Direct Neighbor Search for Curve-based Vector Field Processing

NSF IIS 1553329
NSF OAC 2102761

Nguyen K Phan
University of Houston

Guoning Chen
University of Houston

Problem Statement

- Vector fields and their analysis are crucial for many applications involving various dynamical systems. To reduce the storage cost of large-scale simulated vector field data, curve-based representation can be needed.
- However, curve-based representation poses challenges to the subsequent analysis tasks, because (1) they do not cover the entire domain and (2) they have varying densities in space. This uneven distribution of integral curves may affect tasks that prefer a near-uniform spatial distribution of neighboring curve segments. Figure 1 shows the impact of the uneven distribution of neighbors to a vector field reconstruction task, indicating the need of evenly distributed neighbors.
- In addition, some tasks may require direct neighbors (there should be no other neighbors between a selected neighbor and the query point).
- Existing neighbor search strategies, e.g., KNN and RNN, cannot satisfy the above requirements.

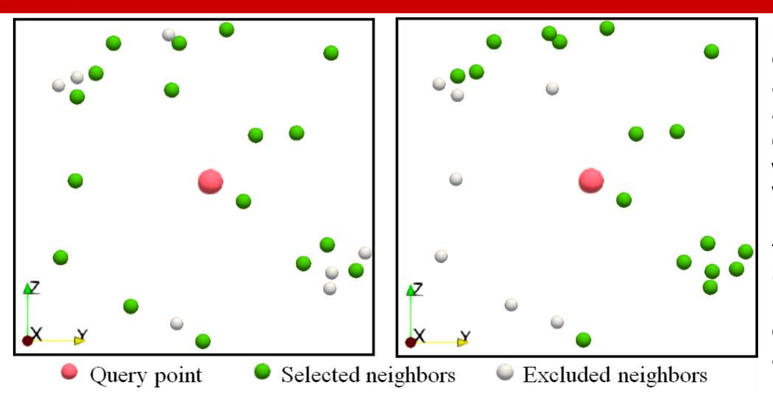

Figure 1: Illustration of a scenario of neighbors (green dots) around a query point (red dot) that have a relatively uniform spatial distribution (left) and a scenario whose neighbors do not (right). We use these neighbors to reconstruct the vector value at the query point. The difference (or error) between the reconstructed vector to the original vector is 15.79 for the left and 21.05 for the right.

Contributions

- We introduce a direct neighbor search for curve-based vector field processing and propose a method to achieve it.
- We applied our new search strategy to support two tasks on a few 3D curve-based datasets to demonstrate its effectiveness.

Proposed Methodology

We partition a curve (e.g., a streamline) into a set of segments based on the individual integration points. By default, two consecutive points form a segment. After this decomposition, we consider two different scenarios of neighbor search below.

- Point-Center Direct Neighbor Search(PCN)

We search the neighboring segments of a given **query point *p***. To identify uniformly distributed neighbors, we construct a sphere, S_p centered at the query point ***p*** with a radius ***r***. Each point on S_p can be expressed as $(x, y, z) = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$ $(\theta \in [0, 2\pi),\ \phi \in [0, \pi])$ (Figure 2a). We then partition $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ into ***h*** and ***v*** sub-ranges uniformly, creating ***h*v*** sub-volumes within S_p. These sub-volumes, also referred to as **search sections**, form the **Point-center Segment Neighborhood** (PSN) (Figure 2b). The curve segments falling within S_p (referred to as the **candidate segments**) will intersect with one or more of these search sections. Each candidate segment is assigned to a search section $S_{i,j}$ $i \in \{1, .., h\}, j \in \{1, \dots, v\}$ if it has at least one point falling in $S_{i,j}$. Within each $S_{i,j}$, we identify the segment that is the closest to ***p*** based on their Euclidean distance as a direct neighbor. To counter the same segment crossing multiple search sections and the multiple candidates within a search section having a similar distance to ***p,*** we allow up to ***K*** nearest neighbors in each section. By default, *K=1* (Figure 2c).

- Curve-Center Direct Neighbor Search(CCN)

We search the neighboring segments of a given segment ***c***. To achieve that, we construct a 3D domain, S_c that consists of a cylinder ***T*** with ***c*** as its axis and with radius ***r*** and two hemispheres H_b and H_t with radius ***r*** attaching the two base faces of ***T*** (Figure 2d). We refer to this region as a **Curve-center segment neighborhood (CSN).** S_c encloses completely or partially a few (candidate) curve segments, from which the direct neighbors to ***c*** will be identified. To account for the uneven distribution of the segments surrounding ***c***, we partition the cylinder ***T*** into ***2m*** search sections uniformly using ***m*** cut planes across the cylinder axis. Two adjacent cut planes form a search section. These search sections are illustrated in Figure 2e, which shows their configuration in the x'y' plane of the cylinder ***T***. From this, the direct neighbor of each search section is identified as the segment with the smallest distance to ***c***. For the two hemispheres of S_c , we simply identify up to ***K*** nearest segments to ***c.***

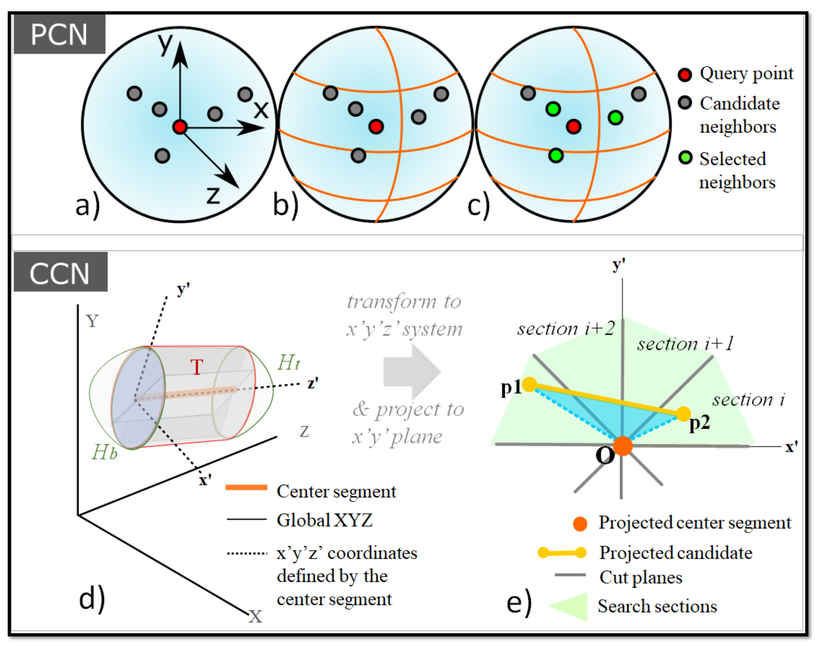

Figure 2: Illustration of PSN (a-c) and CSN (d-e) construction.

Results and Applications

- Saliency Measurement

To aid the exploration of the curve-based data, we adapt the saliency metric for a point [1] to a curve segment **c** as below

$$s_{\mathbf{c}} = 1 - \exp(-\frac{1}{n}\sum_{i} d_{sim}(\mathbf{c}, \mathbf{g}_i))$$

where **n** is the number of neighboring segments around **c** and $d_{sim}(c, g_i)$ measures the orientation difference between ***c*** and one of its neighbors g_i.

If all neighbors have a similar orientation to ***c***, $s_{\mathbf{c}}$ is close to 0; otherwise, $s_{\mathbf{c}}$ will be large. In other words, $s_{\mathbf{c}}$ puts emphasis to segments whose neighbors behave differently from them, which is usually seen in features such as vortex cores and flow separation. Figure 3 compares the visualization of the obtained $s_{\mathbf{c}}$ using neighbors identified by KNN, RNN and our CSN. We see that $s_{\mathbf{c}}$ computed with the neighbors of our CSN correctly highlights places with vortical behavior, while the others either miss important features or occlude important structure.

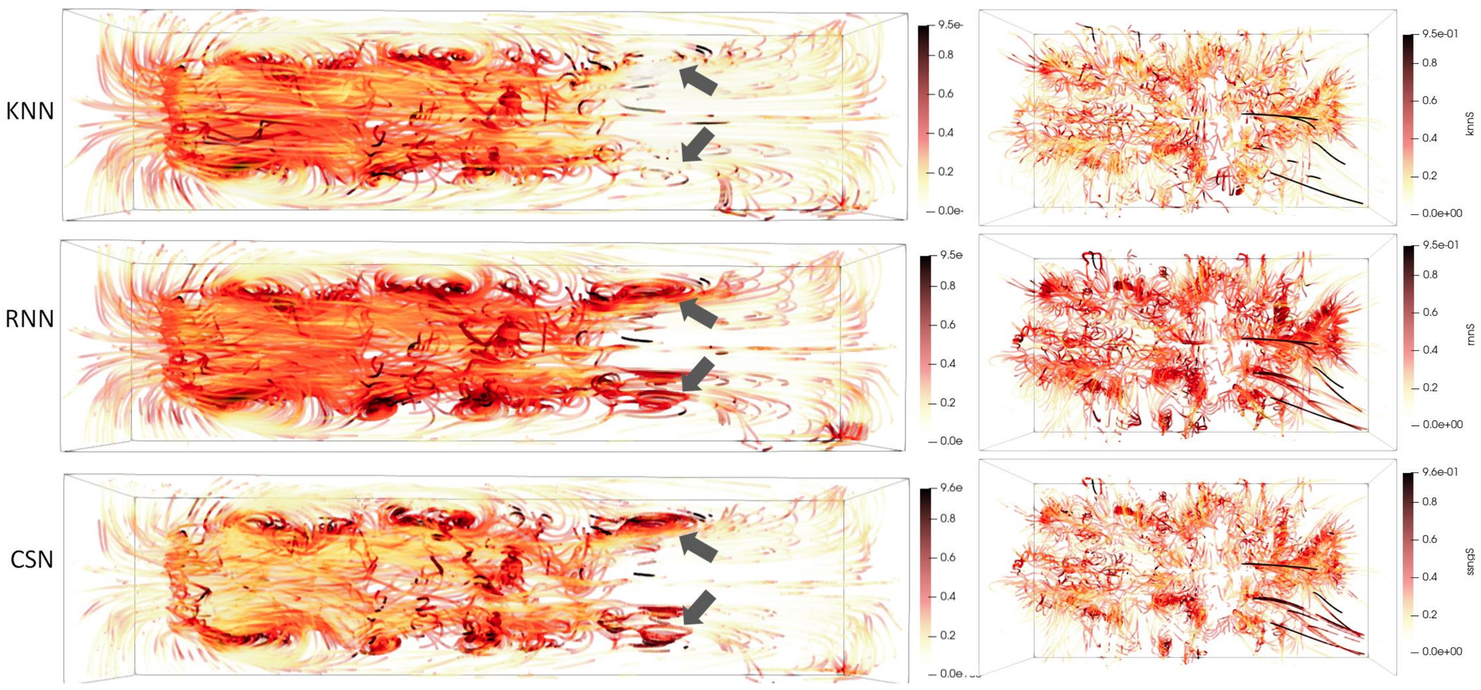

Figure 3: Feature emphasis based on the saliency of the individual segments for the Solar Plume dataset (left) and crayfish(right). For KNN, K = 10; for RNN,r = 1.5%; for our CSN, r = 1.5%, m = 12, and K = 1. Our CSN (middle) better highlights places with vortical behaviors.

- Vector Field Reconstruction

We apply KNN, RNN, and our PSN, to identify neighboring segments of each grid point to reconstruct vector fields from four streamline datasets [2], respectively. We use $e = \sqrt{MAE_{vx}^2 + MAE_{vy}^2 + MAE_{vz}^2}$ that combines the mean average errors (MAE) of the x, y, and z components of the reconstructed velocity vectors at the individual grid points to quantify the reconstruction errors. From this comparison, we see that our PSN outperforms the KNN and RNN in terms of the reconstruction error. KNN is the fastest approach, while our method has comparable computation time to KNN in some cases. Both our method and KNN are faster than RNN. This is because both KNN and PSN usually find a much smaller number of neighboring segments for each grid point than RNN. PSN is slower than KNN due to the search region construction and distance calculation.

Dataset	RNN		KNN		PSN	
Cylinder	0.101483	26s	0.101036	**11s**	**0.092892**	12s
Bernard	0.420877	**2s**	0.417736	4s	**0.417552**	**2s**
Crayfish	0.039921	88s	0.039363	**16s**	**0.038615**	51s
Plume	0.410805	109s	0.410167	**20s**	**0.40148**	33s

Table1: The absolute MAE error values of the reconstructed vector fields using RNN, KNN, and PSN, respectively.

Figure 4: Comparison of reconstructed errors with RNN, KNN, and our PSN on the four datasets. To facilitate comparison, we normalize the errors

Conclusion and Future Works

- We address the non-uniform spatial coverage problem often seen in the existing neighbor search approaches (e.g., KNN) by proposing a direct neighbor search for curve-based representation that focuses on uniform spatial distribution of neighboring segments. We demonstrate their effectiveness through the visualization of the saliency metric and vector field reconstruction.
- However, our definition of direct neighbors is rather heuristic and not precise. In addition, our search region construction may not handle some extreme configurations (e.g., two nearby streamlines that are orthogonal to each other while one has sharp changes at places where they are the closest to each other). We plan to address these limitations and extend the direct neighbor search for pathlines.

Reference

[1] Y. Lu, L. Cheng, T. Isenberg, C.-W. Fu, G. Chen, H. Liu, O. Deussen, and Y. Wang. Curve complexity heuristic kd-trees for neighborhood- based exploration of 3d curves. In *Computer Graphics Forum*, vol. 40, pp. 461–474. Wiley Online Library, 2021.
[2] L. Shi, R. Laramee, and G. Chen. Integral curve clustering and simplification for flow visualization: A comparative evaluation. *IEEE transactions on visualization and computer graphics*, 27(3):1967 – 1985, 2021.

UNIVERSITY of HOUSTON