Progress Report
Data-driven map matching through regularity patterns

Yang Li*

May 24, 2013

Following up my previous work on single-trajectory map matching, this report present a new formulation of our map matching algorithm that captures regularity patterns in massive number of trajectories. It also reports some findings on testing the density distribution of randomly generated trajectories on a grid map.

Overview

Map matching is the process of mapping a sequence of sparse GPS positions (trajectory) to paths on the road network (map). In most cases there are many possible mappings for a given trajectory. However, we observed from large number of trajectories and sub-trajectories that share the same source and destinations, that there often exists one or few routes that are very popular, while other viable routes are almost not taken at all. We call such behavior the *regularity* of trajectories. Our goal is to find such regularity from massive trajectory data, and enforce this regularity when projecting trajectories onto the maps. In a boarder picture, the proposed approach is motivated from the emergence of data-driven techniques in analyzing and processing 2D images and 3D shapes. These techniques have shown the strength of analyzing a collection of together as opposed to analyzing objects or pairs of objects in isolation.

We represent such regularity pattern using a small set of weighted low-order road segments, i.e., paths on the map. These road segments describe sub-trajectories that are likely to be taken by existing or future trajectories. To capture these road segments, we enumerate all road segments of a particular order, and then extract these segments with high popularity scores. where the popularity score of a road segment is defined as the sum of its similarity scores to all input trajectories.

Given the extracted road segments, the proposed approach perform map matching for each input trajectory independently. This is done by stitching a subset of road segments to form the projected path of each trajectory on the map. We formulate this step as a MAP estimation problem. The objective function considers densities of the selected road segments, the consistency between selected road segments as well as the smoothness of road segments.

Algorithm Details

Now we describe each step of the algorithm in detail. A simple grid map is used to illustrate the basic concepts.

Segment sampling

Define an **order k segment** s (or simply **k-segment**) to be the sequence of edges with total length is less than k units, while any segment that strictly contains s has length greater than k. (For simplicity, we can think the unit as kilometer.) The segment sampling step is to find S_k, the set of all k-segments in a road network.

To simplify the computation, we first interpolate long edges in the road network such that the maximum edge length is l ($l < k$). Next we enumerate through all vertices on the road network. For each vertex v, we

*Joint work with Peter Huang and Michael Kerber.

find all k-segments starting from v by traversing v's k-neighborhood[1]in a depth-first recursion. A k-segment is reported when a leaf u ($u \neq v$) of the neighborhood is reached. Figure 2. illustrates a simple case when l=1 and k=2.

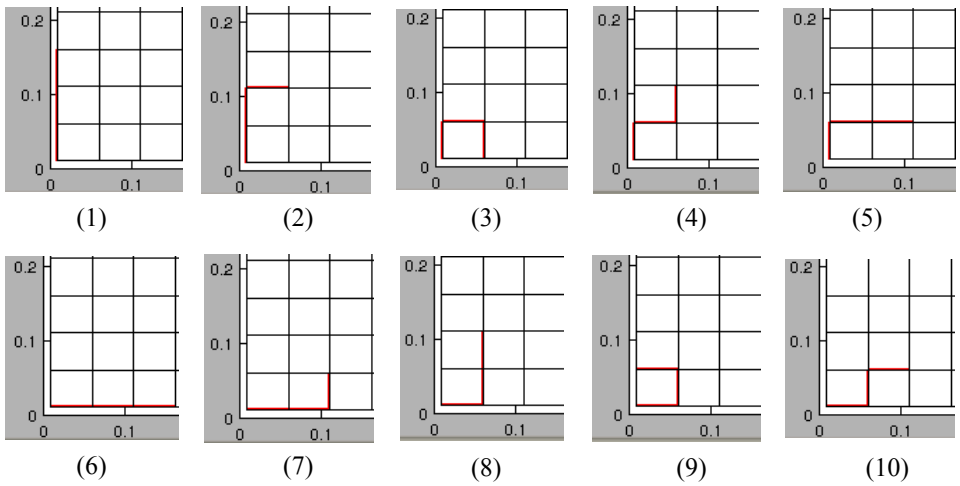

Figure 1: All 3-segments starting from $(0, 0)$

Compute segment density

The density of a segment measures how likely it can be taken by future trajectories. Because of our assumption that regularity exists in trajectories, we can approximate its value by analyzing the proximity between the segment and it neighbouring trajectories.

Let T be the set of all trajectories. The density of segment s is defined as $e^{-D(s)}$, where

$$D(s) = \sum_{T_j \in T} d_H(s, T_j) = \sum_{T_j \in T} \max_{s_i \in s} \min_{t \in T_j} d(s, t)$$

$d_H(s, T_j)$ is the one-sided Hausdorff distance. It provides an upper bound to the distance between the points on segment s and trajectory T_j. We use the Manhattan distance for $d(s, t)$ in our implementation.

[1]The k-neighborhood used here is different from the graph theory definition.

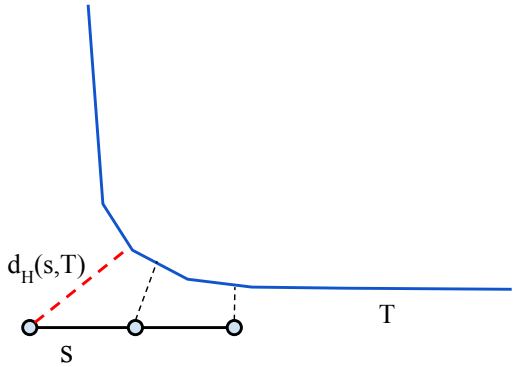

Figure 2: $d_H(s, T_j)$ is the one-sided Hausdorff distance.

Segment matching

The segment matching step maps an input trajectory to a collection of k-segments in S_k that best represent the trajectory, and conform to the regularity pattern found in existing trajectories. We achieve this by first projecting the input trajectory onto a set of candidate segments that are locally similar to the input, compute the globally optimal sequence of segments, and finally stitch the segments into a complete path. As shown in Figure 3, we are essentially lifting the trajectory into a higher dimensional space (segment space) which encodes local geometry and regularity information around the sample points.

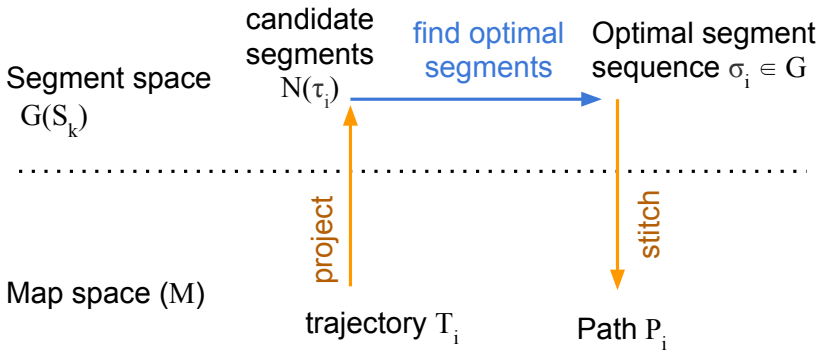

Figure 3: The segment matching process

Input processing

In order to match trajectories to segments, we subdivide the trajectories into partially overlapping components τ_i of length k around sample points. (See Figure 4 .) However, overlapping is not required when two consecutive components τ_i and τ_{i+1} are far apart.

For each sub-trajectory τ_i, we define its candidate set $N_\epsilon(\tau_i)$ to be all segments whose Hausdorff distance to τ_i is less than ϵ.

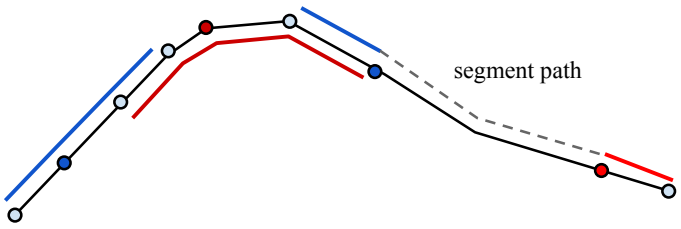

Figure 4: Subdividing a trajectory into overlapping and non-overlapping components.

Constructing segment graph

Next we construct a segment graph $G(S_k)$, where (s_i, s_j) is an edge if s_i and s_j defer by exactly 1 edge. Figure 5 shows the subgraph induced by the 10 segments given in a previous example (Figure 2). The weight of an edge $w_{i,j} = d_H(s_i, s_j)$ is the Hausdorff distance between the end segments. We will use this graph to find shortest paths between non-overlapping segments.

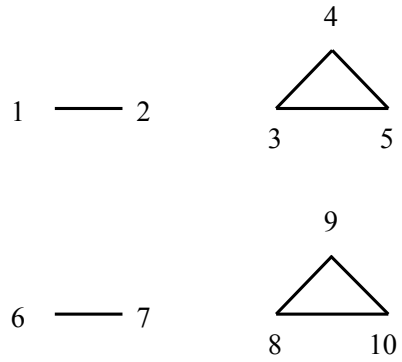

Figure 5: Subgraph of $G(S_k)$ induced by segments in Figure 2

MAP estimation (tentative)

Let $\tau_1, ..., \tau_l$ be the subdivision of trajectory T, and let $N_\epsilon(\tau_1), ..., N_\epsilon(\tau_l)$ be the candidate sets of the sub-trajectories. We build a Markov random field where the sites are τ_i, and each of which takes value from the finite set $N_\epsilon(\tau_i)$.

Let A be an assignment function that maps τ_i to some $s_j \in N_\epsilon(\tau_i)$. We define indicator vector x, [2] where $x_{i,a} = 1$ if $A(\tau_i) = s_a$, and 0 otherwise. We find the optimal assignment by solving the MAP estimation as follows

$$x^* = \text{argmin}_x \sum_{i,a} W_{i,a} x_{i,a} + \lambda \sum_{a,b} Q_{i,a;i+1,b} x_{i,a} x_{i+1,b}$$

$$\sum_a x_{i,a} = 1$$

The first order potential function W captures the inverse density of segment s_a and its deviation from τ_i.

$$W_{i,a} = -D(s_a) + \gamma_1 d_H(s_a, \tau_i)$$

The higher order potential Q is the distance from s_a to s_b on the segment graph $G(S_k)$.

$$Q_{i,a;i+1,b} = d_{G_k}(s_a, s_b)$$

We can solve this optimization problem efficiently as in the single trajectory map matching algorithm.

Trajectory stitching

After obtaining optimal path $\sigma = \{s_1, ..., s_M\} \in G(S_k)$, it is easy to compute its corresponding path P in road network M, since all consecutive segments s_i and s_{i+1} differ only by one vertex.

Experiments on segment density

The goal of the following experiment is to verify two important facts that our algorithm relies on: *1). Only a small subset of segments have high density, and 2)trajectories can be constructed incrementally via low order segments* . Here is a description of the experiment setup.

On a 19×19 grid map, we generated 10,000 sparse trajectories of varying length. Since we want these trajectories to simulate GPS trace of vehicles driving in a city, we enforce the following additional properties:

[2] For clarity, we define x as a $|S_k|$ dimension vector. However, in the dynamic programming implementation, we only consider segments in the candidate sets, hence $|x| = |T| max_{\tau \in T}(|N(\tau)|)$.

- Trajectories can not turn 180°, and have a limited number of left and right turns;
- All trajectory are within the boundary of the map (no cut off trajectories);
- The maximum deviation of a trajectory point from the nearest road is ϵ.

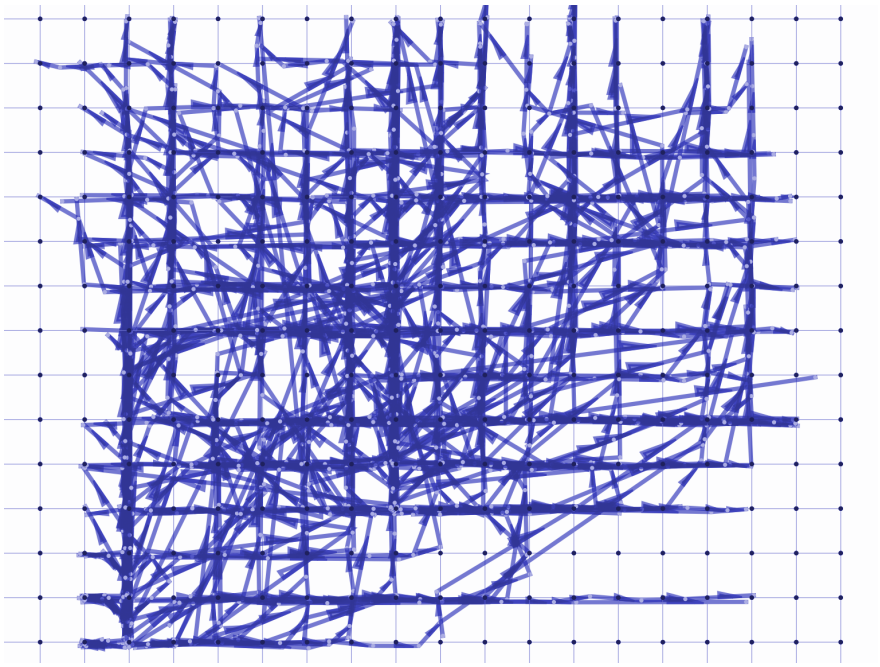

Figure 6: 200 trajectories from the generated dataset

The number of k-segments on the map for $k = 1, 2, 3\ and\ 4$ are as listed in Table 1. By pruning trajectories that are far away from the queried segment, we were able to compute the density efficiently.

k	$\lvert S_k \rvert$
1	1520
2	4328
3	12456
4	35960

Table 1: Total number of k-segments in the 19×19 grid map

The following plot shows the density distribution of all k-segments for $k = 1, 2, 3, and 4$. We can see that no matter what the value of k, at least 2/3 of the segments always have density very close to 0. In practice, we can safely ignore these low density segments so that the search space in the optimization step is not too large.

The next steps

Currently I am working on the implementation of the segment mapping step. After the system is in place, I plan to perform more experiments for finding the best k for a given map, and for adjusting other parameters in the algorithm.

We have also considered two ways of validating our algorithm. First we will compare the matching result with the outcome of the single trajectory map matching algorithm I implemented previously. (Our hypothesis is that the proposed method will yield better accuracy and regularity.) We also plan on manually

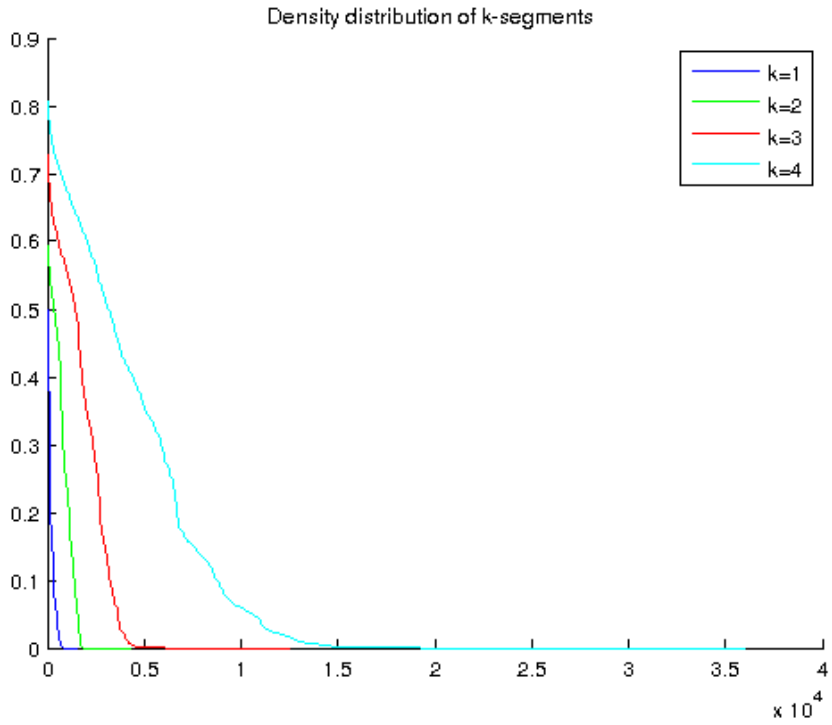

Figure 7: Density distribution of k-segments, sorted in descending order

labelling a small subset of trajectories to test whether the regularity found in our algorithm is consistent with human perceptions.