Novel extraction and tracking method used in multiplelevel for computer vision

Ekhlas Watan Ghindawi, Sally Ali Abdulateef
Department of Computer Science, Al-Mustansiriyah University, Baghdad, Iraq

Article Info

Article history:

Received Feb 10, 2023
Revised Apr 11, 2023
Accepted Apr 15, 2023

Keywords:

Deep learning
Feature extraction
Feature selection
Particle data
Tracking algorithms

ABSTRACT

Various scientific applications, including cosmological simulation, fluid simulation, and molecular dynamics, depend heavily on the analysis of particle data. Although there are techniques for feature extraction and tracking regarding volumetric data, it is more difficult to do such tasks for particle data due to the lack of explicit connectivity information. Even though one could transform the particle data to volume beforehand, doing so runs the risk of incurring error and growing the data size. In order to facilitate feature extraction and tracking for scientific particle data, we adopt a deep learning (DL) method in this research. In order to capture the relation between physical features and spatial locations in a neighborhood, we use a DL model that generates latent vectors. Through clustering the latent vectors, characteristics could be retrieved from the vectors. The Cam-shift tracking algorithm, which just needs inference of the latent vector for chosen regions of interest, is implemented in the feature space to accomplish quick feature tracking. With the use of two datasets, we test our approach and contrast it with other approaches already in use.

Corresponding Author:

Ekhlas Watan Ghindawi
Department of Computer Science, Al-Mustansiriyah University
Baghdad, Iraq
Email: ikhlas_watan@uomustansiriyah.edu.iq

1. INTRODUCTION

The problem of the feature extraction and tracking has turned into a crucial component for sufficient scientific data analyses due to the increasing complexity and size of simulations. Scientists could better understand the scientific phenomena evolution via extraction and tracking of features, and they could also get around problems with large-scale data visualization, like 3D occlusion and expensive I/O, by concentrating only on features of smaller sizes and lower complexity rather than the entire dataset. Feature extraction and tracking techniques have a shared strategy, which is finding temporally coherent and spatially unique regions depending upon a few certain features like size, values, topology, texture or shape. This approach is present in both early work [1] and more recent methods like deep learning (DL) [2], and topology analysis [3].

The challenges of feature extraction and tracking for particle data, like those generated by simulations of fluid flow, cosmology, or molecular dynamics, are still great. The lack of connectivity between particles makes it difficult to define features based on gradients or textures, or directly apply methods such as topology analysis and region growing. One common solution is to re-sample the attributes of particles into a regular grid and then perform feature extraction and tracking [4]. Re-sampling data, on the other hand, might result in lost information or a considerable increase in the amount of data storage required.

While it is true that particles produced by pure Lagrangian methods are uniquely indexed throughout the whole-time sequence, and hence tracking particles can be more easily done, particle tracking does not equate to feature tracking. A feature may evolve over time and hence the set of particles that define the feature will evolve as well. In this case, while feature tracking can be assisted by tracking the movement of corresponding particles, it is still necessary to perform feature tracking to identify the dynamic set of feature particles in all time steps [5]. Moreover, there are situations when particle indices are not available [6] or discarded to reduce the storage size.

Along with the inherent challenges regarding particles data, finding an adequate definition for the salient characteristics could be complicated because of complexity regarding various scientific phenomena. Simple thresholds of attribute values cannot often separate various features, and manually crafting feature descriptors for extracting features is time-consuming and not often reliable. To solve this problem, deep learning has been used to automatically extract feature descriptors based on the knowledge of domain experts or feature statistics [7]. The aforementioned two approaches, however, are not appropriate for “soft-knowledge features”, which are defined by Luciani *et al.* [8] as features that are difficult to be captured and represented computationally, such as viscous fingers in the particle data from 2016 SciVis Contest. Providing feature labels by domain experts as training data is time consuming and difficult without a method to solve the problems of efficient visualization first.

In the presented research, we present a new neural network (NN)-based method to address the two aforementioned significant challenges: i) the lack of connection in particle data and ii) the difficulties in using "soft-knowledge" about features. Throughout this research, we will refer to one particle and its local neighborhood as particle patch. NNs interpret this data to create a latent vector without explicitly requiring connectivity. Feature extraction is then achieved by grouping particles of similar latent vectors into clusters. Our network model is based upon a state-of-the-art architecture [9] for point cloud data input with a modification so that we can enforce that the features learnt by the model are the relation between particle attributes and their spatial location in the local neighborhood. Our model does not need any feature labels in the training data, and instead an unsupervised training scheme called autoencoder is adopted. The domain expert’s knowledge is injected by exploring and choosing the plausible features from clusters of latent vectors. The Cam-shift tracking algorithm, which is based on the distributions of latent vectors, is then used to track the selected features. The suggested method can be used with noisy and multivariate datasets, which adds to the difficulty of feature extraction.

Contributions of the present study can be summarized as:

– Feature extraction with a novel NN architecture that has been designed particularly for particle data that can capture the spatial- physical attribute relation in the local neighborhood without explicit spatial connectivity.
– Fast feature tracking approach that exploits Cam-shift tracking using latent distributions.
– A demonstration of applying our method to successfully perform feature extraction and tracking for two noisy and multivariate particle datasets.

The rest of this research is organized as follows: we outline the related literature for this work in section 2. We describe our suggested method in sections 3 and 4, going from an overview to specifics. In section 5 we evaluate the approach using available particle dataset. Section 6 contains the discussion and the future work of this study.

2. THE PROPOSED SYSTEM

Particle simulations are used across a plethora of scientific fields. Analyzing and understanding this kind of data is a hot topic, as evidenced by the fact that 2015, 2016 and 2019 scientific visualization contests all focused on particle datasets [6], [10], [11]. For each of these datasets, the tasks of feature extraction and tracking were critical for understanding the development of the simulated phenomena over time. Recent work from Han *et al.*[12] also proposed a method to interactively visualize large scale particle datasets by representing them with Gaussian mixture models (GMMs). The method which is suggested in the work has been inspired by Cheng *et al.* [7] DL assisted volume visualization, in which supervised deep learning NNs are utilized as a tool for extracting significant features to aid in the creation of a good transfer function for volume rendering. Current DL research has developed and deployed NN models to point cloud data [13]-[17] with significant success for segmentation and classification, both of which involve the extraction of meaningful features from point clouds. Since successful classification and segmentation indicates successful feature learning from the dataset, we survey former studies in both the fields of feature extraction and tracking and point cloud neural networks. Moreover, in this paper, the Cam-shift tracking algorithm from the domain of computer vision is used. Therefore, we also give a brief survey of this technique.

2.1. Feature extraction and tracking

The majority of methods of feature extraction and tracking may be divided into two groups. Tracking and extraction are carried out independently in the first and most prevalent approaches [18]-[23]. Either spatially coherent properties or manually created feature descriptors to a particular domain are used for extracting features. The characteristics are then matched with the use of spatial overlap over time to perform tracking. The second approach [24]-[26] extracts characteristics using topology analysis or high-dimensional iso-surfacing through directly considering them in the spatial-temporal space. It's important to note that feature extraction and tracking have both been accomplished using machine learning (ML) approaches. A method to extract features and create transfer functions with the use of ML for flow field data was put forth in an early work through [27]. DL could be utilized for detecting shock locations in turbulent combustion tensor fields. Yet, because there is little information available about the connection of the particles, the majority of feature extraction techniques cannot be applied directly to particle data. For particle datasets, there were some research attempting to solve this problem. By using a new data structure for octrees extended their previous volume feature extraction and tracking technique to unstructured datasets [22]. Prior to doing topological analysis on the data set have re-sampled irregular volumetric points to one regular volume [2]. Similar to this, 2016 SciVis contest winners [6] suggested a technique that first create a smooth scalar field before extracting crucial points with the use of a contour tree. In problem to solve features from a particle data set that were defined over many variables have employed a region growing approach in attribute space [28]. An incremental Gaussian mixture model was used in Shen and Dutta's work [19] to both track and detect features. Even though they used volumetric datasets for their tests, their approach may be applied to scattered data. These studies, on the other hand, either fail to adequately demonstrate their usefulness on particle data or are unable to extract "fuzzy" or "soft" features.

2.2. Point cloud NNs and their visualization applications

A thorough summary of the most recent point cloud NN studies can be found in a DL survey for 3D point clouds [29]. One of the earliest approaches to DL for point cloud data is the PointNet. In order to increase robustness and efficiency. Had extended PointNet with the PointNet++ through the recursive application of the PointNet on nested partitioning of input point cloud [30]. PointNet++ borrows ideas of the hierarchical application of the filters for extracting the global as well as the local features from convolutional neural networks (CNN). Numerous recent studies [31]-[34]; enhance PointNet++'s fundamental design. GeoConv [9], which has strong performance with a relatively modest network size, is incorporated into our work and used in our own NNs. One of rare visualizations works we can discover that makes use of 3D point cloud NNs is LassoNet [35], which addresses the issue of the interactive selection of the objects by using lasso.

2.3. Cam-shift tracking

The first application of Cam-shift tracking was proposed by Comaniciu *et al.* [33], where they track non-rigid objects in video clips by comparing color histograms between frames. It can achieve real-time performance with high accuracy, and so is widely utilized in computer vision applications. There have also been Cam-shift applications for point clouds. Asvadi *et al.* [34] utilized Cam-shift in 3D point cloud object tracking in the application of autonomous cars. Hermes *et al.* [35], Cam-shift tracking was used on 3D point cloud laser sensor data to track traffic participants. In our work, the Cam-shift algorithm is applied in a high-dimensional latent space which is learnt by a neural network trained on scientific particle datasets.

3. METHOD

In this work, latent vectors are produced to represent particle patches, and features are extracted and tracked based on the latent representations. A workflow of our approach is described in Figure 1. Our approach can be described in the following steps.

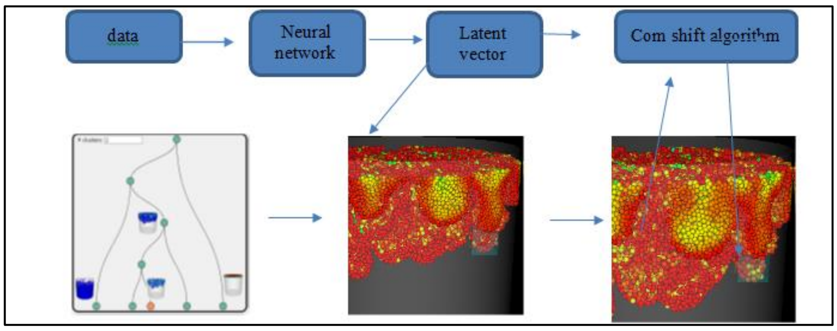

Figure 1. Scheme showing the suggested method. With particle patch samples, the NN has been pre-trained. Only one time step is included in the tracking we display in the scheme. Up until the desired time step is reached, actual feature tracking could be repeated

First, a small sample of the data is selected as the training samples, and it is then organized into particle patches, or groups of particles in local neighborhoods. An autoencoder that takes a particle patch as input, creates a latent vector, and after that reconstructs the input patch with the use of latent vector is trained with the use of the training samples. The NN could also be used to indicate latent vectors for the particle patches at any chosen time step in domain once it has been trained. K-means clustering regarding latent vectors is performed in the latent space for revealing plausible features. The high-dimensional latent vectors are mapped to two dimensions using t-SNE projection to assist users visualize and interact such latent features. This allows users to overcome 3D occlusion and quickly circle the features they are interested in. The user could then visualize the detected feature clusters in the spatial domain and choose a specific region which includes a feature of interest to track it over the time. The Cam-shift approach is used continuously to discover the matched location in the following time step until needed time span is attained in order to track features by maximising the similarity in the distributions created from latent vectors in local regions throughout time. Below, we go into depth about each stage of our algorithm.

3.1. Extraction and tracking of features

We have provided a thorough explanation of how latent vectors can be used for representing particle patches. Using an important feature extractore in proposed system was had major role to benefit result. We give a detailed explanation of our feature extraction and tracking approaches in the following sections.

3.1.1. Exploration of the latent space

A particle patch's compact feature descriptor is the latent vector the NN generates. As a result, in the latent space, patches with comparable features are going to be in a close proximity. Thus, we may examine the clusters in this space and find features. The user can visualize and interact with the latent vectors' distribution using an t-SNE projection of the latent vectors to 2D in our technique, which performs cluster analysis through k-means clustering in high-dimensional latent space. Our choice of the clustering technique, k-means, is mostly based on its effectiveness with large data.

The user must choose a time step that interests them in the first step of exploration. Based on the knowledge of when the interesting features, this could be done, or the user might just choose to explore once every few times steps. To reduce the computation cost for clustering and projection of large amounts of latent vectors, instead of computing the latent vectors for patches around every particle in one time step, we choose particle patch centers with a regular interval, which can be covered by the r sphere similar to what is used for network training. This greatly reduces the number of latent vectors and will not impact the clustering result much, since adjacent particle patches tend to have similar latent vectors. We cluster the latent vectors in their original space using a standard k-means algorithm and color the projected point with the k-means cluster id. Plausible features are already segmented and identified by clustering. However, it is difficult for users to explore the clusters for finding clusters of interest in the high dimensional latent space or in the 3D spatial domain, especially when the number of clusters is large. To alleviate this problem, latent vectors are first projected to 2D space using a standard implementation of t-SNE projection, and then users can visually find clusters and circle the interested ones as shown in Figure 1. This projection also helps the user determine the number of clusters used in k-means, which will have impact on the feature extraction result. For example, if the number of clusters is set to too high for k-means, many small clusters will be produced, and in the 2D space no clear boundaries can be seen to differentiate different features. Users can increase or decrease the number of clusters based on this projection view, so that a more accurate clustering can be achieved.

Patch-wise feature extraction gives us a coarse segmentation of the features. To obtain a particle-wise feature segmentation, an average latent vector is calculated for the circled clusters in the last step. We then infer the latent vectors for the patches around every particle and filter them using its distance to the average vector. Since our network size is relatively small, it is shown in the result that inference of millions of points can be finished in reasonable time.

3.1.2. Cam-shift tracking with the use of latent distribution

The aforementioned feature extraction method involves identifying features of interest from clusters of latent vectors at the time step of interest. Once the interested region is found, the next task is to know how the feature develops in the future time steps. Traditionally, features are extracted at each time step, and after that, they are matched across time steps in order to accomplish feature tracking. In that situation, the extraction efficiency limits the tracking efficiency, which might be costly. When the latent inference takes a while, the problem gets worse. In order to track the latent similarities between time steps, we suggest a feature tracking method. First of all, a return on investment (ROI) is chosen via the user as mentioned in the previous section, where a bounding box in the spatial domain can be computed. This can be easily done since features are already identified in the current time step of interest. Users can simply find and select one or several features they want to track over time. The v dimensional latent vector for each one of the particles in cubic area has been collected. We first reduce the latent dimensions to four using principal component analysis (PCA) so that only the most important dimensions of the latent vectors in PCA space are used to reduce the computational overhead of tracking. Our user-selected feature of interest could be represented via a 4D histogram through treating every one of the latent vectors as sample. As a result, the tracking goal is to find cubic area which has the most similar latent histogram in later time steps between two successive time steps. Since we are unsure of the latent vector's full range of values, we dynamically establish the ranges for histogram creation using samples from the most recent and earlier time steps. We initially assume that the feature's spatial location corresponds to the location of preceding time step, which serves as a tracking starting point. After that, we employ Cam-shift technique that has been developed by Comaniciu *et al.* [33] for moving the cubic region to its most comparable position. We can track the movement of targeted feature via time steps by the repetition of the search step on time sequences. The user can also choose multiple regions to track. Tracking multiple features can be achieved efficiently through parallel computing.

The fundamental principle of Cam-shift tracking is that we first assign various weights to the particles in the target region depending on how their latent vector compares to the target latent distribution. The area is then moved to direction with a higher weight. Up till the area stays still, this is done repeatedly. We follow the same technique in [36]-[38], where they use Epanechnikov kernel to smooth the Cam-shift optimization space. There are two hyperparameters we need to deter-mine for tracking. The first is the number of bins to build the latent histogram. Since our latent space is high dimensional and samples are relatively sparse, we set it to a small number.

4. RESULTS AND DISCUSSION

This section demonstrates how our proposed extraction and tracking algorithm works on datasets is from 2016 scientific visualization contest. A cylindrical flow domain containing water makes up the basic simulation setup [6]. A solid body of salt that has been dissolved by water is at top of the cylinder and is modeled by a corresponding boundary condition. This setup serves as the foundation for each simulation of the given ensemble. Each simulation has about 120-time steps to give significant temporal resolution because of the transient nature of the solution process. The resolution we utilized, which was one of three available for the original data, contains roughly 190,000 particles per time step. A concentration value and a three-dimensional velocity are two of a particle's four physical attributes. These particles are distributed in a cylindrical space whose radius is 5 units and height are 10 units. The feature extraction and tracking goal is to find the finger-like high salt concentration areas.

4.1. Autoencoder training and reconstruction

We sampled about 1,500 particles in each of the 120-time steps, employing the previously mentioned approach, for capturing the feature variance. We considered the coverage of all samples during 120 training steps to be one epoch. In the case when the loss stopped dropping at epoch 15, we ended the training. To verify the quality of our trained autoencoder, we performed a comparison between the reconstructed and the original particles. We chose the particle patch centers every 0.33 units in all three spatial dimensions for reconstruction, while the particle patch radius was set as 0.5. The distance between the particle patch centers was chosen to be smaller than the radius to remove the seam artifacts. Overlapping particles' reconstructed value was averaged between the patches. We might observe that reconstructed values of concentration have been smoother amongst reconstructed particles. These high frequency concentration variations were discovered to be noise on this dataset and needed to be eliminated prior to the process of feature extraction. Due to the smoothing impact of the reconstruction, our following feature extraction benefits from it. Between the reconstruction and the original data, the peak signal-to-noise ratio (PSNR) is 26.9. Given that only latent vector clusters are used in order to extract features and do not use reconstructed data to replace original, PSNR values in this range have been suitable for achieving our objective.

4.2. Feature extraction results

As a starting point for tracking and our feature extraction example, we initially pick time step 25, when, in accordance with earlier work on this data set, the number of the viscous fingers begins rising and their structure begins to get more complex. The same process described earlier is used to select particle patches, and for each patch, a latent vector is produced. After we got the coarse latent representations, k-means algorithm was applied to assign a cluster id to each patch. We use t-SNE projection as an assisting tool to help this cluster analysis. This projection provides us with a clearer view of these clusters. We can see that even though brown and yellow clusters are separated in k-means, they are close to each other in t-SNE projection and, indeed, they are all background blocks in the dataset. This also happens for green and red clusters, where they are both boundary patches around the fingers. These effects indicate that we may have chosen a cluster number that is too high for the k-means algorithm. Users can circle in the projected space and visualize in 3D to further verify the cluster's feature. In this example, the blue cluster is exactly corresponding to the viscous fingers in the dataset. After identifying the clusters of interest, we inferred the latent vector for a particle patch centered at every particle and calculated its cosine distance to the interested cluster center to get a high-resolution extraction result. Particle patches whose latent distance to the cluster of interest is lower than a threshold is chosen to be the extracted features.

In training the autoencoder to generate latent vectors, we need to choose two hyperparameters, the size of neighborhood r and latent vector dimension v. The different choices are tested through the experiments. Firstly, we tested the influence of r on the extraction results. The experiment shows the particles of 4 extraction results from the same time step with r=0.1, 0.3, 0.5, 0.8 respectively. Small r values be seen as fragmented fingers when r=0.1 and r=0.3. Meanwhile, will generally introduce more noises to the extraction result, which can larger r will make small fingers shrink or even disappear, where an example can be seen in the purple squares. We attribute this effect to that larger neighborhood will include the finger boundary in the feature on these observations, we chose r=0.5 in our following experiments. Particle patches, thus small fingers will be smaller or eliminated. Based In the next step, we test how v, the dimensionality of the latent vectors, should be chosen. Generally speaking, v is the bottleneck of the autoencoder, thus controls the network's learning capacity. Hereby, we introduce the intrinsic dimension, which describes how many di- mensions are needed to generate a good approximation of the data set. In order for the latent vector to be a good particle representation, v should be set larger than the intrinsic dimension of the data set. It can be observed that when v=16, there are 3 dimensions that have little variance, and when v=32, there are 16 dimensions that have little variance, which makes these dimensions trinsic dimension of this data set is less than 16. Setting r=0.5, v=16, redundant in describing the input rendering and surface extraction. Through the processing of particle patches that have been defined on regular grid, assigning the value of 1 to patches with finger cores, -1 to patches with other clusters, and extracting iso-surface at the value of 0 with the use of marching cubes, surface extraction is accomplished.

4.3. Compare with other ways to represent particle patches

There are several ways to represent a particle patch besides using an autoencoder to generate feature descriptors. In this part, we contrast our latent representation with two more approaches: neighborhood mean and principal component representations of the patch. Knowing that the viscous fingers only appear where there is a fairly high concentration of salt is one of the prior knowledge about them. Filtering the particles with a concentration threshold is an easy technique to extract fingers. In order to filter out high frequency noises and extract features with threshold, the first approach that is compared involves the averaging of concentration in neighborhood. If we just exclude noises with averaged concentration in particle patches, it exemplifies how challenging it is to calculate the threshold. High thresholds will cause certain fingers to disappear, whereas low thresholds will make it more difficult to differentiate between various fingers. The benefit of our NN approach is that it automatically captures gradient-defined properties. It is likely true that we can calculate concentration gradients to detect features, and earlier research has used gradients for identifying viscous fingers. Yet, the authors use conventional grid data, and it could be expensive and difficult to use mesh-free approaches to calculate gradients on particle data. Another approach is to run the particle patches through a PCA and represent each one of the particle patches using its principal components, we can see the cluster of principle components only vaguely show the boundary between high and low concentration areas. This is due to i) principal components are influenced by the positions of particles in the patch, so that patches with different particle distributing patterns will fall into different clusters and ii) PCA is only a linear transformation of the neighborhood and does not consider any global information between the particle patches.

4.4. Compare with topology analysis

The method we contrast in this section was suggested by the winners of the 2016 SciVis challenge, who put forward two methods for extracting finger structures [6]. Slice-wise clustering is used in the second, while the first one has been based upon topology analysis of data. On same data from time step 25 of a single simulation from the ensemble, we applied both our approach and the topology-based approach. In general, the two approaches produce about the same number of fingers: 24 for the topology analysis and 25 for the proposed approach. We colored fingers for matching same fingers in the two approaches. The two outcomes still clearly differ from one another, though. One finding from the outcome is that the analysis of the topology produces larger-sized fingers. It is mostly due to the fact that analysis has been based upon smooth distance field that will connect boundaries between the fingers. Second, our approach and topological analysis divide fingers in space in different ways. Due to the fact that our approach for separating them uses DBSCAN, connected fingers with same density of particles will be considered to be a single finger. The topological analysis, at the same time, simply employs persistence thresholds for finding fingers, which will miss some and merge the matching region to others, as what is seen in the blue squares.

Quantitatively, we compare the number of fingers extracted by our method with the method in [6] across all time. It can be seen that our method has similar finger count trend with their two methods. In the case where data turn chaotic in later time steps, finger numbers both rise during the first 30-time steps and then fall. The heuristic clustering result in their work and the finger count in this paper are similar. Their findings explains that the selection of the threshold is the primary source of the number difference. Generally, for this SciVis contest dataset, our approach achieves equivalent feature extraction outcomes to the winning method, both quantitatively and qualitatively.

4.5. Tracking results

As mentioned in the method section, we need to choose two parameters for feature tracking. The first one is the radius of Epanechnikov kernel and the second is the number of bins b in each dimension for histogram building. We did experiments on different radius and find that the choice of kernel size should be based on the actual feature size. As for instability in the tracking. Setting b=2 will give us 16 histogram bins b, we found any number larger than 2 in each dimension will cause in total, since we have four latent dimensions after PCA projection. The experiment results could be found in the supplementary materials. The termination criteria for the Cam-shift algorithm are when the selection window's shift distance is less than 0.0001, or after the algorithm reaches 100 iterations. In experiment we show the number of iterations before the algorithm ends for the 4 tracked cases. All tracking examples terminate before reaching the iteration limit, we set showing that the algorithm converges well.

We select 2 fingers from time step 25 in the same run from the ensemble simulation to demonstrate our tracking method. We show the tracking result every 5-time steps and the animation can be found in the supplementary materials. The first tracking does not go off the feature through 20-time steps. The second tracking sticks to the feature before the feature starts to deform and finally disappear beginning at the time step 35. Generally speaking, tracking small features in the dataset is more challenging because our method relies on the overlap between time steps.

4.6. Time and memory consumption

Under the hyperparameters chosen as discussed, our training of autoencoder described took 10.5 minutes per epoch, which made the total training time of 15 epochs to be 2.7 hours. Once the autoencoder is trained for this data set, inference of latent vectors takes 75 seconds per million particles on the same machine. Since one time step con- sists of about 190,000 particles. It takes approximately 15 seconds to inference all particles in one time step and produce high-resolution extraction result in the experiment. Considering I/O, data-preparation and clustering, the total feature extraction time for one time step in this data set is around 20 seconds. The time that it takes to track a feature depends on its size. More latent vectors will be generated as feature size increases. The time of the latent vector generation accounts for majority of the total tracking time. The tracking time is roughly 90 ms when the feature of interest is fingertip that consists of 200–400 particles. This is much more efficient than the method where features are first extracted and then matched in 2-time steps.

Our technique processes the entire dataset patch by patch, which reduces memory usage. The memory need for latent generation can be almost neglected. The most memory consuming part of our approach is the t-SNE projection and clustering. However, we can increase the distance between particle patch centers to reduce the number of patches in this step to alleviate this problem. Once the interesting cluster is found we can always produce high-resolution extraction result afterwards.

5. CONCLUSION

On the basis of the latent vectors produced by an autoencoder, we offer a particle feature extraction and tracking technique. Our approach avoids grid re-sampling, which could be costly and lossy, by processing particle data directly. We use two separate particle datasets to show the results of effective feature extraction. We demonstrate in the first dataset that our latent vector could result in the reconstruction of the particle patches with lower input noise. Results of the finger structure extraction are better than those of the concentration threshold approach or principal component representation of the block. We contrast our extraction with the approach used by the SciVis contest winner, and similar outcomes are displayed. Our tracking approach was verified by manually picking finger structure tips and follow the tracking result temporally, where the method worked well as long as feature regions have overlap between time steps. In the second dataset, our approach is adapted to a larger particle data with multi-dimensional physical attributes. Still, interesting features are captured by the latent representation. Even though the particle segmentation is not completely the same as the original halo finding algorithm, our method provides an efficient way to extract related regions for tracking and further analysis. Tracking is quantitatively analysis in this dataset using the ground truth halo positions. Our tracking results are stable for most halos in this dataset.

One drawback of our approach is our feature tracking algorithm only considers the local similarity across time steps for efficiency reason. This may potentially lead to the problem that our tracking result is not globally optimal and we cannot identify feature changing events like split or merge along time steps. Use neural networks to solve this temporal global optimization problem is another possible way. This includes modifying the neural network to recurrently take inputs from multiple time steps and generate latent vectors for spatial-temporal features, which is a future work of our study.

REFERENCES

[1] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing features and tracking their evolution computer," *IEEE, Translations and Content Mining are Permitted for Academic Research Only,* vol. 27, no. 7, pp. 20–27, 1994, doi: 10.1109/2. 299407.

[2] J. Lukasczyk *et al.*, "Viscous fingering: a topo- logical visual analytic approach. applied mechanics and materials." *Trans Tech Publications Ltd, Switzerland,* vol. 869, pp. 9–19, 2017, doi: 10.4028/www.scientific.net/amm.869.9.

[3] 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,* pp. 375–392, 2017, doi: 10.1007/978-3-319-61358-1.

[4] G. Favelier, C. Gueunet, and J. Tierny, "Visualizing ensembles of viscous fingers." In *IEEE Scientific Visualization Contest,* 2016.

[5] F. Sauer, H. Yu, and K. L. Ma, "Trajectory-based flow feature tracking in joint particle/volume datasets," *IEEE Transactions on Visualization and Computer Graphics,* vol. 20, no. 12, pp. 2565-2574, 2014, doi: 10.1109/TVCG.2014. 2346423.

[6] G. Reina, S. Gumhold, and T. Ertl "Visual and structural analysis of point-based simulation ensembles," *IEEE Computer Graphics and Applications,* vol. 38, no. 3, pp. 106–117, 2018, doi: 10.1109/MCG.2017.3301120.

[7] H. C. Cheng *et al.*, "Deep-learning-assisted volume visualization," *IEEE Transactions on Visualization and Computer Graphics,* vol. 25, no. 2, pp. 1378–1391, 2019, doi: 10.1109/TVCG.2018.2796085.

[8] T. Luciani, A. Burks, C. Sugiyama, J. Komperda, and G. E. Marai, "Details- first, show context, overview last: Supporting exploration of viscous fingers in large-scale ensemble simulations," *IEEE Transactions on Vi- sualization and Computer Graphics,* vol. 25, no. 1, pp. 1225–1235, 2019, doi: 10. 1109/TVCG.2018.2864849.

[9] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using geo-CNN," *arXiv,* pp. 998–1008, 2018, doi: 10.1109/CVPR.2019.00109.

[10] T. Christoudias, C. Kallidonis, L. Koutsantonis, C. Lemesios, L. Markou, "Sophocleous. Visualising the dark sky," *IEEE Scientific Visualization Conference, SciVis 2015 - Proceedings,* pp. 79–86, 2016, doi: 10.1109/SciVis.2015.7429496.

[11] T. Rapp, C. Peters, and C. Dachsbacher, "Visual analysis of large mul- tivariate scattered data using clustering and probabilistic summaries," *IEEE Transactions on Visualization and Computer Graphics,* pp. 1–1, 2020, doi: 10.1109/tvcg.2020.3030379.

[12] J. Han, H. Zheng, Y. Xing, D. Z. Chen, C. Wang, and S. Member, "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, 2021, doi: 10.1109/TVCG.2020.3030346.

[13] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *Proceedings of the IEEE International Conference on Com- puter Vision,* 2019, pp. 8159–8170, doi: 10.1109/ICCV.2019. 00825.

[14] 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, doi: 10.1145/3326362.

[15] J. Chen, D. Silver, and L. Jiang, "The feature tree: visualizing feature tracking in distributed AMR datasets," in *Proceedings - IEEE Symposium on Parallel and Large-Data Visualization and Graphics,* 2003, pp. 103–110, doi: 10.1109/PVGS.2003.1249048.

[16] S. Dutta and H. W. Shen, "Distribution driven extraction and tracking of features for time-varying data analysis," *IEEE Transactions on Visualiza- tion and Computer Graphics,* vol. 22, no. 1, pp. 837–846, 2016, doi: 10.1109/TVCG. 2015.2467436.

[17] G. Ji and H. Shen, "Feature tracking using earth mover's distance and global optimization," in *PG'06: Proceedings of the Pacific Graphics 2006,* 2006.

[18] C. Muelder and K. L. Ma, "Interactive feature extraction and tracking by utilizing region coherency," *IEEE Pacific Visualization Symposium, Paci- ficVis 2009 - Proceedings,* 2009, pp. 17–24, doi: 10.1109/PACIFICVIS. 2009.4906833.

[19] D. Silver and X. Wang, "Tracking scalar features in unstructured datasets," *Proceedings of the IEEE Visualization Conference,* vol. 98, 1998, pp. 79–86, doi: 10.1109/visual.1998.745288.

[20] J. Xu, S. Dutta, W. He, J. Moortgat, and H.-W. Shen, "Geometry-driven detection, tracking and visual analysis of viscous and gravitationalfingers," *IEEE Transactions on Visualization and Computer Graphics,* vol. 28, pp. 1514-1528, 2019, doi: 10.1109/TVCG.2020.3017568.

[21] G. Ji, H. W. Shen, and R. Wenger, "Volume tracking using higher dimen- sional isosurfacing," in *Proceedings of the IEEE Visualization Conference,* pp. 209–216, 2003, doi: 10.1109/visual.2003.1250374.

[22] 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, doi: 10.1109/TVCG.2017. 2674918.

[23] G. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci, "Feature track- ing using reeb graphs," *Topological methods in data analysis and visualization,* pp. 241–253, 2011, doi: 10.1007/978-3-642-15014-2-20.

[24] F. Yet. T. Zeng, and K. Ma, "Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations," in *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing,* 2005, pp. 6–6, doi: 10.1109/SC.2005.37.

[25] L. Linsen, T. V. 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, doi: 10.1109/TVCG.2008.167.

[26] 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,* pp. 1–27, 2019, doi: 10. 1109/TPAMI.2020.3005434.

[27] J. Wang, S. Lan, M. Gao, and L. Davis, "InfoFocus: 3D object detection for autonomous driving with dynamic information modeling," 2020, *arXiv:2007.08556v1,* doi: 10.1007/978-3-030-58607-2_24.

[28] P. Hermosilla, T. Ritschel, P. P. Va´zquez, A. Vinacua, and T. Ropinski, "Monte Carlo convolution for learning on non-uniformly sampled point clouds," *ACM Transactions on Graphics (TOG),* vol. 37, no. 6, 2018, doi: 10.1145/3272127.3275110.

[29] 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," *Advances in Neural Information Processing Systems Conference,* pp. 820–830, 2018.

[30] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: flexible and deformable convolution for point clouds," *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, doi: 10.1109/ICCV.2019.00651.
[31] W. Wu, Z. Qi, and L. Fuxin, "PointCONV: Deep convolutional networks on 3D point clouds," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9613–9622, doi: 10.1109/CVPR.2019.00985.
[32] 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, doi: 10.1109/TVCG.2019. 2934332.
[33] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using Cam-shift," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 7, pp. 142–149, 2000, doi: 10.1109/CVPR.2000.854761.
[34] A. Asvadi, P. Gira˜o, P. Peixoto, and U. Nunes, "3d object tracking using rgb and lidar data," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1255–1260, doi: 10.1109/ITSC.2016.7795718.
[35] C. Hermes, J. Einhaus, M. Hahn, C. Wohler, and F. Kummert, "Vehicle tracking and motion prediction in complex urban scenarios," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 26–33, doi: 10.1109/IVS.2010.5548014.
[36] J. Lukasczyk, G. Weber, R. Maciejewski, C. Garth, and H. Leitte, "Nested tracking graphs," *Computer Graphics Forum*, vol. 36, no. 3, pp. 12–22, 2017, doi: 10. 1111/cgf.13164.
[37] E. Watan, S. Ali, and L. Mohammad, "Proposed image forgery detection method using important features matching technique," *Journal of Engineering Science and Technology*, vol. 16, no. 2, pp. 1525-1538, 2021.
[38] M. Salam, H. Ayad, and I. Watan, "Lung segmentation using proposed deep learning architecture," *Article in International Journal of Online and Biomedical Engineering (iJOE)*, Dec. 2020, doi: 10.3991/ijoe.v16i15.17115.

BIOGRAPHIES OF AUTHORS

Ekhlas Watan Ghindawi is associate professor at college of education, department of computer science, Al-Mustansiriyah University, Iraq. She got a Ph.D. degree in computer science with specialization in computer vision. Her research areas are image/signal processing, biometrics, medical image analysis and pattern recognition. Her research interests include image/signal processing, biometrics, medical image and analysis, and pattern recognition. She can be contacted at email: ikhlas_watan@uomustansiriyah.edu.iq.

Sally Ali Abdulateef is associate professor at college of education, department of computer science, Al-Mustansiriyah University, Iraq. She got the M.Sc. degree in computing from the University of Baghdad. Her research interests include soft computing, machine learning, and intelligent systems. She can be contacted at email: samh_forever2000@uomustansiriyah.edu.iq.