This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

Local Latent Representation based on Geometric Convolution for Particle Data Feature Exploration

Haoyu Li and Han-Wei Shen

Abstract—Feature related particle data analysis plays an important role in many scientific applications such as fluid simulations, cosmology simulations and molecular dynamics. Compared to conventional methods that use hand-crafted feature descriptors, some recent studies focus on transforming the data into a new latent space, where features are easier to be identified, compared and extracted. However, it is challenging to transform particle data into latent representations, since the convolution neural networks used in prior studies require the data presented in regular grids. In this paper, we adopt Geometric Convolution, a neural network building block designed for 3D point clouds, to create latent representations for scientific particle data. These latent representations capture both the particle positions and their physical attributes in the local neighborhood so that features can be extracted by clustering in the latent space, and tracked by applying tracking algorithms such as mean-shift. We validate the extracted features and tracking results from our approach using datasets from three applications and show that they are comparable to the methods that define hand-crafted features for each specific dataset.

Index Terms—Data transformation, Particle data, Feature extraction and tracking, Deep learning

1 Introduction

With the growing size and increasing complexity of simulation outputs, feature extraction and tracking have become essential tasks for scientific data analysis. Through extraction and tracking of salient features, scientists can better understand the evolution of scientific phenomena. In the domain of data visualization, visualizing extracted features avoids 3D occlusion and improves visualization efficiency, especially when the dataset is large. From the early work [1], [2] to the more recent approaches such as topology analysis [3], [4] and deep learning [5], feature extraction and tracking methods generally share a common strategy, which is to find the spatially unique and temporally coherent regions based on some feature specific attributes such as variable values, size, shape, topology or texture.

There have been a plethora of studies that focus on designing and engineering feature descriptors in different datasets from various domains [3], [4], [6], [7], [8], [9], [10]. However, finding a precise definition for salient features can be difficult due to the complexity of many scientific phenomena. Extracting features with hand-crafted feature descriptors can involve tedious manual intervention and is not always robust. In order to solve this problem, automatic feature extraction approaches based on machine learning, or more specifically neural networks, have been explored. A new representation, called latent vectors, is produced from the raw data using a neural network. After that, feature exploration is performed based on analyses of clusters and distances between the clusters in the latent space. Automatic feature extraction studies showed promising results with little human intervention [11], [12].

- *H. Li is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210. E-mail: li.8460@osu.edu*
- *H. Shen is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210. E-mail: hwshen@cse.ohio-state.edu*

When it comes to particle data, such as those produced by molecular dynamics, cosmology, or fluid flow simulations, applying neural network based automatic feature exploration is not a trivial problem. Most of the previous works use convolution kernels to automatically learn the feature descriptors. This requires a regular grid mesh around each pixel/voxel, which is not present in particle datasets. Although there are workarounds that first re-sample the data into a regular grid and then apply convolution [12], re-sampling data may cause loss of details or add a significant storage cost. Most importantly, particle geometry defined by the particle positions is lost in this process. Many features, especially those in cosmology simulations, are related to particle geometries, e.g. the density of the particles or the shapes that particles form.

To tackle the challenges, we propose the use of Geometric Convolution (GeoConv) for producing latent representations that integrate the positional information and physical attributes for particle data. In the booming research field of neural networks for point clouds, where the main goal is to classify shapes formed by 3D point clouds, GeoConv is the key component used in one of the state-of-the-art network architectures [13]. Similar to the classical Convolutional Neural Networks (CNN), GeoConv builds convolution kernels to detect features using the local information. However, since the neighborhood of particle data or point clouds is not positioned regularly, GeoConv kernel weights depend on the relative position of the point or particle to the kernel center.

Similar to previous latent space analysis works for visualization, we use an autoencoder with GeoConv to create a latent representation for particle data, which captures the features from the raw input data without any supervision. One particle and its local neighborhood, referred to as a particle patch, is processed by the autoencoder to create a latent vector which is a succinct presentation for the particle patch. This succinct latent representation can then be used for feature exploration.

We show that clusters in the space of our latent representation for the particle data capture dataset-specific features, and that these

clusters separate into distinct features. We create a visual analytics system to allow users to cluster the latent vectors hierarchically and to identify the features interactively. Additionally, we show that this latent representation can be utilized for feature tracking by applying the mean-shift algorithm to match latent space distributions across time steps. We summarize the contribution of this study as follows:

- A neural network based approach specifically designed for particle data, which transforms particles into a local latent representation where the positional and physical attribute features from the particles are preserved.
- A demonstration that this latent representation is useful for feature extraction and tracking.
- A visual analytics system that helps users interactively explore the produced latent representations and detect the features of interest.

2 Related Works

Our work contributes to feature extraction and tracking using latent space methods on particle datasets. We, therefore, review related works to these fields.

Latent Space Approaches in Visualization: The approach proposed in this paper is inspired by the study from Cheng et al. [11], where supervised deep learning neural networks are used to extract meaningful features to help the design of a good transfer function for volume rendering. Besides this work, many recent studies in visualization utilize latent space exploration and analysis in their application. For volumetric time-varying data, Porter et al. [14] produce the latent representation for volumes from different time steps, and select representative time steps using latent vectors. For streamlines and stream surfaces, Han et al. [12] train a CNN based autoencoder with voxelized and down-sampled data. Their approach avoids the problem of defining feature descriptor for lines or surfaces but inevitably introduces error in the process of voxelization and down sampling. Meanwhile, the computation and storage overheads of their method are also large. Another related recent study by Han et al. [15] uses the proximity in the latent space to select the best pairs of variables to perform volumetric data transformation. These latent space related works all use a CNN to produce latent vectors. This requires the dataset to be volumetric or to be re-sampled to a volume.

Meanwhile, recent works in the deep learning domain have designed and applied neural networks [16], [17], [18], [19] to point cloud data and have seen great success for classification and segmentation, where both require meaningful feature extraction from point clouds. Applying neural networks designed specifically for point cloud data will not be prone to the error incurred from the re-sampling process as the mentioned approaches do.

Feature Extraction and Tracking: Feature extraction and tracking are two applications for our particle latent representation. We summarize the previous works on feature extraction and tracking in this section. First, there are a group of studies focused on attribute space analysis, for example, feature level-sets [20] select traits defined by arbitrary geometries in the attribute space and perform iso-surface extraction for multi-variate data, and Linsen et al. [21] used a region growing method in the attribute space to solve the problem of extracting features defined over multiple variables in a particle dataset. These works are similar to ours in the sense that we all extract features in the spatial domain based on the selection in another space, which is either a latent space produced by neural networks or the original attribute space. Attribute space methods are possible to be directly applied to particle datasets. However, compared to the attribute space for particle datasets, a latent space is able to capture more information such as the particles' spatial distribution and the relation between positions and attributes.

Most of the other feature extraction and tracking techniques that do not focus on attribute space fall into two categories. The most prevalent approaches [1], [6], [22], [23], [24], [25] perform extraction and tracking separately. Features are either extracted by spatially coherent attributes or extracted by domain specific hand-crafted feature descriptors. After that, tracking is done by matching the features using spatial overlap over time. Other methods [2], [26], [27], [28] consider features directly in the spatial-temporal space and extract them through topology analysis or high-dimensional iso-surfacing. However, most of these feature exploration methods cannot be directly applied to particle data due to the unstructured nature and the lack of connectivity information among the particles. So there have been some methods designed specifically for particle datasets. Lukasczyk et al. [3] resampled irregular volumetric points into a regular volume before the topology analysis is applied to the data set. Similarly, the 2016 SciVis contest winners [4] proposed an approach that constructs a smooth scalar field before using a contour tree to extract critical points. In the dataset for 2015 SciVis Contest, a dark matter halo merger tree that records how halos evolve is obtained using the method by Takle et al. [29]. Han et al. [30] perform flow map interpolation using ML model on unstructured particle data. The work by Dutta and Shen [22] utilized an incremental Gaussian Mixture Model to both detect and track features. Although their experiments are done with volumetric datasets, their method can be adopted to particle data.

Point Cloud Neural Networks and their Visualization Application: A deep learning survey for 3D point clouds provides a comprehensive overview of the recent point cloud neural network studies [31]. The PointNet [17] is one of the first methods in deep learning for point cloud data. Subsequently, Qi et al. expanded the PointNet with PointNet++ by applying the PointNet recursively on a nested partitioning of the input point cloud to improve efficiency and robustness [16]. PointNet++ borrows the ideas of hierarchically applying filters to extract both local and global features from CNN. Other recent works improve on the basic architecture of PointNet++ [13], [18], [19], [32], [33], [34], [35]. In our work, GeoConv [13] is adopted in our neural networks as it shows good performance with a relatively small network. LassoNet [36], which solves the problem of interactive selection of objects with the lasso is one of the few visualization works we can find that utilizes 3D point cloud neural networks.

3 Learning Local Latent Representation

In this section, we describe how we generate latent vectors for particle datasets using an autoencoder. A classical autoencoder is composed of two parts: the encoder, whose objective is to compress the original data into a reduced dimensional representation called a latent vector, and the decoder, which takes the latent vector as input and reconstructs the original data. Scientific data can be prohibitively large to compute on directly. Therefore, we instead identify salient features in small local regions by organizing the particles into local patches. This not only reduces the memory working set size but also makes it easier for neural networks to learn the local representation of particles. A particle patch is defined as follows. Considering a center location $\vec{p} \in \mathbb{R}^3$ is given for a 3D particle data, the corresponding particle patch $N(\vec{p}, r)$ is defined by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

all the particles whose distance to $\vec{p}$ is less or equal to a distance r: $N(\vec{p},r) = \{\vec{q} \mid \|\vec{p}-\vec{q}\| \leq r\}$.

The autoencoder is used to transform a particle patch into a latent vector. The feature descriptors suitable for the dataset are automatically learned by the encoder, and the latent vector represents the high-level features detected by the learned feature descriptors. Therefore, the latent space has two characteristics that make it more appropriate for feature-related analysis. First, noise in particle patches has less influence on the latent space compared to the influence on the original particle representation [37]. Based on other training samples seen, the encoder detects the most similar feature by the learned feature detection kernel and ignores the noise. This makes the latent vector more stable than raw positions and attributes. This effect can be observed from our reconstruction result in Sect. 5.1.2. Second, the distances in latent space are more consistently related to the presence of high-level features compared to the attribute space for the particle patches because they are encoded by these learned high-level feature detectors. This characteristic of latent spaces is also found in other works utilizing latent representations [12], [14], [15]. Next, we provide the detailed architecture of our autoencoder model.

3.1 Geometric Convolution and Deconvolution

Unlike image or volume data, particle data do not have meshes to connect the particles, hence the convolution kernels used for feature extraction like those in CNNs cannot be easily applied. In this work, we adopt an elegant solution called GeoConv [13] whose core idea is to project the coordinates of a point into orthogonal directions and define a kernel on those directions. The full architecture of GeoConv is shown in Fig. 1 (b). Here, we briefly explain the basic building block of GeoConv that is used to construct our autoencoder. We can define any particle dataset as a mapping from spatial location to d particle attributes, $f : \vec{p} \mapsto X_{\vec{p}}$, where $X_{\vec{p}} \in \mathbb{R}^d$ is the physical attribute at the particle location $\vec{p}$. In a particle patch $N(\vec{p},r) = \{\vec{q} \mid \|\vec{p}-\vec{q}\| \leq r\}$, vector $\overrightarrow{pq}$ is projected into three orthogonal bases $\{\vec{x},\vec{y},\vec{z}\}$, and the projected norm along each basis represents the "energy" in that direction. We aggregate the attributes of a particle $\vec{q} \in N(\vec{p},r)$ into the center $\vec{p}$ based on its energy in different directions. To differentiate positive and negative directions, we define the following six bases in 3D: $B = \{(0,0,1),(0,0,-1),(0,1,0),(0,-1,0),(1,0,0),(-1,0,0)\}$. Then the aggregation of q's attributes to the center p is as following:

$$g(\vec{p},\vec{q}) = \sum_{\vec{b}\in B} cos^2(\overrightarrow{pq},\vec{b}) W_{\vec{b}} X_{\vec{q}} \tag{1}$$

where $\vec{b}$ is one of the basis vectors, $W_{\vec{b}}$ is the learnable weight matrix in the corresponding direction and $X_{\vec{q}}$ is the vector of attributes for particle $\vec{q}$. A 2D example of this projection can be found in Fig. 1 (d). The result of $g(\vec{p},\vec{q})$ is a vector, and vectors for each particle in the patch are sent through one shared fully connected layer (shared FC). We use the term shared FC to refer to a fully connected layer whose weight matrix is shared among different particles. As suggested in the original GeoConv study [13], the dimensionality of the resulting vector $g(\vec{p},\vec{q})$ and the fully connected layer is set to 64 and 256 . This means the weight matrix in GeoConv shared FC (Fig. 1 b) is 64×256 in size, and each of the n particles are multiplied with the same matrix. Shared FC is efficient and can be implemented with 1×1 convolution. Finally, we calculate a vector representation for a particle patch as a summation of all particles in the patch weighted by their distance

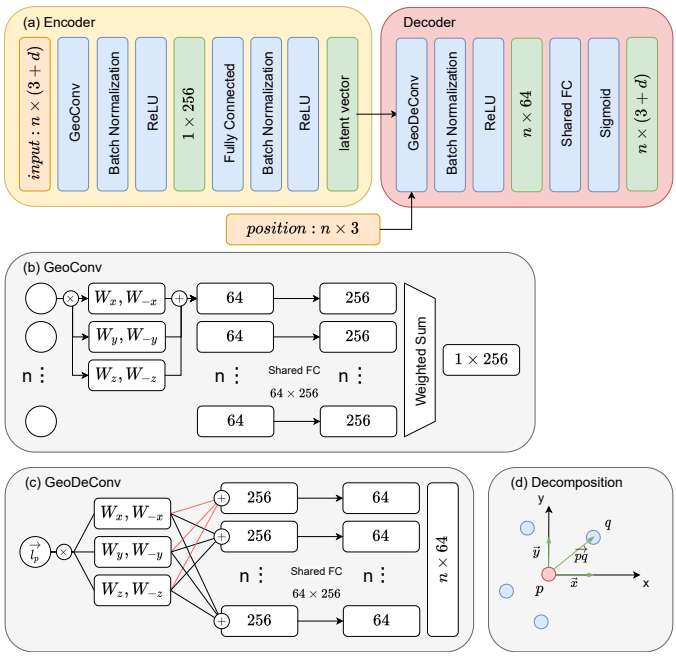


Fig. 1. (a) The architecture of our autoencoder. The input is a particle patch with n particles, each of which has a 3D position and a d-dimensional attribute. (b) GeoConv architecture. The node list on the left stands for the input n particles. The W blocks refer to Equation 1 and the weighted sum block refers to Equation 2. (c) GeoDeconv architecture. The W blocks refers to Equation 4. (d) Decomposition blocks show a 2D example of GeoConv, where $\overrightarrow{pq}$ is projected onto two orthogonal bases.

to the center as shown in Equation 2 and Equation 3. This last step is illustrated in Fig. 1 (b) weighted sum block.

$$GeoConv(\vec{p},r) = \frac{\sum_{\vec{q}\in N(\vec{p},r)} d(\vec{p},\vec{q},r) g(\vec{p},\vec{q})}{\sum_{\vec{q}\in N(\vec{p},r)} d(\vec{p},\vec{q},r)} \tag{2}$$

$$d(\vec{p},\vec{q},r) = \begin{cases} (r - \|\vec{p}-\vec{q}\|)^2, & \text{for } \|\vec{p}-\vec{q}\| < r \\ 0, & \text{for } \|\vec{p}-\vec{q}\| > r \end{cases} \tag{3}$$

Here, distance function $d(\vec{p},\vec{q},r)$ gives more weights to the particles near the center.

Similar to the idea of GeoConv, we propose a new building block for particle neural networks called GeoDeConv. GeoDeConv takes as input a latent vector for the center particle of a patch and relative coordinate for every particle in the patch and outputs a new vector for every particle. This vector is then used to reconstruct the positions and attributes for each particle through shared FCs. We also define the learnable parameters in the fixed orthogonal direction. The GeoDeConv process can be regarded as a dispersion of the latent vector to the neighbor particles. The new vectors $GeoDeConv(\vec{q},\vec{p})$ given the patch center $\vec{p}$ is calculated by:

$$GeoDeConv(\vec{q},\vec{p}) = \sum_{\vec{b}\in B} cos^2(\overrightarrow{qp},\vec{b}) W_{\vec{b}} l_{\vec{p}} \tag{4}$$

where $l_{\vec{p}}$ represents the input latent vector, B represents the same six orthogonal bases, and $W_{\vec{b}}$ are similarly defined as weight matrices in different directions. However, they are not the identical weights used in the GeoConv. We illustrate GeoDeConv in Fig. 1 (c).

3.2 Particle Autoencoder

Having the GeoConv and GeoDeConv as building blocks for the neural network, we propose an autoencoder architecture to transform a particle patch into a latent vector. Our autoencoder

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

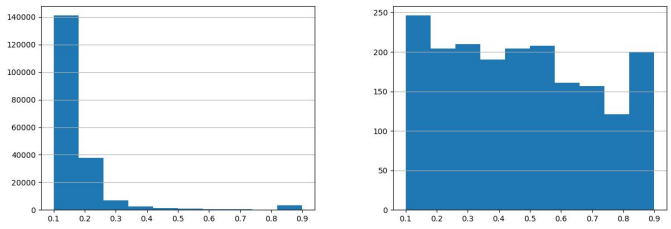


Fig. 2. Left: Distribution of salt concentration of all particles (about 190k particles in total). Right: Distribution of the sampled particles (1900 samples)

architecture is composed of two parts: the encoder and the decoder. Our encoder takes as input a particle patch defined by all the particle positions and related physical attributes and produces a single latent vector $l_{\vec{p}}$. The decoder takes $l_{\vec{p}}$ as input and reconstructs the positions and attributes in the particle patch. We present the architecture of our autoencoder in Fig. 1 (a). Since the particle patch size is relatively small in our experiments, we only use one layer of GeoConv followed by one fully connected layer, and the decoder is constructed using one layer of GeoDeconv followed by one shared FC. Batch normalization is applied after each layer to stabilize the training and to converge faster [38]. We use ReLU as the activation function except for the last layer [39]. Sigmoid is used at the last layer to constrain the output value between $[0,1]$. To train our network, we use a mean squared error loss, measured between the ground truth attributes and the reconstructed attributes. In our experiment, we choose training batches to be as large as our memory constraint allows.

3.3 Data Sampling and Preprocessing

To improve network training, we train our network with samples from all time steps of the particle dataset that capture a more diverse feature set than random sampling. We sample based on the distributions of the particle attributes by drawing fewer samples with values that occur frequently, and more samples that have less frequent attribute values. This sampling rule is based on the observation that very frequent attribute values usually correspond to the background or homogeneous regions with nearly no features. In addition, there is no benefit to train the neural networks with redundant training data. This strategy is also suggested by a recent work of probabilistic data-driven sampling [40]. To realize this sampling strategy, we used a method similar to the value-based sample algorithm in [40]. It can be briefly described by first building a histogram in the particle attribute space and then sampling using the inverse density of the histogram as the weight. The attribute distribution from the Salt Dissolution dataset is shown in Fig. 2 to demonstrate the effect of our sample strategy.

There is a computation overhead to find all particles within a distance r from a particle. To locate the particles in a radius efficiently, we perform a spatial partitioning of the particles using a kd-tree [41] before neighborhood query. We normalize all the particle coordinates to be in the range $[0,1]$ to avoid having to select radius r specific to the data spatial range. After the particles in a patch are queried using the kd-tree, since we are only interested in the particles' local relative positions to the center but not their absolute positions, particles in patch $N(\vec{p},r)$ are translated relative to the center particle $\vec{p}$, and then again normalized to be in the range $[0,1]$ to match the network output range.

3.4 Determine the Network Hyper-parameters

There are several hyper-parameters to decide when constructing our network, such as the dimensionality of the latent space and the radius of particle patches. We discuss our approach to determine the parameters here and evaluate the chosen parameters in Sect. 5.1.1.

We choose the latent dimensionality using a random search strategy. Random samples are drawn from a plausible range of the latent dimensionality (from zero to the dimensionality of input particle patch). The computation budget for each sample is fixed and the parameter that gives the best reconstruction quality is used. The random search strategy is shown to be more efficient than a grid search of hyper-parameters [42], can be performed in parallel, and is usually used in neural network designs [11].

In the GeoConv architecture, the radius of the particle patch equals the bandwidth of the GeoConv kernel (defined by Equation 3). The size of particle patches is of significance in our approach, since choosing an inappropriate radius r will either overly smooth or add noise to the extracted features. We follow the assumption that if the bandwidth is appropriate for estimating a regression function $f:\vec{p}\mapsto X_{\vec{p}}$ between the particle position and its attributes, it will also be suitable to be used in neural networks [43]. We reformulate the problem as a kernel regression estimation. The Nadaraya–Watson estimator [44] using our kernel is written as:

$$\hat{m}(\vec{p};r)=\frac{\sum_{\vec{q}\in N(\vec{p},r)} d(\vec{p},\vec{q},r)X_{\vec{q}}}{\sum_{\vec{q}\in N(\vec{p},r)} d(\vec{p},\vec{q},r)}, \tag{5}$$

where $\vec{p}$ is a point in the 3D spatial domain and r is the kernel bandwidth to be estimated using cross validation. The Least Squares Cross Validation for our Nadaraya-Watson estimator is written as:

$$LSCV(r)=\frac{1}{n}\sum_{i=1}^{n}(X_{\vec{p}_i}-\hat{m}_{-\vec{p}_i}(\vec{p};r))^2, \tag{6}$$

where $\hat{m}_{-\vec{p}_i}(\vec{p};r)^2$ indicates the leave-one-out estimator without the sample point $\vec{p}_i$. Instead of using all particles across every time step from the dataset, we sample 1% of particles from every time step to perform cross validation. The sampling technique is the same as the one presented in Sect. 3.3. We find the particle patch defined by bandwidth r around the sample $\vec{p}_i$ to build the leave-one-out estimator $\hat{m}_{-\vec{p}_i}(\vec{p};r)$ using Equation 5. Following Equation 6, we calculate the $LSCV(r)$ for every time step and average the $LSCV(r)$ across time steps to get the final $LSCV(r)$ for the dataset. Then the golden-section search method [45] is applied to search for the optimal bandwidth r which minimizes the LSCV. Because the particle positions are normalized to be in the range $[0,1]$, we search the optimal r in this range. The estimated bandwidth is then used for the kernel in GeoConv and as the radius for particle patches.

4 Feature Extraction and Tracking

In this section we introduce our feature extraction and tracking methods in detail. A workflow of the extraction and tracking process is described in Fig. 3. After the neural network is trained, we can use it to infer the latent vectors for arbitrary particle patches at any selected time step in the domain. As discussed, latent vectors produced by the autoencoder capture high-level features in particle patches. Therefore, patches with similar features will be in close proximity in the latent space. This enables us to explore and find features by analyzing the clusters in this space. A time step of interest is first selected by users. With the help of our visual analytics system, users identify one or several feature clusters of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

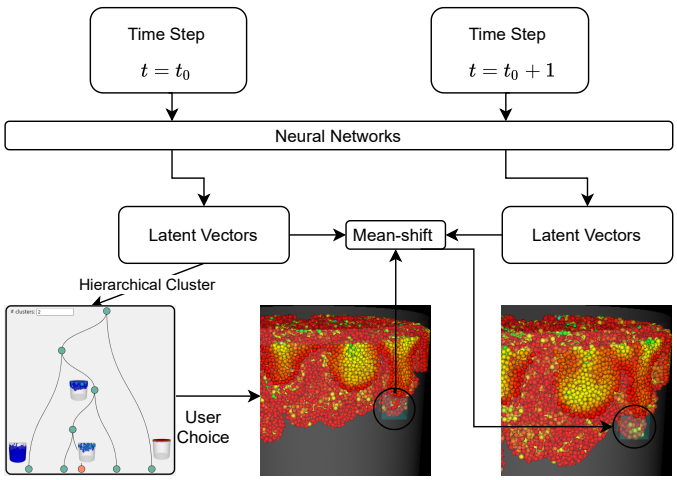


Fig. 3. Scheme showing our proposed approach. The neural network is pre-trained with particle patch samples. The tracking we show in the scheme is only for one time step. Actual feature tracking can be repeated until the desired time step is reached.

interest. After that, these clusters can be visualized in the physical space, and users can select any specific region that contains the feature of interest to track over time. Features are tracked using the mean-shift algorithm to find the matched location in the consecutive time steps within the desired time span.

4.1 Latent Space Visual Exploration

A typical workflow of feature exploration starts with selecting a time step of interest by the user. This can either be done based on prior knowledge about when the interesting features may occur, or by simply selecting once every few time steps to explore. After the time step of interest is found, our visual analytics system can help the user interactively explore the dataset and find the features of interest [1]. This system is composed of three linked parts: a hierarchical cluster view, a 2D feature space view of the latent vectors, and an original physical space view. Domain knowledge from the users are injected by modifying the hierarchical clustering, while feature space and physical space views are updated interactively to provide real-time feedback based on features automatically captured by the latent representation.

4.1.1 Hierarchical Cluster View

Similar to the method used by He et al. [46] and Cheng et al. [11], we arrange the latent vectors in a hierarchical clustering tree. The root of this tree represents all the latent vectors. Links between parent and child nodes depict splitting the latent vectors from the parent node into multiple clusters using the k-means algorithm. Whether to split a node and the number of children (k in k-means) to which the parent node splits are decided by the user. By default, the number of children of each node is set to two but can be increased by the user using the input box based on how many particle groups are shown in the feature space projection (Sect. 4.1.2) of the parent node or when the physical space views (Sect. 4.1.3) of the child clusters are not desirable. We choose the k-means clustering algorithm instead of spectral clustering used by Cheng et al. [11] and agglomerative hierarchical clustering used by He et al. [46] mainly because of its efficiency on large data.

1. Source code is available at https://github.com/harviu/particleNN

The hierarchical cluster view provides a vertical tree layout showing the current hierarchy of the clusters. Initially, it's merely a root node representing all particles in the same cluster. After the users choosing to split any node, the view will be updated interactively. The tree layout we choose arranges all the leaf nodes on the same level, which helps the user focus on the clusters of interest they have just created. An example of the hierarchical cluster view is shown in the left half of Fig. 5.

4.1.2 Feature Space View

We adopt t-distributed stochastic neighbor embedding (t-SNE) as our dimensionality reduction technique and visualize the projected latent vectors as 2D scattered points. This visualization only provides a view of the relations between the clusters, and we note that it may not be sufficient for users to decide the features of interest solely by the 2D projection. However, when the 3D visualization in the original physical space is too crowded, 2D feature space projection may reveal the number of possible distinct features. This helps the user decide whether to split a node and how many child nodes it should split into. To avoid applying t-SNE on all the particles which may require extensive computation time, we applied the same sampling technique as used in training data sampling discussed in Sect. 3.3 to pick around 1% of all particles from the selected time step for visualization.

When a node is selected in the hierarchical cluster view, the t-SNE projection of the corresponding particles will be highlighted to help the user focus on the cluster of interest currently. The right panel in Fig. 5 presents an example for the feature space view.

4.1.3 Physical Space View

This view shows a subset of particles (defined by a selected node) in the original 3D physical space. We assign the colors to the particles which encode a specific scalar attribute of interest. We normalized the coordinates of the particles in the subset to be in the range $[0, 1]$, so that the camera can be set to be always facing the center of the visualized particles. When users interact with a node in the hierarchical cluster view, the corresponding particles are visualized in this view. Users can refer to the visualization to decide whether to split a cluster into smaller clusters or not and finally spot the desired cluster containing the features of interest. Since the clustering result is influenced by k that users choose, users can refer to the physical space view to validate the parameter k they set when splitting the parent node. If the clustering result is not desirable, the users can easily revoke the splitting operation and test with other more suitable values for k. The physical space view is shown in a separate pop-up window in our system. For a better presentation, we display the physical space views beside the cluster nodes and feature groups in Fig. 5.

4.2 Mean-shift Tracking Using Latent Distribution

Traditionally, people perform feature tracking by first extracting features in every time step and then matching them across the time steps. In that case, the tracking efficiency is bounded by the extraction efficiency and can be expensive. The situation becomes worse when the latent inference is time-consuming. Therefore, we adopt mean-shift tracking that finds the local region with the highest latent similarity between consecutive time steps.

First, a region of interest is selected by the user with the help of the system mentioned in the previous section. The v dimensional latent vector for every particle in the cubic region is collected.

We reduce the latent dimensionality to four using the principal component analysis (PCA) to reduce the computation time for the multi-dimensional histogram and to increase tracking stability. Treating each latent vector as a sample, our feature of interest selected by the user can be represented by a four-dimensional histogram. Therefore, our tracking goal between two consecutive time steps is to identify the cubic region that has the most similar latent histogram in the latter time step.

The high-level idea of mean-shift tracking introduced in [47] is that first weights are given to particles in the region of interest according to the comparison between their latent vector with target latent distribution. Then the region is shifted to the higher similarity direction. This is repeated until the region does not move.

Initially, we assume the spatial location of the feature is the same as that of the previous time step, which is used as the starting point for tracking. Then we move the cubic region to the position with most similar distribution using the mean-shift algorithm and follow the parameters suggested in [48]. Repeating the search step on the time sequences will give us the movement of the interested feature through time steps.

5 Experiments

We evaluated our neural networks based latent representations using three datasets: Salt Dissolution, Dark Sky, and SNSPH. Before the evaluation, we present the dataset information and model training details in Sect. 5.1. Several aspects of our approach are evaluated. (1) We validate the methods used for determining the hyper-parameters. (2) We demonstrate the result of feature exploration in three datasets with the help of a hierarchical cluster exploration tool. The extracted features with our latent representations were verified by comparing with features specifically designed for the dataset or provided by the domain scientists. (3) We compare the latent representation with alternative methods, local neighbor statistics, and principal component analysis. (4) We demonstrate that feature tracking using mean-shift on latent space distributions is efficient and accurate. (5) We detail the computation time and memory use for the proposed approach.

5.1 Dataset and Experiment Setting

Our approach was evaluated on three datasets generated from the Salt Dissolution simulation (SciVis Contest 2016), the DarkSky simulation (SciVis Contest 2015), and another supernova simulation called SNSPH. Below we give a brief introduction to the three datasets and their feature extraction target.

The Salt Dissolution dataset is from the 2016 Scientific Visualization Contest [49], which models a solid body of salt dissolved by the water using the Finite Points Method. The original data come with 3 different resolutions and 120 time steps each. We tested on the lowest and highest resolution data which have around $190,000$ and $1,800,000$ particles at every time step respectively. Every particle has four-dimensional physical attributes: a salt concentration value and 3D velocity. The feature extraction target is to find features called *viscous fingers*, which is defined only by the salt concentration.

DarkSky is a cosmology simulation dataset from 2015 SciVis Contest [50]. This particle data set is composed of 100 time steps with around 2,000,000 3D particles in each time step. Every particle has a position vector and seven-dimensional physical attributes: velocity vector, acceleration vector, and gravitational potential energy *phi*. Along with the raw particle information, the data set comes with a list of *halos*, which are features mainly defined by the particle density. The halos are found and tracked by the method proposed by Behroozi et al. [9]. We use the halo list to verify our feature extraction and tracking results based on the latent vectors.

TABLE 1
Hyper-parameters used for different datasets. The particle positions in all datasets are normalized to be in the range $[0,1]$ and the radius in the table is the normalized radius. The number of samples shows the average sample size from each time step of the dataset. Attr. and Conc. stand for Attribute and Concentration respectively in the table.

Dataset	Radius	Attr.	LatDim	#Samples	#Epochs	PSNR
Salt-LowRes	0.05	Conc.	16	1,895	800	**33.21**
	0.01	Conc.	16	1,895	800	31.88
	0.03	Conc.	16	1,895	800	30.21
	0.08	Conc.	16	1,895	800	32.48
Salt-HighRes	0.008	Conc.	16	16,175	500	32.65
DarkSky	0.010	ϕ, v, a	32	20,529	900	35.48
SNSPH	0.005	T	20	9,456	400	**46.40**
	0.001	T	16	9,456	400	45.51
	0.030	T	32	9,456	400	44.30

SNSPH, SuperNova Smooth Particle Hydrodynamics, is our third particle dataset used to model core-collapse supernovae [51]. The dataset comes in a time series of 61 time steps and each time step contains around 900,000 particles. Features of interest are the finger-like structure reaching from the inner core to the outer core and the surface where these finger-like structures form. According to the domain scientists, these features are related to the temperatures and the densities of particles.

5.1.1 Hyper-parameters

To train the autoencoder on three datasets, we chose the hyper-parameters using the method discussed in Sect. 3.4 and sampled the training data using the method in Sect. 3.3. To capture the feature variance in all time steps, we sampled around 1% of particles in each time step for training. We treated the coverage of all samples in one time step as one epoch. The hyper-parameters and training statistics for all datasets are presented in Table 1.

To validate the estimation method for determining the optimal patch radius, we trained four different models for the Salt Dissolution dataset and three different models for the SNSPH dataset. We choose the optimal radius and several smaller or larger radii to train these different models. Details for the radius chosen can be found in Table 1. It is worth noting that training multiple models is not necessary for our approach to determine the radius, we only did that to validate the LSCV estimation result. To qualitatively validate the estimated radius, we compare the data reconstruction and feature extraction between these models. In Table 1, we show that for both datasets, the optimal radius model resulted in better reconstruction peak signal-to-noise ratio (PSNR) than the model trained with a smaller or larger radius. This optimal radius also results in better feature extraction result as shown in Fig. 6 (a).

Another noteworthy hyper-parameter is the latent vector dimensionality. The latent dimensionality shown in Table 1 is determined by a random search technique. We found that for all the datasets, the optimal latent dimensionality is related to the number of particles in the particle patch and the attribute dimensionality we used in the experiment. Moreover, higher dimensionality will not decrease the model reconstruction quality, while slightly smaller dimensionality will only decrease the reconstruction moderately and have negligible influence on the feature exploration results.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

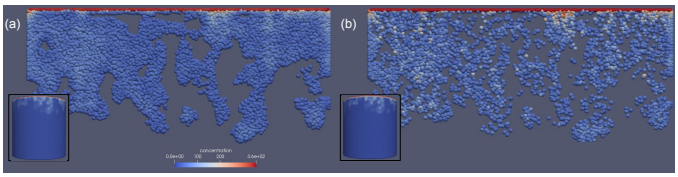


Fig. 4. Reconstructed (a) and original (b). Smaller images in the corner show the full data in one time step. The larger images show the particles filtered with the same salt concentration threshold. In (a) and (b), the color of the particle denotes its concentration value.

5.1.2 Reconstruction Quality

We evaluate the reconstruction quality on the Salt Dissolution dataset. In Fig. 4, we compare a zoomed-in slice along the z-axis of time step 35 in the low-resolution version. This slice was filtered to show only the particles with concentration values higher than 25. We can observe that the reconstructed concentration values are smoother among the reconstructed particles. It was found in the study [4] on this dataset that these high-frequency concentration fluctuations are noise and are to be removed before feature extraction. Therefore, this smoothing effect in the reconstruction is beneficial to our subsequent feature extraction. PSNR between the reconstructed and original data can be found in Table 1. Considering the beneficial smoothing effect, and that we only rely on the clusters of latent vectors to extract features but not to replace the original data with the reconstructed one, PSNR values in this range are acceptable.

5.2 Feature Exploration

Feature exploration is the major use case for our latent vectors produced by the GeoConv autoencoder. The challenges faced in the three datasets are quite different. In the Salt Dissolution dataset, finger features are defined by the gradients in a scalar field. In the DarkSky simulation, halos are defined by the spatial distribution of particles. In the SNSPH simulation, features of interest are defined by a combination of spatial distribution and physical attributes. Through the experiments of these datasets, we demonstrate that the autoencoder-generated latent vectors can capture the salient features dictated by the applications.

5.2.1 Dataset 1: Salt Dissolution

Since the viscous fingers are only related to the particle salt concentration value, we only include the concentration and particle positions to train the autoencoder. To begin feature exploration, we first choose time step 25 as our feature extraction example and the starting point of the tracking, where according to the prior studies [49] on this dataset, the number of viscous fingers starts to increase and their structure start to become complex.

After we produce the latent representations, these vectors are loaded to our exploration system for further analysis. We present our feature exploration process in Fig. 5. At the top level of hierarchical clustering, the dataset is clustered into the salt injection plane (Fig. 5 a) and the water body. We further split the water body cluster to find the finger structure, low concentration background, and the thin finger cores inside as shown in Fig. 5 (b), (c) and (d), respectively. Circles and arrows in Fig. 5 show the correspondence between the cluster particles and their latent space 2D projection. This provides information on how distinct these clusters in the hierarchical cluster view are. However, these particle groups in the 2D projection can be misleading or may not necessarily be

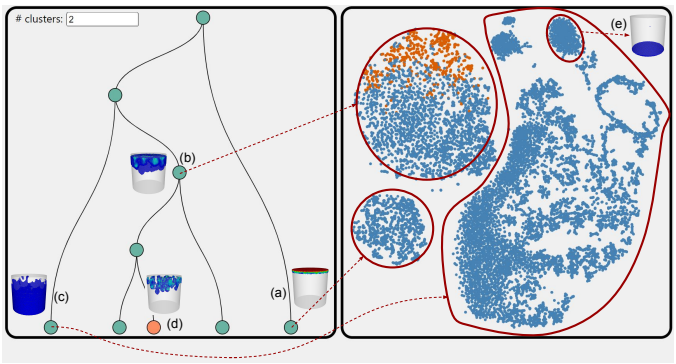


Fig. 5. The left panel contains the hierarchical clusters as the result of latent space exploration. The right panel shows the t-SNE projected particles. The selection of cluster nodes on the left panel will highlight the corresponding particles in the t-SNE view and visualize these particles in the original physical space, where the particles are colored by the salt concentration. Features depicted are salt injection on the border (a), viscous fingers (b), background particles (c), thin cores inside of fingers (d), and a boundary not of interest (e)

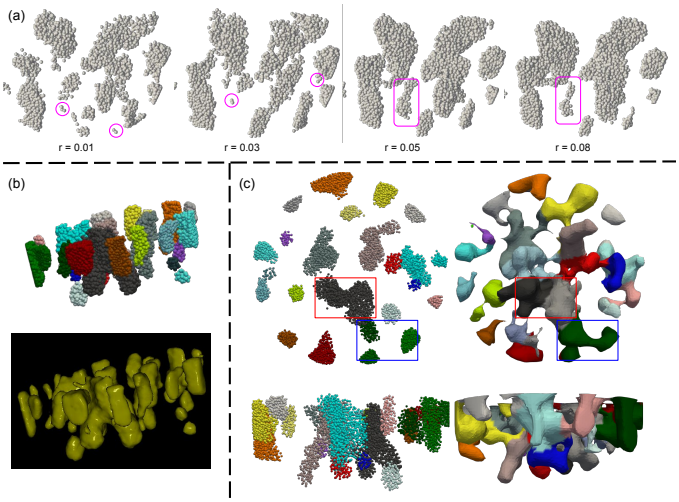


Fig. 6. (a) Extraction results using four different models with different sizes of neighborhood considered. Notice noise (circles) when r=0.01,0.03, and the finger shrinking effect when r=0.88(squares). (b) Extracted fingers shown as particles and surfaces. (c) Different views for comparison between finger extraction through latent vectors and topology analysis. Fingers are colored to best match the same fingers from the two methods. It is observed two methods have different separation of fingers. Some fingers are grouped into one in our methods but divided using topology analysis (red square). Some fingers are in the reversed situation(blue square).

features of interest. For example, the small particle groups at the top of Fig. 5 (e) are corresponding to the boundary particles at the bottom of the water body and are not of interest. This makes it essential to combine both the latent space 2D projection and the physical space particle visualization to explore the features of interest interactively.

We choose the finger core feature in Fig. 5 (d) for further feature analysis. Fig. 6 (a) shows the finger cores both by direct particle rendering and surface extraction. In the direct particle rendering figure (top), we separated the finger cores with the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), so that we can count the number of fingers and verify it both qualitatively and quantitatively by comparing to the features extracted by the winner of the 2016 SciVis contest. Surface extraction (bottom) is done by processing the particle patches defined on a regular grid.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

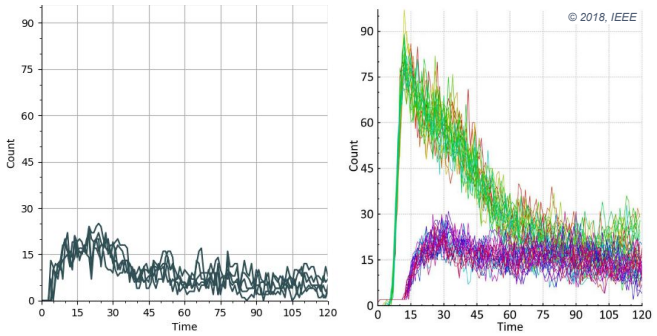


Fig. 7. Left: The extracted finger count across all 120 time steps on 5 different ensemble runs using our method. Right: Finger count from the 2016 SciVis contest winner [4]. Blue and purple lines are from the heuristic clustering method in their work, while green and yellow lines are from topology analysis.

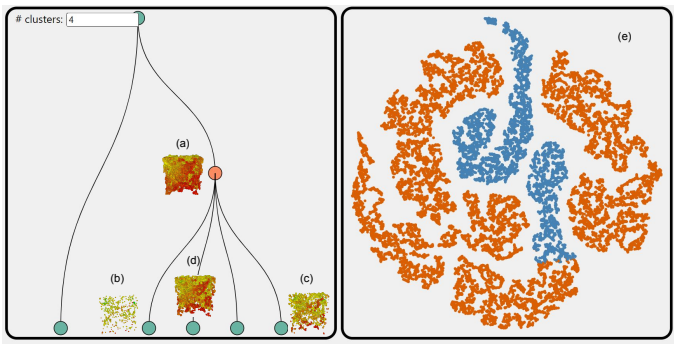


Fig. 8. Feature exploration for time step 49. The figure on the right shows that cluster (a) is corresponding to about four (or more, based on how the user perceives the group connectivity) particle groups in the latent 2D projection. We subdivide the cluster (a) into four lower-level clusters and found that three of them (b), (c), and (d) may contain the halo.

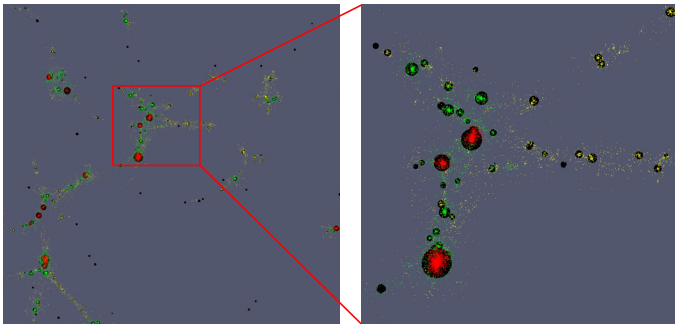


Fig. 9. One slice of the data from time step 49. Particles in Fig. 8 (b), (c) and (d) are colored with red, green and yellow in this figure. We overlay the ground truth halo position and size with black spheres. Large halos are captured by the red cluster and smaller halos are mostly in the green and yellow clusters.

We assign 1 to the patches of finger cores and -1 to the patches of other clusters, and then extract the iso-surfaces at value 0 using marching cubes. The winners of the 2016 SciVis contest proposed two approaches to extract the finger structures [4]. The first one is based on topology analysis on the data and the second one uses slice-wise clustering. We ran the topology-based method and our method on the same data of time step 25. The finger extraction results are shown in Fig. 6 (b). The two methods generate a similar numbers of fingers; 24 are generated from the topology analysis, and 25 are generated from our method. We colored the fingers to match the extracted results using the two methods. However, there are still some noticeable differences between the two results. One observation from the result is that topology analysis generates fingers of larger size. This is likely because the analysis is based on a smooth distance field, which will connect the boundaries between fingers. Secondly, topology analysis and our method separate the fingers differently in space. Since we use DBSCAN in our method to separate them, connected fingers with the same particle density will be identified as one finger. This effect can be seen in Fig. 6 (b) within the red squares. On the other hand, the topology analysis method only uses persistence thresholds to find fingers, which will fail to detect some of them and merge the corresponding region into a single finger as shown in the blue squares. Overall, the finger extraction quality using our proposed approach is comparable to the feature extraction result by topology analysis.

By comparing with the methods [4] specifically designed for this dataset, we verify our proposed approach can capture the target features defined by the particle physical attributes successfully. Across all time steps, our method has a similar trend in finger count to both methods we compare with, and we have a very similar finger count to the heuristic clustering result as shown in Fig. 7. Detected finger count both go up after the first 30 time steps and then drop when the data become chaotic in the latter time steps. It was explained in their work that the number difference between the topology analysis and heuristic clustering is mainly because of the choice of threshold.

5.2.2 Dataset 2: Dark Sky Simulation

Algorithms for locating halos, which use hand-crafted feature descriptors for dark matter halos, primarily focus on the distribution of the particle locations to find the regions of high particle density. The velocity of particles is used in some of the algorithms to exclude the particles with a velocity that exceeds the local escape velocity. Therefore, we include the particle position and velocity to train the autoencoder. Moreover, to make the feature extraction task more challenging, and to verify that our approach could work in situations when the feature-related physical attributes are not known, we also include the acceleration and gravitational potential energy *phi* in the training. After the autoencoder is trained, we followed the same process described above to find the interesting features. In the first level of hierarchical clustering, we found the target feature may be contained in node shown in Fig. 8 (a). There are approximately four (or more, based on how the user perceives the group connectivity) distinct particle groups in the 2D projection of the latent vectors in cluster (a). Therefore, we subdivide the cluster (a) into four lower-level clusters. Based on the physical space visualization of these four clusters, we found three of them contain the target feature of halos. In the case the user chooses a cluster number of more than four and child nodes do not depict desired features, the clustering and splitting step can be easily revoked and the user can change the number of clusters.

To verify the overlap of identified clusters and the ground truth halo features, Fig. 9 shows each particle colored based on their cluster, while the real halos' position and size are drawn as black spheres over the particles. The red cluster (Fig. 8 b) defines the particles in the large halos and the other two clusters mostly include the smaller halos. However, A close look at the left figure in Fig. 9 shows that there are still some halos with a very small size which are not captured by our method. We calculate the percentage of halo particles that are covered by the chosen clusters, where higher percent is better. The red and green clusters along cover

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

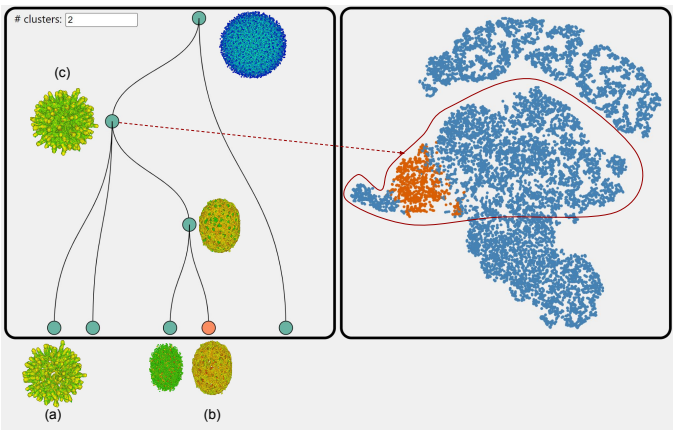


Fig. 10. Left: Hierarchical Cluster View. Right: Latent Space Projection View. Particle visualization in the physical space is only shown in the pop-up windows when the cluster node is selected in the exploration process. Here, particle visualizations are placed beside the cluster node in the left figure for comparison. The latent space projection view also highlights the particles from the selected cluster node.

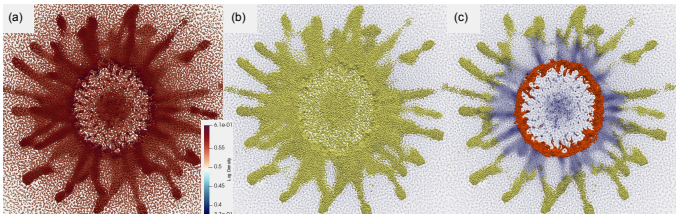


Fig. 11. (a) 2D slice of the supernova simulation with particles colored based on the density. (b) Slice with highlighted particles corresponding to Fig. 10 (c). (c) Slice with highlighted particles. Orange particles correspond to Fig. 10 (a) while yellow particles correspond to Fig. 10 (b).

74.9% of the halo particles across all time steps, while all three clusters (yellow, green, and red) cover 91.9 %. Even though the clusters found by the exploration tool cannot perfectly extract all the halos, the latent representation at least correctly provides the decision boundaries related to the particle density, especially when the irrelevant attributes are included in the purpose to make the task harder.

5.2.3 Dataset 3: SNSPH

With the final dataset, our goal is to verify that our method can capture the features defined by both the particle spatial distribution and physical attributes. As mentioned in the dataset description, SNSPH models a core-collapse supernova. In the process of supernova formation, the inner part of the stellar core is compressed into neutrons. This causes the later infalling material to bounce and form an out-racing shock wave. Scientists are interested in the viscous fingers formed along with the shock wave propagation. Fig. 11 (a) shows a ring of high density. The inner core is roughly enclosed by this ring, and the outer core refers to other parts in the figure. Viscous fingers in this dataset are defined both by temperature and particle spatial distribution (density and the spherical shape). Thus, although there are a total of 91 physical attributes in this dataset, we only choose the temperature to train the neural network.

Using the proposed latent space exploration system, we successfully found two interesting features in this dataset. Our exploration result is presented in Fig. 10. The first split of the hierarchical tree already revealed some interesting structures in the

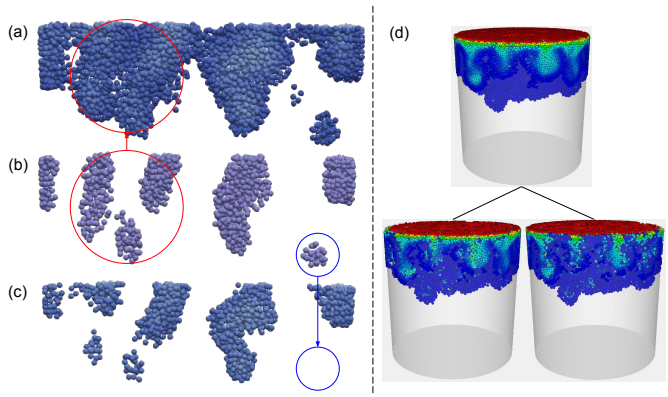


Fig. 12. (a) Average the particle's concentration with particles in a local neighborhood and set a low concentration threshold to extract features. (b) finger core extraction result with latent vectors. (c) The same average set as (a), however, set a high concentration threshold. (d) Three cluster nodes from hierarchical clustering result of PCA. The top cluster shows the high concentration particle clusters found by PCA representation, however, we cannot further divide this cluster to find the finger cores.

dataset. Through the physical space visualization, we find the left-side node cluster shows the finger-like structure (see Fig. 11 (b) for a close-up slice visualization). However, this cluster also includes the particles inside the inner core. Since the domain scientists are interested in the more detailed structure inside the fingers and inner core, we further split the left-side cluster into the second level of hierarchical clustering. Repeating these steps, we successfully find the clusters corresponding to the features of "out-reaching finger" structures as shown in Fig. 10 (a) and the surface between the inner and outer core, where these fingers start to form as shown in Fig. 10 (b). A close up physical space visualization is shown in Fig. 11 (c), where we highlight the "out-reaching finger" feature (Fig. 10 a) in yellow and the finger formation surface (Fig. 10 b) in orange. Since these two features are on different levels of the hierarchical clusters, it is hard to detect both without a hierarchical division of the clusters.

Though the particles forming these two features have high particle density in their regions, the feature depicted in Fig. 10 (b) is defined additionally by high temperature near the supernova inner core and the spherical shape between the inner and outer core. Without the latent vectors produced by the autoencoder in our approach, it is tedious to manually search for the decision boundary in temperature and density to separate these features.

5.3 Comparison with Other Particle Patch Representation

Autoencoders are not the only way to represent the particle patches. In this section, we are going to show that latent vectors produced by the autoencoder are more informative compared with two other methods: representing the patch with a mean in the neighborhood and with principal components. We use the salt dissolution dataset to perform the experiments. Some prior knowledge about the viscous fingers is that they only appear in the region that has relatively high salt concentration. A straightforward way to extract fingers is to filter the particles with a concentration threshold. Therefore, the first method we compare is averaging the concentration in the neighborhood to remove the high-frequency noise and extract features with a threshold. The comparison result is shown in Fig. 12 (a), (b) and (c). It demonstrates the difficulty to determine the threshold if we only remove the noise with an

averaged concentration in the particle patches. A low threshold will make it difficult to distinguish different fingers, while a high threshold will make some fingers disappear. Our neural network method has the advantage of capturing features defined by the change of concentration automatically. It is probably true that we can identify features by calculating concentration gradients and there have been previous work [6] using gradients to identify viscous fingers in a volumetric dataset. However for particle data, calculating gradients is difficult and can be expensive to apply mesh-free methods to achieve this on particles.

Another way is to perform principal component analysis (PCA) on the particle patches and represent each patch with its principal components. We treat each particle as a data sample with four-dimensional variables (three dimensions of position and one dimension of concentration). So we use the four principle components as feature descriptors for the patch. Four principal components are ordered by their explained variance and concatenated into a vector. In Fig. 12 (d), we can find the high concentration clusters which contain our target feature fingers with our hierarchical clustering exploration process. However, further division of this cluster did not reveal any finger cores inside the high concentration region. Particles inside the high concentration region seem to be only randomly assigned into two child clusters. This indicates that principle components of the particle patch failed to capture the information related to the gradient of concentration, which is crucial to define the finger cores and is captured using our method.

5.4 Feature Tracking

As another use case for our autoencoder-generated particle patch representation, successful tracking of features with the mean-shift algorithm in the latent space is demonstrated in this section. In the salt dissolution dataset, since we do not have the ground truth tracking results, we verify the tracking qualitatively by showing the features and the tracking bounding box. As for the DarkSky dataset, we directly compare the tracking result with the halo merger tree generated by [29]. We also compared the latent vector based mean-shift tracking with the original physical attributes based mean-shift tracking in these two datasets.

5.4.1 Viscous Finger Tracking

We select two fingertips from time step 25 and track them consecutively for 20 time steps. In Fig. 13 (a), we present the tracking box at time step 26, 32 and 38 using our latent vectors. The tracking bounding box follows the "fingertip" feature with only a small error. As comparison, we show the tracking bounding box at the same three time steps 26, 32, and 38 using the raw physical attributes in Fig. 13 (c). In the latter two time steps, the tracking box completely went off the feature and was partially blocked by the other particles (indicated by the arrow). In the other fingertip tracking example with latent vectors (Fig. 13 b) and raw attributes (Fig. 13 d), the feature shrinks as time advances, and eventually disappears at time step 45. Compared to the tracking with raw physical attributes, our method still provides better tracking stability. We also provide the tracking animations for these two examples in the supplementary materials.

Generally speaking, tracking when features are small or time resolution is low is more challenging because the mean-shift tracking algorithm we choose relies on the overlap of features between time steps. However, the tracking results prove that our generated latent vectors capture the important features which allow for more accurate and stable feature tracking. The possible reason why the latent vectors work better than the raw attributes in tracking is that the latent vectors encode the features in a more compact space, where some unstable attributes or noise are smoothed or neglected. This effect also helps us easily separate the features in the latent space.

5.4.2 Halo Tracking

For this dataset, Since we had the features' ground truth positions, it is easy to evaluate our feature tracking results by calculating the distance from the center of the extracted feature produced by our method and the real feature center. Similar to the tracking experiment in dataset 1, we also compare the latent vector tracking against using the raw attribute for tracking. Taking the feature size into account, we use $\frac{d}{r}$ as the feature deviation metric, where d is the distance between the centers and r is the ground truth halo radius. If this metric is larger than 1 but smaller than 2, the center of our tracked halo is no longer within the ground truth halo, meaning a slight deviation. If larger than 2, the tracking center and the feature center will have no overlap. We randomly chose 594 halos from the halo list and tracked them for ten time steps. The results of the tracking error are shown in Fig. 14. For both methods, the median deviation generally increases with time and reaches one after ten time steps. In the first five tracking time steps, tracking accuracy with latent vectors is significantly better, while in the last few time steps the tracking accuracy using raw attributes is marginally higher. However, the tracking stability using latent vectors is significantly better than using raw attributes, as the range between the first and the third quartile in latent vector tracking is much smaller.

5.5 Computation Time and Memory Footprint

In this section, we report the computation time and memory consumption for different parts of our proposed approach from the experiments on the three datasets. The machines we used for experiments are as follows: Our autoencoder model is trained on a supercomputer node with 2.4GHz Dual Intel Xeon 6148 (40 cores, but we only use one core in training) and NVIDIA V100 GPU with 16GB video memory. While the extraction and tracking experiments are done on another machine with an 8-core Ryzen 2700X, 32GB of RAM, and an NVIDIA RTX 2060 GPU.

We present the total training time, latent vector inference time, clustering time, t-SNE projection time, and tracking time in Table 2. The model training time is highly related to how many particles we sample in each time step, particle patch size, and how many epochs we train before the network converges. These parameters can be found in Table 1. The autoencoder training needs to be done only once for each dataset. Due to the relatively small number of layers in our network and the training data sampling strategy, our approach's training time is shorter than other works which also use neural networks to extract features [11], [12]. The inference time shown is for generating all particles' latent vectors in a single time step (i.e. forward pass of a trained encoder). This inference process is necessary for latent space exploration and is linear to the number of particles in one time step, which will scale well on larger datasets.

The clustering time shown in the table is calculated by applying the k-means clustering on all the particle latent vectors in one time step, which is the same as the first level clustering in our hierarchical cluster process. This time provides an approximate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2022.3159114, IEEE Transactions on Visualization and Computer Graphics

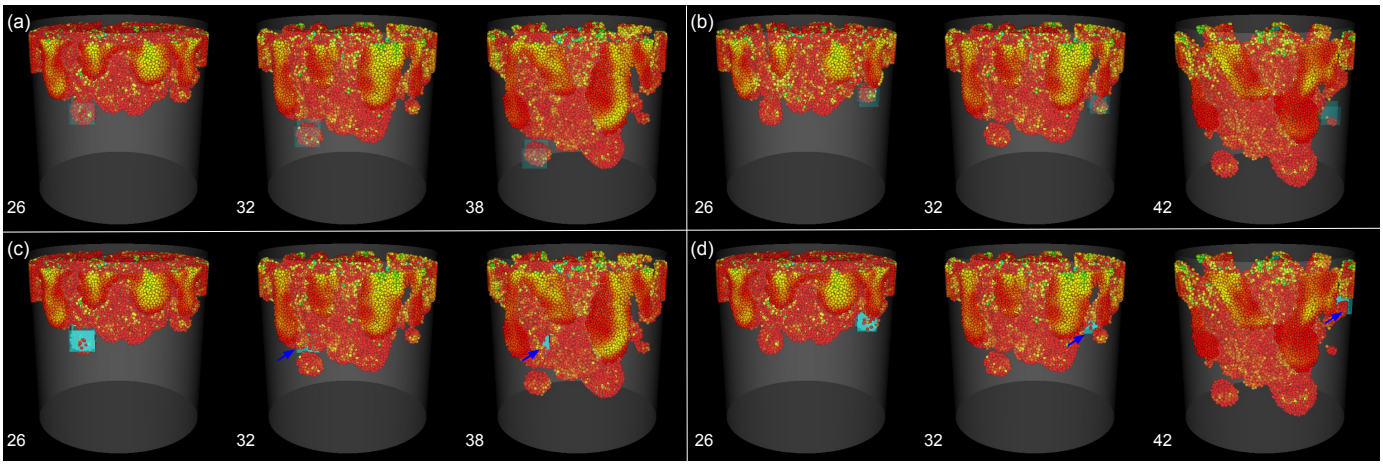


Fig. 13. Time snapshots from two "finger tip" tracking example through time step 25 to 45. (a) and (b) use the latent vectors generated with our approach. (c) and (d) use the raw particle attributes. Time is labeled at the bottom left corner of the image. The user selection of the feature of interest is highlighted by the blue square in the figure. Particles are colored by their concentration values.

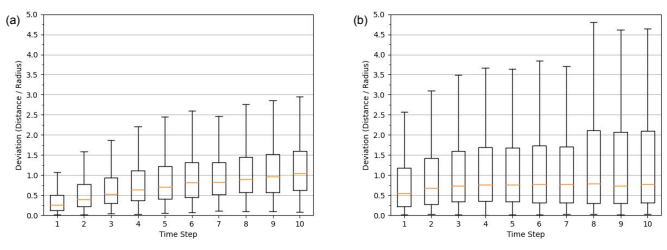


Fig. 14. Tracking deviation for ten time steps from 594 random halos shown with a boxplot. (a) Tracking with latent vectors generated using our method. (b) Tracking with raw attributes.

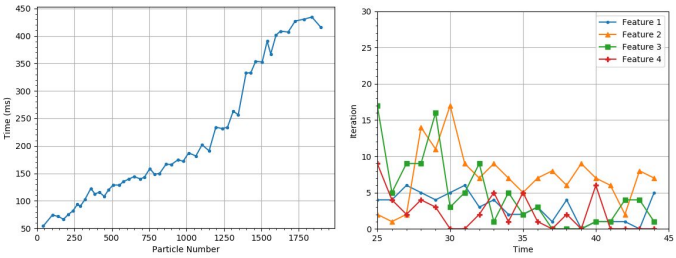


Fig. 15. Left: Relations between the number of particles in the feature region and feature tracking time in the Salt Dissolution dataset. Right: Number of iterations before convergence for 4 different feature tracking examples. Feature 1 and 2 are the same features as shown in Fig. 13.

upper bound for all the user-decided clustering time during the feature exploration process. This quick clustering time facilitates interactive exploratory visualization. Since the k-means time complexity is also linear to the number of particles, this clustering time will also scale well to larger datasets. All particles' latent vectors in one time step are usually too large for t-SNE projection in a reasonable time. As discussed before, the t-SNE projection time shown is for around 1% of all particles. The projection of 20,000 samples from one time step in the DarkSky dataset takes 66.5 seconds. This projection is a preprocessing step before feature exploration and will not influence the system's interactability. Since t-SNE projection does not scale well to a large number of instances. There are two possible solutions if we want to apply our approach to larger datasets: (1) Sample the particles more aggressively to keep a smaller number of samples. (2) Adopt other more efficient projection algorithms such as UMAP [52].

We calculate the computation time for the topology analysis we use to compare our method against for dataset 1. The topology analysis takes 62.5s for the high-resolution version of the dataset. Our approach's preprocessing time (inference + t-SNE projection) for this dataset is on the same order. From the original halo finder work [9], we calculate that their processing time for one time step in the DarkSky simulation is approximately 72s. Considering their time is measured on a legacy 2009 machine (AMD Opteron 2376), the computation time of our method is longer than the original halo feature finder algorithm on this dataset. However, these comparisons only include the actual computation but not the exploration and engineering effort to design the suitable feature descriptor in the comparison methods. Our proposed approach can achieve this exploration semi-automatically at the cost of around 1 hour of model training and seconds of extra preprocessing time.

Lastly, in the process of tracking, the number of iterations before the mean-shift algorithm converges is similar for different features. Therefore, the feature tracking time depends on the size of the feature. A larger feature size means more latent vectors need to be generated. The total tracking time is dominated by the latent vector generation time. For dataset 1, in the case that the feature of interest is a fingertip, which is consisted of approximately 200-400 particles, the tracking time is around 90ms. In Fig. 15, we present the results that show a linear relationship between tracking time and feature size, as well as the number of iterations before convergence.

As for the memory consumption, our method samples the whole dataset and processes the samples in a patch-by-patch manner. Because of this and the relatively small network size, our model can be trained on GPU with lower memory (for example, RTX 2060 with 6GB GPU memory) if we decrease the mini-batch size in training. We provide the size of networks and latent vectors for different datasets in Table 2. When it comes to an even larger dataset where computation time and memory are issues, we can reduce the sample density in the latent vector generation stage to make the method scalable.

6 Discussion, Limitation, and Future Work

Usability. One challenge for particle feature extraction is tedious feature descriptor design for the specific dataset. We overcome this challenge by automatically learning feature descriptors in a data-driven way. With the help of hyper-parameter estimation methods

TABLE 2
Training, inference, clustering, tracking time, and size for the three datasets.

Dataset	#Particle	Training	Inference	Cluster	t-SNE	Tracking	Network Size	Latent Size
Salt Dissolution-Low Resolution	192k	8.4 min	14.3 s	1.9 s	5.4 s	90 ms	285 KB	11.7 MB
Salt Dissolution-High Resolution	1,664k	38.4 min	59.1 s	6.7 s	50.8 s	-	285 KB	101 MB
DarkSky	2,097k	1.25 h	81.4 s	8.5 s	66.5 s	107 ms	302 KB	228 MB
SNSPH	946k	23.6 min	38.2 s	4.8 s	44.1 s	-	289 KB	57 MB

and an interactive hierarchical clustering system, our approach can be easily used by domain scientists. They only need to provide the knowledge on which physical attributes are related and interactively identify the features extracted by the autoencoder.

Scalability. In the experiments, our method works within reasonable computation time and memory consumption on the dataset containing up to 2,000,000 particles. However, the performance on larger datasets could be an issue. This scalability issue is mostly caused by the need to produce a latent vector for every particle in a time step before exploration. This issue could be alleviated by sampling a subset of particles to produce latent vectors, which may lead to reduced extraction quality or artifacts.

Interpretability. Even though latent space visualization methods such as our approach and others [11], [12], [14], [15] are helpful in different applications, latent spaces are still not fully understood. We provide some intuitions and evidence on the characteristics of our latent spaces, such as how latent representations remove noise and extract high-level features. How to understand neural network based latent vectors and why distances in the latent space are useful are still open questions.

In the future, scalability and interpretability are two fields where we want to explore. Compression methods that quantize the latent vectors in [11] could be a solution for the scalability issue on large datasets. Another possible way is to sample the representative particle patches in the datasets, which will reduce the computation cost. How to define representative patches and design a sampling technique are the keys to this problem. Regarding latent space interpretability, studies in the field of explainable AI provides some insights. Visual analytics systems can be useful to probe the encoded high-level features in different dimensions of the latent space. Understanding the encoded features will help us diagnose and improve the model.

7 Conclusion

In this paper, we show that latent vectors produced from a GeoConv based autoencoder are useful for feature extraction and tracking. We demonstrate successful feature extraction results in three different particle datasets and show how mean-shift tracking is useful to track features through the latent space for two of these datasets. In the first dataset, we show our latent vector can reconstruct particle patches with less noise than raw data. The extraction results for viscous fingers, which are defined by concentration, are superior to the threshold method or representing the neighborhood with principle components. We validate the extraction results by comparing them with the SciVis Contest winner's approach. In the second and third datasets, we show that the latent representations can capture the features related to particle spatial distribution and physical attributes. Finally, the detailed performance of our approach is provided. The efficiency of the approach is comparable to the benchmark methods, and the effort to design and engineer specific feature descriptors for the dataset is not required in our approach.

Acknowledgments

This work is supported in part by the National Science Foundation Division of Information and Intelligent Systems-1955764, the National Science Foundation Office of Advanced Cyberinfrastructure-2112606, U.S. Department of Energy Los Alamos National Laboratory contract 47145, and UT-Battelle LLC contract 4000159447 program manager Margaret Lentz.

References

[1] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing Features and Tracking Their Evolution," *Computer*, vol. 27, no. 7, pp. 20–27, 1994.

[2] G. Ji, H. W. Shen, and R. Wenger, "Volume Tracking Using Higher Dimensional Isosurfacing," *Proceedings of the IEEE Visualization Conference*, pp. 209–216, 2003.

[3] J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, and H. Leitte, "Viscous Fingering: A Topological Visual Analytic Approach," *Applied Mechanics and Materials*, vol. 869, pp. 9–19, 2017.

[4] P. Gralka, S. Grottel, J. Staib, K. Schatz, G. Karch, M. Hirschler, M. Krone, G. Reina, S. Gumhold, and T. Ertl, "2016 IEEE Scientific Visualization Contest Winner: Visual and Structural Analysis of Point-based Simulation Ensembles," *IEEE Computer Graphics and Applications*, vol. 38, no. 3, pp. 106–117, 2018.

[5] M. Monfort, T. Luciani, J. Komperda, B. Ziebart, F. Mashayek, and G. E. Marai, "A deep learning approach to identifying shock locations in turbulent combustion tensor fields," *Mathematics and Visualization*, no. 9783319613574, pp. 375–392, 2017.

[6] J. Xu, S. Dutta, W. He, J. Moortgat, and H.-W. Shen, "Geometry-driven detection, tracking and visual analysis of viscous and gravitational fingers," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[7] J. Caban, A. Joshi, and P. Rheingans, "Texture-based feature tracking for effective time-varying data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1472–1479, 2007.

[8] M. Jiang, R. Machiraju, and D. S. Thompson, "A novel approach to vortex core region detection." in *Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization*, vol. 2, 2002, pp. 217–225.

[9] P. S. Behroozi, R. H. Wechsler, and H.-Y. Wu, "The rockstar phase-space temporal halo finder and the velocity offsets of cluster cores," *The Astrophysical Journal*, vol. 762, no. 2, p. 109, 2012.

[10] C. Correa and K.-L. Ma, "Size-based transfer functions: A new volume exploration technique," *IEEE transactions on visualization and computer graphics*, vol. 14, no. 6, pp. 1380–1387, 2008.

[11] H. C. Cheng, A. Cardone, S. Jain, E. Krokos, K. Narayan, S. Subramaniam, and A. Varshney, "Deep-learning-assisted volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 2, pp. 1378–1391, 2019.

[12] J. Han, J. Tao, and C. Wang, "FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 4, pp. 1732–1744, 2020.

[13] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3d point clouds using geo-cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 998–1008.

[14] W. P. Porter, Y. Xing, B. R. von Ohlen, J. Han, and C. Wang, "A deep learning approach to selecting representative time steps for time-varying multivariate data," in *2019 IEEE Visualization Conference (VIS)*. IEEE, 2019, pp. 1–5.

[15] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang, "V2v: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1290–1300, 2020.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 5100–5109, 2017.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, 2019.

[19] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 8159–8170, 2019.

[20] J. Jankowai and I. Hotz, "Feature level-sets: Generalizing iso-surfaces to multi-variate data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 2, pp. 1308–1319, 2018.

[21] L. Linsen, T. Van Long, P. Rosenthal, and S. Rosswog, "Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1483–1490, 2008.

[22] S. Dutta and H. W. Shen, "Distribution Driven Extraction and Tracking of Features for Time-varying Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 837–846, 2016.

[23] G. Ji and H.-W. Shen, "Feature tracking using earth mover's distance and global optimization," in *Pacific graphics*, vol. 2. Citeseer, 2006.

[24] D. Silver and X. Wang, "Tracking scalar features in unstructured datasets," *Proceedings of the IEEE Visualization Conference*, vol. 98, pp. 79–86, 1998.

[25] J. Chen, D. Silver, and L. Jiang, "The feature tree: Visualizing feature tracking in distributed AMR datasets," *PVG 2003, Proceedings - IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003*, pp. 103–110, 2003.

[26] G. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci, "Feature tracking using reeb graphs," in *Topological Methods in Data Analysis and Visualization*. Springer, 2011, pp. 241–253.

[27] F. Sauer and K. L. Ma, "Spatio-temporal feature exploration in combined particle/volume reference frames," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 6, pp. 1624–1635, 2017.

[28] C. Muelder and K. L. Ma, "Interactive feature extraction and tracking by utilizing region coherency," *IEEE Pacific Visualization Symposium, PacificVis 2009 - Proceedings*, pp. 17–24, 2009.

[29] J. Takle, K. Heitmann, T. Peterka, D. Silver, G. Zagaris, and S. Habib, "Tracking and visualizing evolution of the universe: In situ parallel dark matter halo merger trees," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE, 2012, pp. 1482–1483.

[30] M. Han, S. Sane, and C. R. Johnson, "Exploratory lagrangian-based particle tracing using deep learning," *arXiv preprint arXiv:2110.08338*, 2021.

[31] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[32] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[33] W. Wu, Z. Qi, and L. Fuxin, "PointCONV: Deep convolutional networks on 3D point clouds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 9613–9622, 2019.

[34] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 820–830, 2018.

[35] P. Hermosilla, T. Ritschel, P.-P. Vázquez, À. Vinacua, and T. Ropinski, "Monte carlo convolution for learning on non-uniformly sampled point clouds," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–12, 2018.

[36] Z. Chen, W. Zeng, Z. Yang, L. Yu, C. W. Fu, and H. Qu, "LassoNet: Deep Lasso-Selection of 3D Point Clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 195–204, 2020.

[37] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning*, vol. 1, pp. 448–456, 2015.

[39] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *27th International Conference on Machine Learning*, 2010, pp. 807–814.

[40] A. Biswas, S. Dutta, E. Lawrence, J. Patchett, J. C. Calhoun, and J. Ahrens, "Probabilistic Data-Driven Sampling via Multi-Criteria Importance Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 8, pp. 1–1, 2020.

[41] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[42] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[43] B. Warner and M. Misra, "Understanding neural networks as statistical tools," *The american statistician*, vol. 50, no. 4, pp. 284–293, 1996.

[44] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[45] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American mathematical society*, vol. 4, no. 3, pp. 502–506, 1953.

[46] W. He, H. Guo, H. W. Shen, and T. Peterka, "EFESTA: Ensemble Feature Exploration with Surface Density Estimates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 4, pp. 1716–1731, 2020.

[47] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 7, pp. 142–149, 2000.

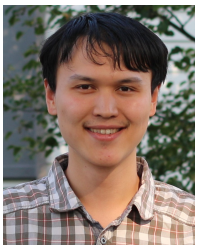
[48] A. Asvadi, P. Girão, P. Peixoto, and U. Nunes, "3d object tracking using rgb and lidar data," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1255–1260.

[49] J. Kuhnert, "Meshfree numerical schemes for time dependent problems in fluid and continuum mechanics," *Advances in PDE modeling and computation, New Delhi:Ane Books*, pp. 119–136, 2014.

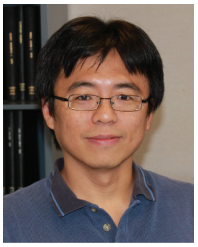
[50] T. Christoudias, C. Kallidonis, L. Koutsantonis, C. Lemesios, L. Markou, and C. Sophocleous, "Visualising the dark sky IEEE SciVis contest 2015," *2015 IEEE Scientific Visualization Conference, SciVis 2015 - Proceedings*, pp. 79–86, 2016.

[51] C. L. Fryer, G. Rockefeller, and M. S. Warren, "SNSPH: A Parallel Three-dimensional Smoothed Particle Radiation Hydrodynamics Code," *The Astrophysical Journal*, vol. 643, no. 1, pp. 292–305, 2006.

[52] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.



Haoyu Li is a Ph.D. student in the Department of Computer Science and Engineering at the Ohio State University. He received his B.S. degree in Psychology from Beijing Normal University in 2017. His research interests are mainly in visualization for particle simulations and machine learning for scientific visualization.



Han-Wei Shen is a full professor at the Ohio State University. He received his B.S. degree from Department of Computer Science and Information Engineering at National Taiwan University in 1988, the M.S. degree in computer science from the State University of New York at Stony Brook in 1992, and the Ph.D. degree in computer science from the University of Utah in 1998. From 1996 to 1999, he was a research scientist at NASA Ames Research Center in Mountain View California. His primary research interests are scientific visualization and computer graphics. He is a winner of the National Science Foundations CAREER award and U.S. Department of Energys Early Career Principal Investigator Award. He also won the Outstanding Teaching award twice in the Department of Computer Science and Engineering at the Ohio State University.