Progress report: Improving Trajectory Completion through Junction Extraction

Yang Li

February 2, 2015

Collaborated work with Yangyan Li.

1 Overview

This report summarizes my continued work on the problem of trajectory completion: recovering dense vehicle trajectories from a collection of sparse trajectories in an urban environment. Previously, we have developed a point-based algorithm which works well on simple road structure. However, the completion accuracy is not desirable at complicated junctions, due to interference between sample points that belong to trajectories traveling in different directions.

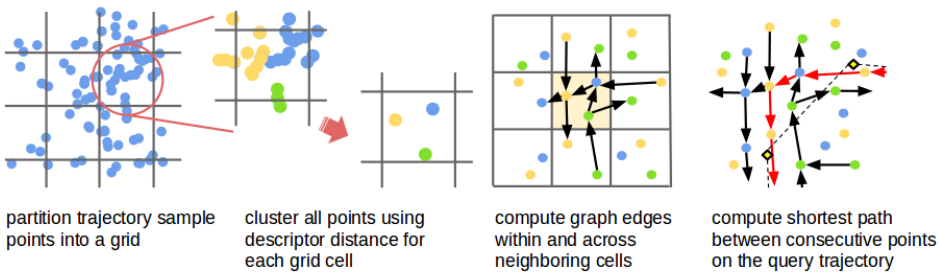


Figure 1: Main steps in the original trajectory completion algorithm

We observed that although the internal structure of highway junctions has many variations, the vehicle's path can be determined by the order and direction it enters and exits the junction. Our problem reduces to identifying junctions and the configuration of their main branches from trajectory sample points. This will enable us to accurately distinguish sample points that belong to different paths at the junction, leading to better trajectory completion result.

In the remaining report, we will show one approach to compute junction location and connectivity using skeleton extraction.

2 Junction graph extraction

In this problem, we define a trajectory of length n as a sequence of 2D points $p_1, ..., p_n$ recorded at fixed time intervals. We are also given the vehicle's heading direction $\hat{d}_1, ..., \hat{d}_n$ measured at the same time intervals.

The junction graph extraction problem is defined as follows: Given a collection of input trajectories T and the associated heading information, find the embedding of an undirected graph over all junctions (intersections) traversed by the trajectories. The junction graph can be considered as an abstraction of the underlying road network.

To solve this problem, we adapt the L1 medial skeleton extraction algorithm proposed in [1] to GPS trajectory data. The basic steps in our adaptation are the following:

1. Upsampling sparse trajectories to create the voting image,

2. Iteratively extract skeleton branches from points sampled on the voting image

3. Construct the junction graph from skeleton branches and smooth the graph.

2.1 Upsampling sparse trajectories

Despite of having a large collection of trajectories that cover most of the roads in the given region, we have no guarantee that the input points are well distributed. To solve this problem, we upsample the sparse trajectories using a simple interpolation scheme.

Let p_i be the ith point in sparse trajectory t, and let unit vector $\hat{d}_i$ be its heading direction. If the angle between $\hat{d}_i$ and $\hat{d}_{i+1}$ is less than $\pi/6$ or greater than $7\pi/6$, we linearly interpolate between p_i and p_{i+1} . Otherwise, compute the intersection s_i between the ray originating from p_i in the direction of $\hat{d}_i$ and the ray originating from p_{i+1} in the direction of direction $-\hat{d}_{i+1}$. Then perform a "L-shape" piecewise linear interpolation across p_i, s_i, and p_{i+1}.

To abstract away unimportant roads and to reduce problem size, we aggregate the input trajectories into a 2D image using the following voting precedure. Let h and w be the size of the bounding box of the interpolated trajectory collection $\bar{T}$ in meters. Define the **voting image of** $\bar{T}$ as a grayscale image of height $\lfloor h/resolution \rfloor$ and width $\lfloor w/resolution \rfloor$, the intensity on row i column j is computed by

$$V_{i,j} = normalize\left(\sum_{\bar{t}\in\bar{T}}\sum_{(x,y)\in\bar{t}} 1(i,j,x,y)\right)$$

Identifier function $1(i,j,t)$ evaluates to 1 if $(i)resolution < x < (i+1)resolution$, and $(j)resolution < y < (j+1)resolution$, and 0 otherwise. In our implementation, the resolution for the the voting image is chosen to be 15m.

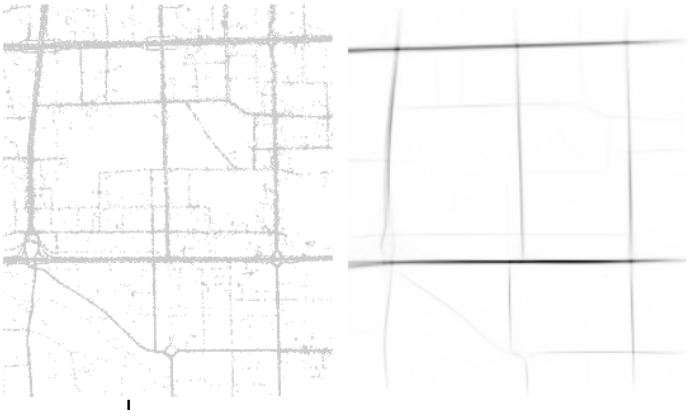


Figure 2: A portion of input trajectories visualized as point cloud (left) and as a voting image (right).

2.2 Iterative skeleton extraction

First we briefly outline the iterative skeleton extraction algorithm introduced in [1] for the problem context.

The input points $Q = \{q_j\}_{j\in J} \in \mathbb{R}^2$ used for skeleton extraction consists of all pixels in the voting image that has intensity greater than ϵ, a small positive value used to filter out noise. From Q, we use a pixel grid filter to uniformly sample a smaller set of points $X = \{x_i\}_{i\in I} \subset Q$ as the initial skeleton.

Next we solve an minimization problem that pushes X toward the L1 median skeleton of Q:

$$x = argmin \sum_{j\in J} ||x - q_j|| + R(x) \tag{1}$$

The first component moves x toward spatial median of the input points, the second term $R(x)$ is a regularization function that applies repulsive force when a skeleton branch forms locally.

$$R(x) = \sum_{i\in I} \gamma_i \sum_{i'\in I\ \{i\}} \frac{exp(-||x_i - x_{i'}||^2/(2\sqrt{h}))}{\sigma_i||x_i - x_{i'}||}$$

To decide whether a branch forms locally around skeleton point x_i, we compute the eigenvalues λ_i^0, λ_i^1 of the covariances matrix for neighbors of x_i, as in a weighted PCA. The likelihood of a branch forming is represented by the ratio $\sigma_i = \lambda_i^1/(\lambda_i^0 + \lambda_i^1)$ with $\lambda_i^0 < \lambda_i^1$.

The regularization term $R(x)$ contains a Gaussian term whose scale is controlled by the neighborhood parameter h. γ_i is a balancing constant that can be written in terms of a weight constant μ, which controls the speed of convergence.

After finding the optimal solution of Equation 1, we mark all skeleton points x_i whose weighted local average $\bar{\sigma}_i$ value is greater than a threshold $\tau = 0.8$ as branch candidate. Then iteratively pick the candidate with the highest $\bar{\sigma}$, and start tracing a branch from it among other candidates by following the PCA direction. After all candidates has been visited, we increase the neighborhood size h by $h/2$, and recompute the optimal positions of the skeleton points that has not been traced. Additional steps are used to extract "bridge points", which connects branches together.

Figure 3: Example of the first iteration in the skeleton extraction process. Figure 1-3 illustrates the contraction of skeleton points to the road centerline. In the left most figure, blue points show the initial location of the skeleton points, sampled uniformly from the voting image; The middle two images are two intermediate steps of the contraction. The right most figure draws the branches traced after the contraction stops.

2.2.1 Skeleton extraction for trajectory data

We added a few improving features on top of the original algorithm to make use of the vehicle heading data and the density information inherited from the voting image.

Directional neighborhood. Instead of using a circular neighborhood when computing distances in Equation 1, we use the heading distribution at a given location position to prune out neighbors with completely different heading directions. The difference between the circular neighborhood and the directional neighborhood is shown in Figure 4.

Figure 4: Visualization of circular neighborhood (left) and directional neighborhood (right) of a selected skeleton point. The selected point is highlighted in red, its direction is displayed by the black triangle. The neighborhood points are highlighted in cyan.

Density weighted median skeleton. In the voting image, pixels with high intensity (sample density) implies many trajectories travel through that point. Therefore they tend to have higher probabilities of belonging to a main road than pixels with lower intensity. Intuitively, we want to give higher weight to neighboring points with higher density in the first term in Equation 1. This will bias the median skeleton towards denser locations.

To compute density, we apply a Gaussian blur filter to the voting image, and denote the resulting matrix as V'. Given input point $q_j = (q_{jx}, q_{jy}) \in Q$, the density function q_j is $density(q_j) = V'(q_{jx}, q_{jy})$.

Adaptive neighborhood size. We also adjust the neighborhood parameter h based on the density.

2.3 Skeleton graph construction

To organize the extracted branches into a graph, we first remove all vertices of degree two, such that the remaining vertices are either intersections or road terminals. Then we apply a series of post-processing heuristics to smooth the graph, e.g.

- Remove dangling edges of length 1.
- Connect bridge points with its associated branch endpoints
- Merge nearby vertices into a single vertex

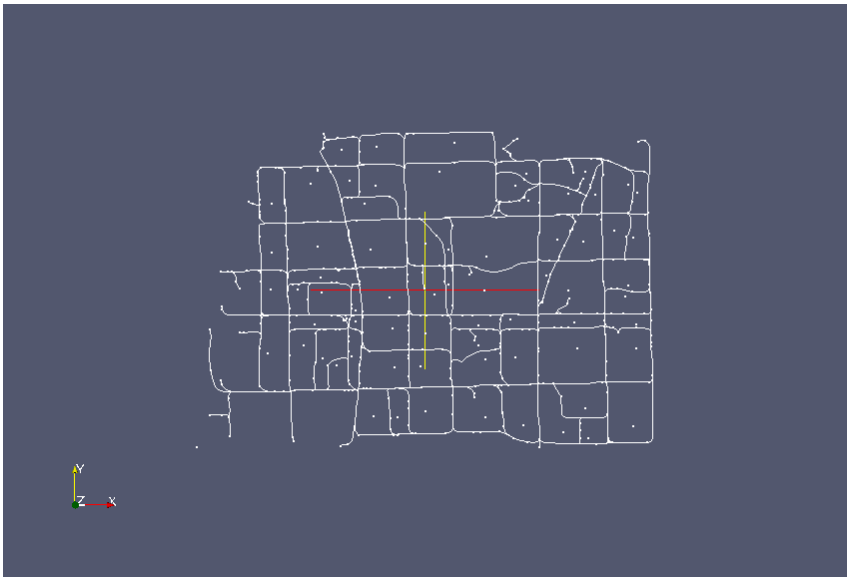
The following figures show the intermediate results for two test regions. (I hope to present better results during the presentation.)

Figure 5: Extracted junction graphs. (work in progress) The first two are computed on 500 trajectories from the Beijing taxi data set, the third one on synthesized trajectories on a grid network.

3 What's next?

Trajectory completion. The next step on this project will be integrating the extracted junction graph into the trajectory completion framework. I look forward to find out how much improvement it will have on trajectory segment clustering.

Alternative method for extracting junction graph. I have been in contact with Professor Yusu Wang from Ohio State University regarding to her work on graph reconstruction using discrete Morse complex [3][2]. The following graph is produced by running the reconstruction code on one of our test dataset. It demonstrates that persistence homology is another viable approach for junction extraction.



Courtesy of Yusi Wang, Ohio State Univeristy

Future research projects. An intriguing problem, "assessing road safety through taxi driver behaviors" has been brought up in a recent meeting with Professor Lin Zhang. I will discuss this topic during the presentation if time allows.

References

[1] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B.Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics*, 32:65:1–65:8, 2013.

[2] T. Sousbie. DisPerSE: robust structure identification in 2D and 3D. *ArXiv e-prints*, February 2013.

[3] Thierry Sousbie. The persistent cosmic web and its filamentary structure–i. theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 414(1):350–383, 2011.