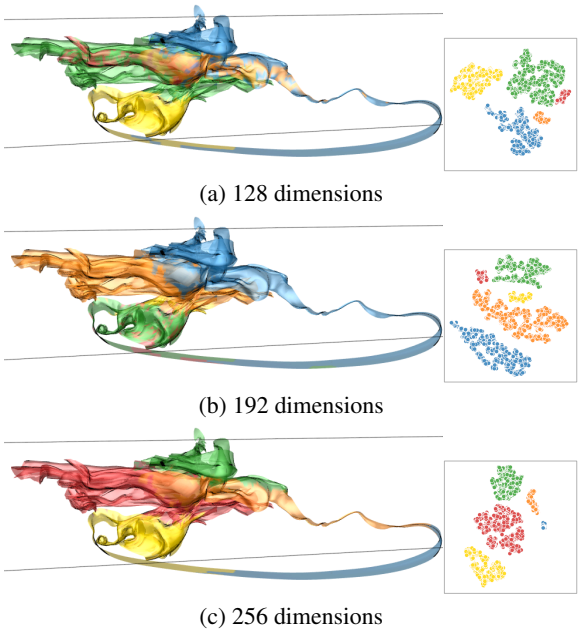


(a) 2 clusters (b) 3 clusters (c) 4 clusters

Figure 8: *Direct embedding and SurfNet embedding of a stream surface under t-SNE projection using the five critical points data set. Top: direct embedding, and bottom: SurfNet embedding.*

Embedding dimension. To determine the appropriate node embedding dimension, we set it to 128, 192, and 256, respectively, to train SurfNet. The node clustering results are shown in Figure 9. The clustering results are not satisfactory under 128 and 192 embedding dimensions. For example, under 128 dimensions, green, red, and orange parts are mixed, and under 192 dimensions, green and red parts cannot be separated well. However, using 256 embedding dimensions, all the structures are clearly separated. In addition, we also use 384 and 512 dimensions to embed node information; however, there is no significant difference compared to the result of 256. Therefore, we set the node embedding dimension to 256 for SurfNet.

Node information initialization. To investigate how to initialize the input surfaces' features, we train SurfNet with different feature initialization options (e.g., position, normal, velocity) using the square cylinder data set. As shown in Figure 10, we display the surface clustering results under different feature initializations. Leveraging position as input features, SurfNet achieves the best clustering performance. For normal, SurfNet cannot separate meaningful



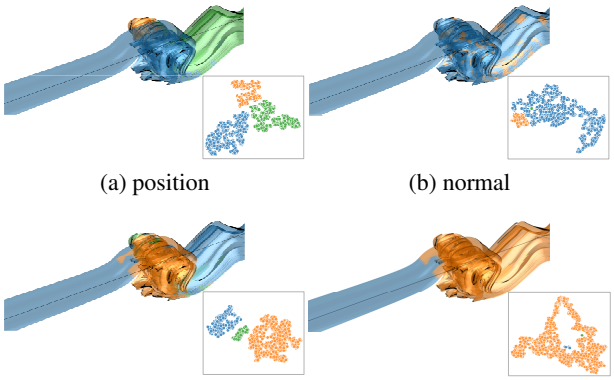
(a) 128 dimensions

(b) 192 dimensions

(c) 256 dimensions

Figure 9: *SurfNet node clustering results of a stream surface under different numbers of embedding dimensions using the solar plume data set.*

parts. For velocity, SurfNet detects major structures but still misclassifies some of the orange parts into the green part, as shown in Figure 10 (c). In addition, if we use both position and velocity as input features, the clustering quality does not improve. In summary, using normal, velocity, or position+velocity as the initialized features does not lead to good node clustering results. This is because SurfNet is optimized using the shortest-path loss. Only position-related information can provide an appropriate initialization for SurfNet optimization. Hence, we suggest only using position as the initialized feature for SurfNet training.



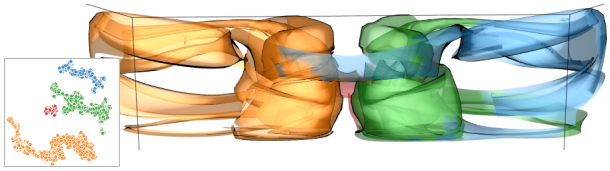
(a) position (b) normal

(c) velocity (d) position+velocity

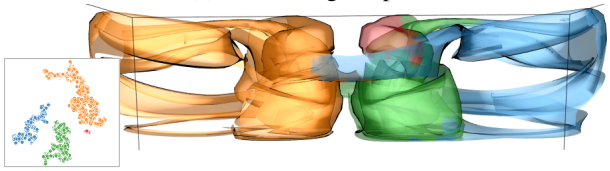
Figure 10: *SurfNet node clustering results of a stream surface under different feature initializations using the square cylinder data set.*

Training samples. We evaluate the influence of the number of

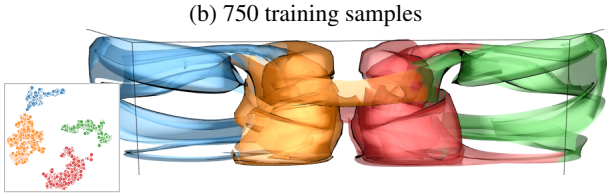
training samples on clustering quality using the Bénard flow data set. We use 500, 750, and 1,000 training samples to train SurfNet. The clustering results are shown in Figure 11. It is clear that using 500 or 750 training samples, SurfNet cannot detect the four major patterns of the stream surface. However, with 1,000 training samples, these patterns can be discovered. Besides, our experiment shows that using more than 1,000 training samples does not further improve clustering quality. Hence, we suggest using 1,000 stream surfaces for training.



(a) 500 training samples



(b) 750 training samples



(c) 1,000 training samples

Figure 11: *SurfNet node clustering results of a stream surface under different numbers of training samples using the Bénard flow data set.*

References

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH Conference* (1997), pp. 209–216. 1

[HBS*21] HU Z., BAI X., SHANG J., ZHANG R., DONG J., WANG X., SUN G., FU H., TAI C.-L.: VMNet: Voxel-mesh network for geodesic-aware 3D semantic segmentation. In *Proceedings of International Conference on Computer Vision* (2021). 3

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of IEEE International Conference on Computer Vision* (2015), pp. 1026–1034. 2

[Kru64] KRUSKAL J. B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27. 1

[MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 1

[TdSL00] TENENBAUM J. B., DE SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323. 1

[vdMH08] VAN DER MAATEN L. J. P., HINTON G. E.: Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 1