Direct Neighbor Search for Curve-based Vector Field Processing

Nguyen K Phan*
University of Houston

Guoning Chen†
University of Houston

Abstract

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.

Keywords: curve-based vector field, neighbor search

1 Introduction and Background

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, intermediate representations are usually needed. As the foundation for many other representations, the curve-based representation via sets of integral curves is usually adopted. However, curve-based representation poses challenges to the subsequent analysis tasks. This is because the integral curves do not cover the entire domain and may have varying densities due to the non-homogeneous characteristics of general flows. This uneven distribution of integral curves may affect tasks (e.g., vector field reconstruction and the definition of a differential operator for curve segments) that prefer a near-uniform spatial distribution of neighboring curve segments.

In the example shown in Fig. 1, two sets of neighbor segments (represented by their respective middle points) are selected from a candidate set of neighbors found within a sphere centered at a query point (red dot) from a streamline data set. The selected neighbors are colored in green, while the excluded ones are in gray. We see that the selected neighbors in the left case have a more uniform spatial distribution than those in the right (which are clustered in the upper right region). With these two sets of neighbors, we use a simple inverse distance-based interpolation strategy to reconstruct the velocity vector at the query point, respectively. Our results show that the error for the left case when compared with the ground truth vector at the query point, is 15.79, while for the right one it is 21.05. This indicates the need for a neighbor search strategy that can return neighboring segments around a query point (or a query curve) that have an as-uniform-as-possible distribution.

In addition to spatial uniformity, the selected neighbors should be *direct*, that is, there should be no other neighbors between a selected neighbor and the query point, as the non-direct neighbors represent redundant information given the spatial coherency of vector fields. Unfortunately, current standard neighbor search strategies, including k-nearest neighbors (KNN) and the radius-based nearest-neighbors (RNN) [1], do not take into account the uneven spatial distribution of the input curves, nor do they return neighbors directly adjacent to the query point (or curve).

To address this challenge, we introduce a direct neighbor search framework for the individual curve segments. Our framework makes use of some spatial partitioning strategies to select an optimal set of direct neighbor segments for a given curve segment or a query point that also maintains uniform spatial coverage. We apply our framework to support two tasks on a number of integral curve datasets [2] to demonstrate its effectiveness.

*e-mail: nguyenpkk95@gmail.com
†e-mail: gchen22@central.uh.edu

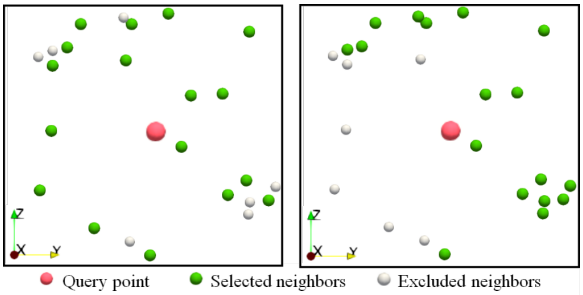

Figure 1: Illustration of a scenario of neighbors (green dots) around a query point (red dot) that have a relatively uniform spatial distribution (left) and a scenario whose neighbors do not (right).

2 Method

3D Curve Decomposition: We first partition each 3D curve (e.g., a streamline) into a set of curve segments. Given a 3D curve with N line segments, we consider every L line segment forming a curve segment. By default, $L = 1$. With this decomposition, we next describe the direct neighbor search for two situations (1) *search neighbor segments around a point* and (2) *search neighbors around a curve segment*.

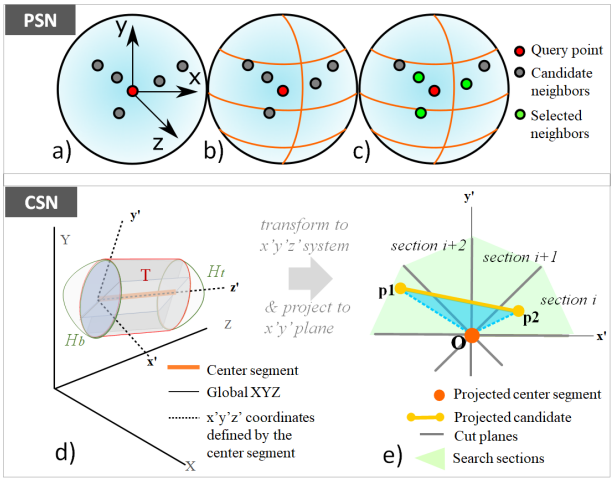

Figure 2: Illustration of PSN (a-c) and CSN (d-e) construction.

Point-Center Direct Neighbor Search: There are curve-based data processing and exploration tasks that require to find the neighboring segments of a (query) point, such as the structure-aware line selection and opacity specification [1]. To search for direct neighboring segments with a more uniform spatial distribution, we construct a sphere $\mathbf{S_p}$ centered at the query point $\mathbf{p}$ with a radius r. Each point on the boundary of $\mathbf{S_p}$ can be expressed as $(x,y,z) = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$ $(\theta \in [0,2\pi),\ \phi \in [0,\pi])$ (Fig. 2a). We then partition θ and ϕ into h and v sub-ranges uniformly, creating $h * v$ sub-volumes within $\mathbf{S_p}$. These sub-volumes, also referred to as *search sections*, form the *Point-center Segment Neighborhood* (PSN) (Figure 2b). The curve segments falling within $\mathbf{S_p}$ (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 $S_{i,j}$ $(i \in \{1,..,h\}, j \in \{1,...,v\})$ if it has at least

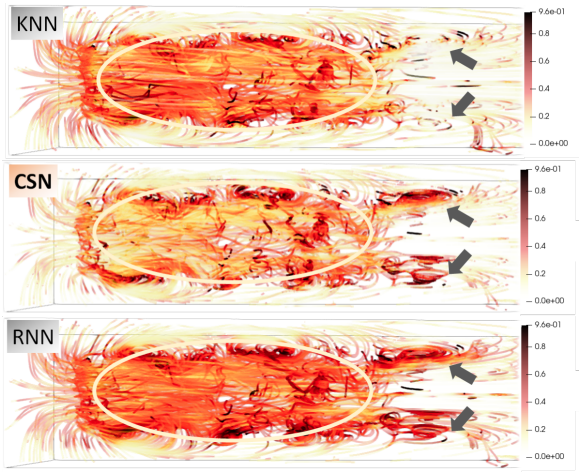

Figure 3: Feature emphasis based on the saliency of the individual segments for the Solar Plume dataset.For KNN, K = 10; for RNN, r = 1.5%; for our CSN, r = 1.5%, m = 12, and K = 1. Our CSN (middle) better highlights places with vortical behaviors.

one point falling in $S_{i,j}$. Within each $S_{i,j}$, we identify the segment that is the closest to $\mathbf{p}$ 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 $\mathbf{p}$, we allow up to K nearest neighbors in each section. By default, $K = 1$.

Curve-Center Direct Neighbor Search: For tasks that need to compute the dissimilarity of neighbor curves, identifying the neighboring segments of a given center segment $\mathbf{c}$ is required. To locate direct neighbors around $\mathbf{c}$ with near-uniform spatial coverage, we construct a 3D domain $\mathbf{S_c}$ that consists of a cylinder T with $\mathbf{c}$ as its axis and with radius r and two hemispheres (or caps) H_b and H_t with radius r attaching the two base faces of the cylinder (Fig. 2d). We refer to this region as a *Curve-center segment neighborhood (CSN)*. $\mathbf{S_c}$ encloses completely or partially a few (candidate) curve segments, from which the direct neighbors to $\mathbf{c}$ will be identified. To account for the uneven distribution of the segments surrounding $\mathbf{c}$, 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 Fig. 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 $\mathbf{c}$. For the two hemispheres of $\mathbf{S_c}$, we simply identify up to K nearest segments to $\mathbf{c}$.

3 Results and Discussion

We apply our direct neighbor search strategy to two tasks to assess its effectiveness.

Saliency measurement: To aid the exploration of the curve-based data, we adapt the saliency metric for a point from [1] to a curve segment $\mathbf{c}$, which we define below

$$s_{\mathbf{c}} = 1 - \exp\left(-\frac{1}{n}\sum_i d_{sim}(\mathbf{c}, \mathbf{g}_i)\right) \quad (1)$$

where n is the number of neighboring segments around $\mathbf{c}$ and $d_{sim}(\mathbf{c}, \mathbf{g}_i)$ measures the orientation difference between $\mathbf{c}$ and one of its neighbors $\mathbf{g}_i$. For simplicity, $d_{sim}(\mathbf{c}, \mathbf{g}_i) = \frac{\vec{\mathbf{c}}}{\|\vec{\mathbf{c}}\|} \cdot \frac{\vec{\mathbf{g}}_i}{\|\vec{\mathbf{g}}_i\|}$.

With this saliency measure, if all neighbors have a similar orientation to $\mathbf{c}$, $s_{\mathbf{c}}$ 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. Note that $s_{\mathbf{c}}$ is not to be used for the automatic detection of these features; we are utilizing $s_{\mathbf{c}}$ to visually reveal potential areas in the vector field that may contain these features.

Figure 3 shows that the streamlines that have high saliency values (in the circle) based on both KNN and RNN are less interesting. These streamlines also occlude important inner features (see the RNN result). This is because RNN considers the more neighbors for each segment than needed, which leads to the highest concentration of saliency color value and emphasizes the largest portions of the vector field (highlighting more than necessary). The result with KNN in general put less emphasis on the locations with features than the results of RNN and CSN. KNN also misses two vortices (pointed out by arrows). In contrast, the saliency computed using our CSN properly highlights places with strong rotational behavior, while de-emphasizing other places with little interesting behavior.

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] computed from (1) the simulation of flow behind a cuboid cylinder, (2) Bernard, (3) Solar Plume, and (4) Crayfish, respectively. We performed parameter optimization and selected the respective parameters that minimize the error for each neighbor search strategy. To compare the performance of the three neighbor search strategies, 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.

Table 1: The absolute MAE error values of the reconstructed vector fields using RNN, KNN, and PSN, respectively.

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

Table 1 provides some statistics of the vector field reconstruction results with the three neighbor search strategies. 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.

4 Conclusion and Future Work

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.

Acknowledgments

This research was partially supported by NSF IIS 1553329 and OAC 2102761.

References

[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.