Large-Scale Joint Map Matching of GPS Traces

Yang Li
Stanford University
Stanford, CA, USA
yangli1@stanford.edu

Qixing Huang
Stanford University
Stanford, CA, USA
huangqx@stanford.edu

Michael Kerber
Stanford University and
MPC-VCC
Stanford, CA, USA
mkerber@mpi-inf.mpg.de

Lin Zhang
Tsinghua University
Beijing, China
linzhang@tsinghua.edu.cn

Leonidas Guibas
Stanford University
Stanford, CA, USA
guibas@cs.stanford.edu

ABSTRACT

We present a robust method for solving the map matching problem exploiting massive GPS trace data. Map matching is the problem of determining the path of a user on a map from a sequence of GPS positions of that user — what we call a trajectory. Commonly obtained from GPS devices, such trajectory data is often sparse and noisy. As a result, the accuracy of map matching is limited due to ambiguities in the possible routes consistent with trajectory samples. Our approach is based on the observation that many regularity patterns exist among common trajectories of human beings or vehicles as they normally move around. Among all possible connected k-segments on the road network (i.e., consecutive edges along the network whose total length is approximately k units), a typical trajectory collection only utilizes a small fraction. This motivates our data-driven map matching method, which optimizes the projected paths of the input trajectories so that the number of the k-segments being used is minimized. We present a formulation that admits efficient computation via alternating optimization. Furthermore, we have created a benchmark for evaluating the performance of our algorithm and others alike. Experimental results demonstrate that the proposed approach is superior to state-of-art single trajectory map matching techniques. Moreover, we also show that the extracted popular k-segments can be used to process trajectories that are not present in the original trajectory set. This leads to a map matching algorithm that is as efficient as existing single trajectory map matching algorithms, but with much improved map matching accuracy.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

SIGSPATIAL'13, November 05 - 08 2013, Orlando, FL, USA

http://dx.doi.org/10.1145/2525314.2525333 .

Figure 1: Left: A sparse trajectory (black) and its ground truth path (green); Right: Projected paths of the trajectory in the left figure using data-driven map matching (blue) and single-track map matching (red).

General Terms

Algorithm, Theory

Keywords

map matching, data-driven, trajectory regularity, dynamic programming, GPS data

1. INTRODUCTION

Map matching is the procedure of determining the path of a user on a map from a sequence of GPS positions of that user – what we call a *trajectory*. This procedure finds use in many mobility related application, such as urban traffic modeling [1][2], dynamic road map generation [3], and mobility pattern mining [4]. Since collecting highly accurate GPS traces on a city scale is quite costly, most of the trajectory data available today were obtained indirectly through GPS-equipped vehicles or users with GPS-enabled cellular phones. The majority of the collected trajectories inevitably contain a large amount of uncertain and incomplete information. For example, one form of uncertainty comes from GPS noise, which is particularly severe in urban environment due to signals obstructed by or reflected off buildings (urban canyons). Incomplete data is often the result of a low sampling rate, due to limitations on storage and communication bandwidth. For instance, 50% of the Beijing Taxi Trajectories we employed in this study have at most one

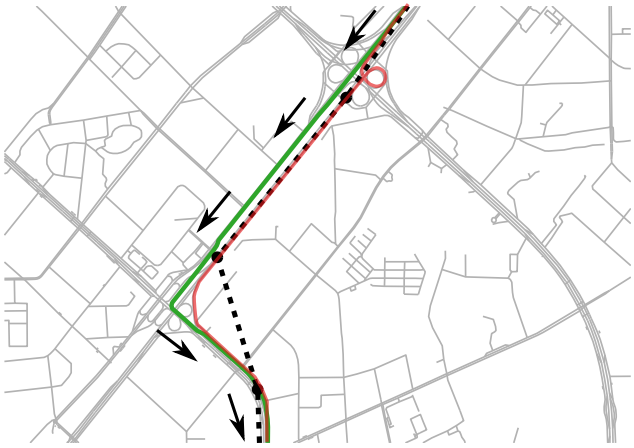


Figure 2: We show an example of a trajectory (black) whose underlying path (green) differs from the path that is closest to the trajectory samples points (red). The arrows indicate the travel direction of the user.

sample per minute. Building a robust algorithm for map matching under such conditions is therefore a challenging task that has led to ongoing research over several years.

Most existing map matching algorithms take a single GPS trajectory as input. We refer them as *single-track map matching algorithms* (SMM) [5]. They are typically formulated with the objective of minimizing the distance between the projected path on the map and the input trajectory, and of achieving some other regularization objectives, such as minimizing the length of the path. These algorithms work well when the input trajectory is densely sampled and the sampling error is small. However, their performance drops significantly when the input trajectory becomes noisy and sparse. In this case, the estimated path does not necessarily need to be close to the input trajectory, and it may not always follow or approximate the shortest path on the map. (See Figure 2.)

We propose to address these issues using *multi-track map matching*, i.e. simultaneously matching a collection of trajectories to a map. The advantage of this approach comes from the observation that human trajectories show a "high degree of temporal and spatial regularity" [6]. In the context of map matching, we have observed large amount of regular structures in vehicle trajectories despite being driven by different drivers. Hence the aim of multi-track map matching is to recover the *regularity* patterns (i.e., frequently used road segments) among the input trajectories, and to preserve the regularity in the matched paths. From a data driven perspective, multi-track map matching offers additional regularization constraints that improve the map matching results of individual trajectories — effectively using the "wisdom of the collection" to compensate for noise and gaps in individual trajectories. Specifically, for a set of partially overlapping trajectories, we enforce that the projected paths of their overlapping regions coincide. Such a formulation implicitly increases the sampling density of trajectories. Moreover, the overlapping parts of these trajectories are jointly determined, which improves the robustness of the map matching procedure. The multi-track idea was first seen in [5], with the assumption that all trajectories are sampled from the same underlying path. Our algorithm, designed to apply multi-track map matching on heterogeneous data, offers more practical use on large-scale map matching applications.

1.1 Approach Overview

The input to our problem is a road map and a collection of trajectories. The output consists of the projected paths of the trajectories on the map. We tackle multi-track map matching of heterogeneous trajectories in three main steps.

1. *Segment Initialization:* Extract a collection of road segments $\mathcal{S}$ in the given map. These road segments will be the building blocks of all projected paths.
2. *Joint Segment Selection:* Assign a segment from $\mathcal{S}$ to each trajectory sample point so that all segments collectively optimize some object function.
3. *Map Matching using Selected Segments:* Stitch the selected segments of each trajectory into its projected path.

The Joint Segment Selection step plays the key role in our algorithm. We formulate this stage as solving a constrained optimization problem. The variables are binary indicators that specify whether a segment is selected for a particular trajectory sample point. The objective function combines a matching term, a stitching term, and a regularity term, as illustrated in Figure 4. The matching term evaluates the proximity (in terms of a distance metric to be defined) between a segment and a trajectory in the neighborhood of a sample point. The stitching term enforces that the selected segments of each pair of trajectory sample points are consistent. Last but not least, the regularity term, which plays the data-driven role, forces more segments to be shared among partially overlapping trajectories.

To effectively solve the proposed optimization problem in the Joint Segment Selection step, we introduce latent variables to reformulate the objective function into a form that is easier to optimize. This reformulation admits alternating optimization, where the optimization task at each step can be decoupled into small-scale optimization sub-problems, including ones with close-form solutions. In practice, these sub-problems can be solved in parallel, allowing us to process hundreds of thousands of trajectories in a couple of hours.

Another advantage of the proposed approach is its ability to handle trajectories not present in the "training" dataset, assuming that the training set contains sufficiently many trajectories.

We have also created a benchmark for evaluating map matching algorithms, including our own. The benchmark data were sampled from a large collections of GPS traces recorded by 8602 taxi cabs in Beijing, China, during May, 2009. Experimental results using the benchmark show that the proposed approach outperforms existing single trajectory map matching algorithms in matching accuracy.

1.2 Related Work

Map matching algorithms are generally categorized as *incremental* or *global* algorithms. The former match portions of a trajectory onto a path based on the local geometry. This is also known as "online map matching" since it can generate vehicle paths on the fly as new samples become available.

(a) Input trajectories (b) Candidate search (c) Joint segment selection (d) Map matching by stitching selected segments

Figure 3: Workflow of data-driven map matching for two trajectories. The dotted lines in (b)-(d) represent input trajectories. Solid-colored lines in (b) and (c) represent candidate segments for a trajectory point.

Such algorithms are often used in applications such as navigation, where the sampling rate is relatively high (1-30s per sample), but a trade off has to be made between accuracy and the speed of computation. In contrast, global algorithms find the globally optimal path after reading in complete trajectories. Because of their offline nature, these algorithms tend to focus on map matching accuracy and robustness.

One group of global algorithms is based on curve fitting using Fréchet distance and its variants [7][8]. However, these algorithms are often designed and work well only for densely sample trajectories. Statistical models are popular tools for handling uncertainty in trajectory data. In particular, Hidden Markov Model based algorithms [9][10] have proven to recover accurate paths from relatively sparse trajectories. In this method, the trajectory is treated as a sequence of noisy observations of the true positions along the paths, modeled as a Markov chain. The emission probability is found using a Gaussian noise model around the observation, the transition probability is based on the topological and geometric constraints on the map, and the matched path is the one with maximum a posteriori probability. Other statistical models such as Conditional Random Fields [11] have also been applied to reduce the sampling bias in simple HMM models.

Statistical methods can be considered a subset of the general max-weight algorithms. In general, a candidate set is determined for each observation, and each candidate is assigned a weight based on its offset from the observation and topological fitness with candidates of neighboring samples. Then the output is a sequence with the maximum weight. Algorithms in this category include the shortest-path based ST matching [12] and the interactive voting method [13]. Max-weight methods can often be adapted to online settings using approximation techniques [14][15]. As an alternative to point-wise matching, segment-based algorithms match piecewise trajectory segments to path segments [16]. Our proposed approach fits within the framework of segment-based map matching.

Recent works on map matching often adopt data-driven techniques to improve the matching result using historical trajectories. One common application is determining the parameters of the statistical models or the weight function, using machine learning techniques [15][11]. This approach makes it easier to fine tune the model, but it also has the tendency of over-fitting the predefined model. In another approach presented in [17], historical trajectories are directly processed into reference routes that can be queried to generate new routes. The algorithm is able to complete the matching even with very sparse data. However, as the generated path is solely based on a fairly accurate reference route database, it may fail at the case when a trajectory is sampled from a path not in the database.

Although drawing information from large scale trajectory data, the aforementioned methods only match a single single trajectory at a time. They are referred as single-track map matching algorithms in [5]. The focus of this work is in multi-track map matching, where one matches a large number of possibly sparse trajectories simultaneously to the map. In [5], multiple sparse trajectories of the same route, and with the same starting and ending positions are used to produce an accurate path on the map. This method extracts a global order on the sample points based on the partial order defined by individual trajectories, then uses a single-track map matching to produce the matched path.

1.3 Paper Organization

The first half of this paper is organized around the three main steps of the algorithm. Section 2 introduces the formal definition for road segments and explains how to extract them from a road network; Sections 3 explains the joint segment selection procedure; Section 4 describes how to compute the projected paths from selected segments for trajectories both present and not present in the input dataset.

The second half focuses on the experimental evaluation of the algorithm on a large collection of taxi trajectories (Section 5). This is followed by a discussion on challenges and future prospects of data-driven map matching in Section 6.

2. SEGMENT DEFINITION & INITIALIZATION

The initial step of the proposed algorithm generates a collection of road segments that represent all possible paths within the map. We refer them as *order-k-segments*.

Defining k-segments. We define the road network (a.k.a. map) $\mathcal{M}$ as a drawing of an oriented graph in $\mathbb{R}^2$, i.e., the vertices of $\mathcal{M}$ represent intersections, and the edges of $\mathcal{M}$ represent road segments that connect adjacent intersections. Each road segment is represented as a polygonal curve.

DEFINITION 1. *Let $||e_i||$ denote the length of edge e_i in $\mathcal{M}$. An order-k-segment (or simply k-segment) is a sequence of s nonrepeating edges $e_1, \ldots, e_s$ that satisfies $\sum_i^{s-1} ||e_i|| < k$ and $\sum_i^{s} ||e_i|| \geq k$. i.e. The total length of a k-segment is approximately k units.*

Practically, we may also represent a k-segment as a sequence of vertices, and we say a point p is in segment s if p is an endpoint of some edge e_i in s. The choice of k usually depends on the data (See Section 5.2 for experimental results on the effect of segment order). In this paper, we typically use $k = 3$ kilometers (km).

With $\mathcal{S}$ we denote the set that collects all segments of order k, while omitting k in the notation by considering it as a constant. In practice, we also incorporate total turning angles in the computation of segments to prevent repeating edges. (See Section 2.)

Generating segments from a map. Below we introduce a simple approach to obtain a good representation of segment collection $\mathcal{S}$. Notice that this procedure only needs to be done once per map, therefore it does not impact the efficiency of the map matching algorithm.

For each vertex v in the map, we collect all k-segments starting from v through traversing the relevant network connected component in a breadth first search manner. Let p be the path from v to a connected node u. We report a k-segment when one of the following is true:

1. The total length of p is less than or equal to k, but extending p by adding any of u's outgoing edge makes p's length exceed k.
2. The total turning angle exceeds 2π.

The second criterion effectively reduces loops and self intersection. It also penalizes sharp turns by forcing such segments to be shorter, thus less likely to be selected in the next step.

3. JOINT SEGMENT SELECTION

3.1 Problem formulation

In this section, we describe in detail the formulation of joint segment selection as an optimization problem. We begin with defining a distance metric between a segment and a trajectory point, followed by explaining the objective and constraints in the proposed formulation.

DEFINITION 2. *A trajectory $t = \mathbf{v}_1 \cdots \mathbf{v}_{|t|}$ is an ordered sequence of points in $\mathbb{R}^2$. With $|t|$ we denote the cardinality of t.*

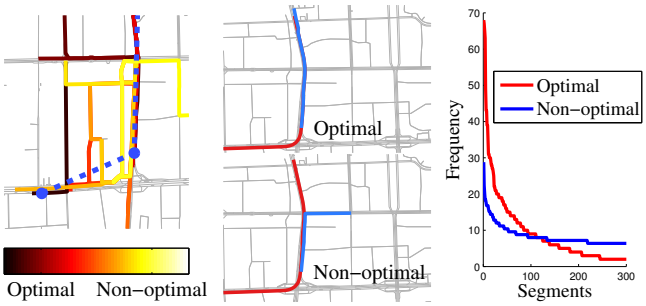
Each trajectory is considered a sparse and noisy sampling of an underlying path p_t in map $\mathcal{M}$. Note that p_t may start and end in the middle of edges of $\mathcal{M}$.

DEFINITION 3. *The distance between a segment s and a trajectory sample point $\mathbf{v}_{t,i}$ is*

$$d_{\text{curve}}(s, \mathbf{v}_{t,i}) = \max\left\{d(\mathbf{v}_{t,i}, s), \max_{\mathbf{p}\in s} d(\mathbf{p}, t)\right\},$$

where $d(\mathbf{p}, c)$ denotes the Euclidean distance between point $\mathbf{p}$ and its closest point in piecewise linear curve c.

While it is straightforward to use the distance between a point and a segment, $d(v_{t,i}, s)$, in the map matching optimization, we add a regularizing term $max_{\mathbf{p}\in s} d(\mathbf{p}, t)$ to reflect the consistency between the segment and the whole trajectory. If $max_{\mathbf{p}\in s} d(\mathbf{p}, t)$ is very large, there is a good chance that the segment is globally incompatible to the trajectory, even if the evaluated point $d_{t,i}$ is close to the segment, indicating local compatibility. We therefore penalize such options by taking the maximum of the two distances.



(a) Matching (b) Stiching (c) Regularity

Figure 4: Illustration of the three objective terms in joint segment selection. (a) Segments in darker colors are closer to the input trajectory (dotted blue line) than segments in lighter colors, hence having higher matching scores. (b) The segment assignment in the upper figure has higher stitching score than the one in the lower figure, since the stitching term favors segments of consecutive trajectory points that are consistent. (c) This figure displays the frequency drop-off of 300 most popular segments in two different segment assignments. Comparing to the blue curve, the red curve has a steeper fall-off that signifies a higher regularity score. i.e. only a small number of segments are selected among all input trajectories.

Given the input trajectories $\mathcal{T} = \{t = (\mathbf{v}_{t,1} \cdots \mathbf{v}_{t,|t|})\}$ and the set of all k-segments $\mathcal{S}$ in $\mathcal{M}$, we solve a constrained optimization problem to jointly select a candidate segment $s_{t,i} \in \mathcal{S}$ for each trajectory sample point $\mathbf{v}_{t,i}$. In the following, we first describe how to parametrize the optimization variables, i.e., the selected segments of each trajectory. Then we present the expression of each term in the optimization objective.

Variable parametrization. We introduce binary indicator variables $x^s_{t,i} \in \{0, 1\}$ to specify whether a segment $s \in \mathcal{S}$ is selected for a trajectory sample point $\mathbf{v}_{t,i}, 1 \le i \le |t|, t \in \mathcal{T}$, i.e., $x^s_{t,i} = 1$ if s is the selected segment of $\mathbf{v}_{t,i}$, and $x^s_{t,i} = 0$ otherwise.

For each $v_{t,i}$, we filter out segments with a large distance in order to reduce the number of indicator variables: Let d_{max} be an upper threshold for the allowed distance of trajectory point and matched segment; we set $d_{max} = 200m$ in this paper. Using classical range search data-structures such as quad-trees

$$\mathcal{S}_{t,i} = \{s \mid s \in \mathcal{S}, d_{\text{curve}}(s, \mathbf{v}_{t,i}) < d_{\max}\},$$

Given the candidate sets, we remove all binary indicators $x^s_{t,i}$ from the variable set if $s \notin \mathcal{S}_{t,i}$. With the remaining indicator variables, we formulate the constraint that exactly

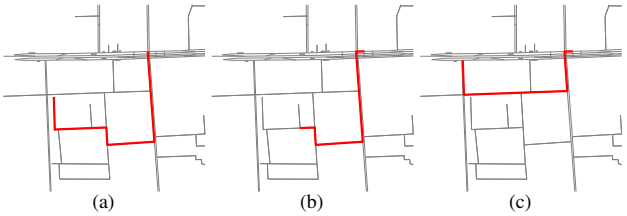


Figure 5: Sample k-segments from the map of Beijing. Segments a and b are consistent, while Segment c is not consistent with either a or b, despite of having overlapping edges with both.

one candidate segment can be selected as a linear equality:

$$\sum_{s\in\mathcal{S}_{t,i}} x^s_{t,i} = 1. \tag{1}$$

Matching term. The matching term f_{match} prioritizes the fact that the selected segments are close to the corresponding sub-trajectories. Using segment indicators, we formulate the matching term as

$$f_{\text{match}} = \sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|}\sum_{s\in\mathcal{S}_{t,i}} d_{\text{curve}}(s,\mathbf{v}_{t,i})x^s_{t,i}. \tag{2}$$

Stitching term. The stitching term f_{stitch} measures the consistency of the selected segments of each input trajectory $t\in\mathcal{T}$. First we introduce *consistency*, an asymmetric relation for a segment pair. (See Figure 5.)

DEFINITION 4. *A segment s is consistent with another segment s', or equivalently, they can be merged into a single segment of order $\geq k$, if there exists three segments s_1, s_2 and s_3 such that*

$$s = s_1 s_2, \quad s' = s_2 s_3.$$

Using the consistency relation, we define the stitching score $\text{stitch}(s,s')$ between two segments $s\in\mathcal{S}_{t,i}$ and $s'\in\mathcal{S}_{t,i+1}$:

$$\text{stitch}(s,s') = \begin{cases} 1-\frac{\text{length}(s\cap s')}{\text{length}(s\cup s')} & s \text{ is consistent with } s' \\ 1 & \text{otherwise} \end{cases}$$

In other words, $\text{stitch}(s,s') < 1$ if and only if s and s' are consistent (i.e., they can be merged into a segment), in which case $\text{stitch}(s,s')$ specifies their overlap ratio. We then define f_{stitch} by summing up the stitching scores of all pairs of neighboring selected segments:

$$f_{\text{stitch}} = \sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|-1}\sum_{s\in\mathcal{S}_{t,i}}\sum_{s'\in\mathcal{S}_{t,i+1}} \text{stitch}(s,s')\; x^s_{t,i}x^{s'}_{t,i+1}. \tag{3}$$

Regularity term. Merely optimizing the matching term and the stitching term will match each trajectory in isolation. We thus introduce a regularity term $f_{\text{regularity}}$ that counts the total number of selected segments among all trajectories. The resulting objective encourages segments to be shared by partially overlapping trajectories.

Let $n(s) = \sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i}$ be the number of times that a segment $s\in\mathcal{S}$ is selected. A straight-forward definition of the regularity term is given by $\sum_{s\in\mathcal{S}} Id(n(s)>0)$, where indicator function $Id(n(s)>0)=1$ if $n(s)>0$ and $Id(n(s)>0)=0$ otherwise. However, such an expression is hard to optimize.

Our formulation is based on the fact that the sum of $n(s)$ over all segments is a constant:

$$\sum_{s\in\mathcal{S}} n(s) = \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i} = \sum_{t\in\mathcal{T}}|t|.$$

We propose to minimize a concave function over $n(s)$, which prioritizes the sparsity in $n(s)$ and thus implicitly minimizes the number of selected segments. In our implementation, we choose the natural logarithm function and define $f_{\text{regularity}}$ as

$$f_{\text{regularity}} = \sum_{s\in\mathcal{S}}\log(n(s)) = \sum_{s\in\mathcal{S}}\log(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i}+\delta), \tag{4}$$

where δ is introduced to make $f_{\text{regularity}}$ well-defined. In practice, we set $\delta = 10^{-4}$.

Formulation. Combining Equations 1-4, we arrive at the following constrained optimization formulation for joint segment selection:

$$\begin{aligned}\min_{x^s_{t,i}\in\{0,1\}} \quad & \sum_{t\in\mathcal{T}}\Big(\sum_{i=1}^{|t|}\sum_{s\in\mathcal{S}_{t,i}} \text{match}(s,v_{t,i})x^s_{|t|,i} \\ & + \lambda\sum_{i=1}^{|t|-1}\sum_{s\in\mathcal{S}_{t,i}}\sum_{s'\in\mathcal{S}_{t,i+1}} \text{stitch}(s,s')x^s_{t,i}x^{s'}_{t,i+1}\Big) \\ & + \mu\sum_{s\in\mathcal{S}}\log(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i}+\delta) \\ s.t. \quad & \sum_{s\in\mathcal{S}_{t,i}} x^s_{t,i} = 1, \quad \forall t\in\mathcal{T},\ 1\leq i\leq|t|. \end{aligned} \tag{5}$$

where

$$\text{match}(s,v_{t,i}) = \sum_{i=1}^{|t|}\sum_{s\in\mathcal{S}_{t,i}} d_{\text{curve}}(s,\mathbf{v}_{t,i})$$

The weights λ and μ specify the importance of the stitching term and the regularity term, respectively. In this paper, we choose $\lambda=10$ and $\mu=0.5$ for all the experiments.

3.2 Numerical Optimization

Directly solving Equation (5) is hard since *i*) the number variables is large, and *ii*) all variables have integer values. Instead, we reformulate Equation (5) by introducing latent variables, and optimize both the original variables and the latent variables in an alternating manner. In the following, we first explain the derivation of the reformulation, then discuss the alternating optimization strategy and its convergence behavior.

Reformulation. The key idea of the our new formulation is to rewrite the term $\log(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i}+\delta)$ as the solution of a minimization problem.

PROPOSITION 5. *The term* $\log(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x^s_{t,i}+\delta)$ *admits the*

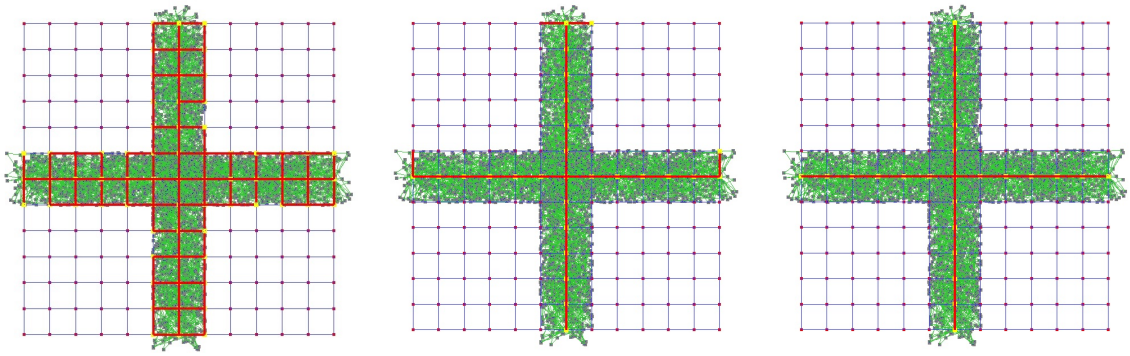


Figure 6: Convergence of k-segments in the first 3 iterations of alternative optimization on a toy example. We randomly generated 100 trajectories crossing an intersection on a grid map from different directions. The matched segments converge to the skeleton of the intersection in 3 iterations.

following expression:

$$\log(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x_{t,i}^s + \delta)$$
$$= \min_{y_s}\Big(y_s\big(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x_{t,i}^s + \delta\big) - \log(y_s)\Big) - 1. \quad (6)$$

Proof. By setting

$$\partial(y_s\big(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x_{t,i}^s + \delta\big) - \log(y_s))/\partial y_s = 0,$$

we observe a global minimum at

$$y_s = 1/(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x_{t,i}^s + \delta). \quad (7)$$

and the claimed equality can easily be verified. □

Substituting (6) into (5), we introduce latent variables y_s, for all $s \in \mathcal{S}$ to be optimized together with $\{x_{t,i}^s\}$. Equation (5) is reformulated into the following:

$$\min_{x_{t,i}^s\in\{0,1\}, y_s} \sum_{t\in\mathcal{T}}\Big(\sum_{i=1}^{|t|}\sum_{s\in\mathcal{S}_{t,i}}\big(\text{match}(s, v_{t,i}) + \mu y_s\big)x_{t,i}^s + \lambda\sum_{i=1}^{|t|-1}\sum_{s\in\mathcal{S}_{t,i}}\sum_{s'\in\mathcal{S}_{t,i+1}}\text{stitch}(s,s')x_{t,i}^s x_{t,i+1}^{s'}\Big) + \mu\sum_{s\in\mathcal{S}}(\delta y_s - \log(y_s))$$
$$s.t. \quad \sum_{s\in\mathcal{S}_{t,i}} x_{t,i}^s = 1, \quad \forall t\in\mathcal{T},\ 1\le i\le n_t. \quad (8)$$

Alternating Optimization We solve (8) by alternating the optimization of variables $\{x_{t,i}^s\}$ with the optimization of variables $\{y_s\}$. As variables $x_{t,i}^s$ associated with different trajectories are decoupled, variables y_s of different s also get decoupled. Hence the alternating optimization at each step is equivalent to solving independent optimization subproblems for each trajectory or each segment. The details are described as follows:

1. *Optimizing* $\{x_{t,i}^s\}$. When $y_s, s \in \mathcal{S}$ are fixed[1], solving (8) is equivalent to solving the following optimization problem for each trajectory $t \in \mathcal{T}$:

$$\min_{x_{t,i}^s\in\{0,1\}} \sum_{i=1}^{|t|}\sum_{s\in\mathcal{S}_{t,i}}\big(\text{match}(s, v_{t,i}) + \mu y_s\big)x_{t,i}^s + \lambda\sum_{i=1}^{|t|-1}\sum_{s\in\mathcal{S}_{t,i}}\sum_{s'\in\mathcal{S}_{t,i+1}}\text{stitch}(s,s')x_{t,i}^s x_{t,i+1}^{s'}$$
$$s.t. \quad \sum_{s\in\mathcal{S}_{t,i}} x_{t,i}^s = 1, \quad 1\le i\le n_t. \quad (9)$$

The latter is a special quadratic integer program where the quadratic terms are defined on consecutive candidate sets. It is well known that such problems can be efficiently solved using dynamic programming [9].

2. *Optimizing* $\{y_s\}$. When $\{x_{t,i}^s\}$ are fixed, solving (8) is equivalent to solving the following optimization problem for each segment $s \in \mathcal{S}$:

$$\min_{y_s\in\mathbb{R}^+} y_s(\sum_{t\in\mathcal{T}}\sum_{i=1}^{|t|} x_{t,i}^s + \delta) - \log(y_s). \quad (10)$$

According to Proposition 5, it is clear that the optimal solution to (10) is given by

$$y_s = 1/(\sum_{t\in\mathcal{T}}\sum_{i=1}^{n_t} x_{t,i}^s + \delta).$$

Let y_s^k denote the value of y_s at the kth iteration. We stop the alternative optimization when the following condition is met.

$$\text{mean}\left\{||y_s^k - y_s^{k+1}||\right\}_{s\in S} \le 10^{-4}.$$

In practice, the algorithm often terminates within 8-9 iterations.

We can interpret the above alternating optimization strategy as follows. At each iteration, we first perform map-matching for each trajectory independently. Then we update a *frequency score* of each segment, i.e., y_s, used to prioritize segments in the map matching step. As a result, segments that are frequently used in the previous iteration are prioritized in the current iteration. Experiments confirm

[1] At the first iteration, we set $y_s = 0, s \in \mathcal{S}$

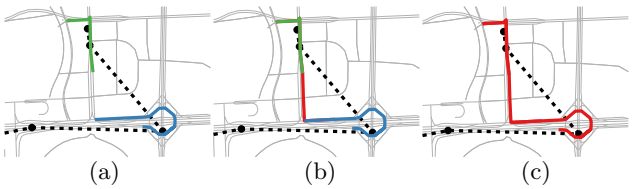


Figure 7: This figure illusrates the two-step process of stitching selected segments. From (a) to (b), an additional segment (red) is added to connect the selected segments. From (b) to (c), we generate the projected path (red) by tracing the selected segments from the starting point to the ending point.

that the number of selected segments decreases during this alternating process, at a speed controlled by parameter μ. In the end, only a fraction of segments are selected for map matching.

4. MAP MATCHING USING SELECTED SEGMENTS

The final step of data-driven map matching is to compute the projected path from the set of selected segments. We will consider two scenarios: *i*) Matching input trajectories; *ii*) Matching additional trajectories not present in the input dataset.

Matching Input Trajectories. Finding the projected path for input trajectories is simply stitching the selected segments for the sample points in each trajectory t.

As illustrated in Figure 7, we first build a segment graph where two segments s, s' are connected if they are consistent with each other, with edge weight $stitch(s, s')$. The segment merging process involves two passes over t's selected segments:

The first pass finds pairs of disconnected segments s_i and s_{i+1}, then adds between them the shortest path from s_i to s_{i+1} in the segment graph. This can be done using A^* search starting at s_i.

For the second pass, consider the connected subgraph G' consisting of all selected segments and segments added in the first pass. We trace a shortest path from s_i to s_{i+1} to eliminate extraneous edges. The final path p_t will be a sequence of non-repeating edges on the shortest path.

Matching Additional Trajectories. The joint segment selection process detailed in Sections 3 can be considered as a training stage for map matching additional trajectories within the map. After the alternative optimization step, we have learned a collection of k-segments that best represent regular structures in all training trajectories, with respective weight y_s^*. Assuming there are sufficiently many training trajectories covering the entire map, we can reuse the extracted k-segments to perform a segment-based single-track map matching on trajectories not in the training set, as illustrated in Figure 9.

Let S^* be the collection of all k-segments matched to the original trajectory set T. To find the underlying path of trajectory $t \notin T$, we first compute the candidate segments of every sample point $\mathbf{v}_{t,i}$ on t in S^*.

$$\mathcal{S}^*{}_{t,i} = \{s \mid s \in \mathcal{S}^*, d_{\text{curve}}(s, \mathbf{v}_{t,i}) < d_{\max}\},$$

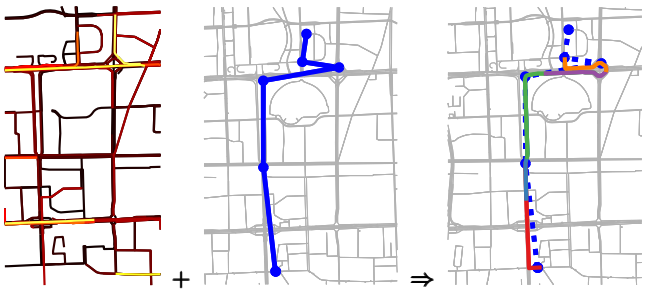


Figure 9: Map matching additional trajectories. Left: k-segments obtained from the joint segment selection step. Lighter segments have higher weights than the darker ones. Center: The trajectory to be matched; Right: A segment-based single-track map matching is applied to the new trajectory.

We then solve Equation 9 for the optimal candidate assignment $x_{i,j}$, where variables y_s are substituted with learned parameters y_s^*. The final path is obtained using the segment stitching method described earlier.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

We created a benchmark dataset for evaluating the performance of map matching algorithms. The trajectories used in the benchmark were taken from the 2009 Beijing Taxi Cab dataset [18]. This dataset contains the GPS traces of 8602 taxi cabs in Beijing, China, collected during an one-month period in May, 2009. (See Figure 11.) The sampling density of these trajectories ranges from 60 seconds to 5 minutes. For our benchmark dataset, we used 121,737 trajectories of 2-minute sampling intervals. We obtained the map data from Open Street Map [19]. The map region covers all trajectories ranges between 116.148° - 116.613° longitude and 39.788° - 40.096° latitude.

We construct the benchmark dataset from 100 randomly selected sparse trajectories. To obtain ground truth map matching results of these trajectories, we recruited four volunteers who had 5+ years of driving experience in Beijing to manually trace the path on the map. As shown in Figure 10, each volunteer was given an interactive application, which showed the input trajectory and a map of the surrounding area. The purpose of this activity is to simulate how a real driver would choose the path based on their knowledge about the road scenario. Each volunteer was responsible for 25 trajectories, so that they brought a level of diversity to the ground truth.

As a baseline comparison, we implemented the HMM based map matching algorithm described in [9]. We chose this algorithm since it is often cited for its robustness under sparse data, while other algorithms we attempted did not perform notably better than the baseline algorithm with our dataset.

We evaluate the performance of a map matching algorithm by computing the distances between the resulting projected paths and the ground truth paths. Specifically, we compute the percentage of vertices whose minimum Euclidean distances to the corresponding ground truth path is below some distance threshold ϵ. We then plot $p(\epsilon)$ with respect to ϵ, which ranges between 0 and 20 meters.

Figure 8: We show the difference between the map matching results of SMM (red) and data-driven map matching (blue) at intersections and curved roads. The bottom row displays the GPS sample points (black) and the manually matched path (green). In most cases SMM prefers "cutting the corner" while data-driven map matching tends to follow larger, popular roads, though the distance travelled along the two paths are very similar. This behavior conforms with the general observation that drivers prefer to stay on the same, faster route for as long as possible.

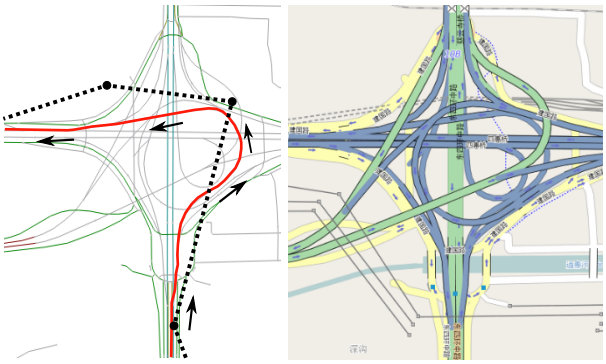


Figure 10: Examples of manually labeled ground truth paths (left) in reference to the Open Street Map image (right). Taxi trajectories and the ground truth paths are colored in black and red, respectively.

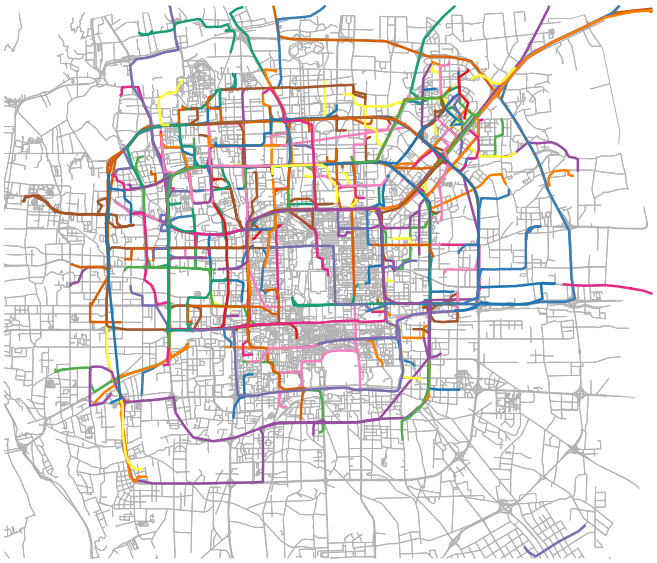


Figure 11: 200 sample trajectories from the Beijing Taxi Cab Dataset drawn on top of the road network.

5.2 Joint Segment Matching Discussion

Figure 8 and Figure 12 compare the performance of the proposed algorithm and the baseline single-track map matching algorithm. Overall, the performance of the proposed approach is better than the baseline algorithm for all tested ranges of ϵ.

When $\epsilon = 0$, $p(\epsilon)$ is the percentage of exact matched points in the projected path, which we refer as map matching accuracy. Figure 12(a) shows that the accuracy of data driven map matching is 0.86, compared to 0.77 for the baseline.

When there are multiple plausible paths between two sample points, the proposed algorithm tends to find the correct one that is agreed on by the majority of the trajectories. In contrary, the baseline algorithm determines the projected path independently, which tends to produce a less consistent result. In the following, we discuss in detail the performance of the proposed algorithm from a variety of perspectives.

Number of selected segments. The number of segments created during the segment generation stage is 7,014,231. The first iteration reduces this number by 95% to 345,617. The number of selected segments after 4 iterations is 121,756,

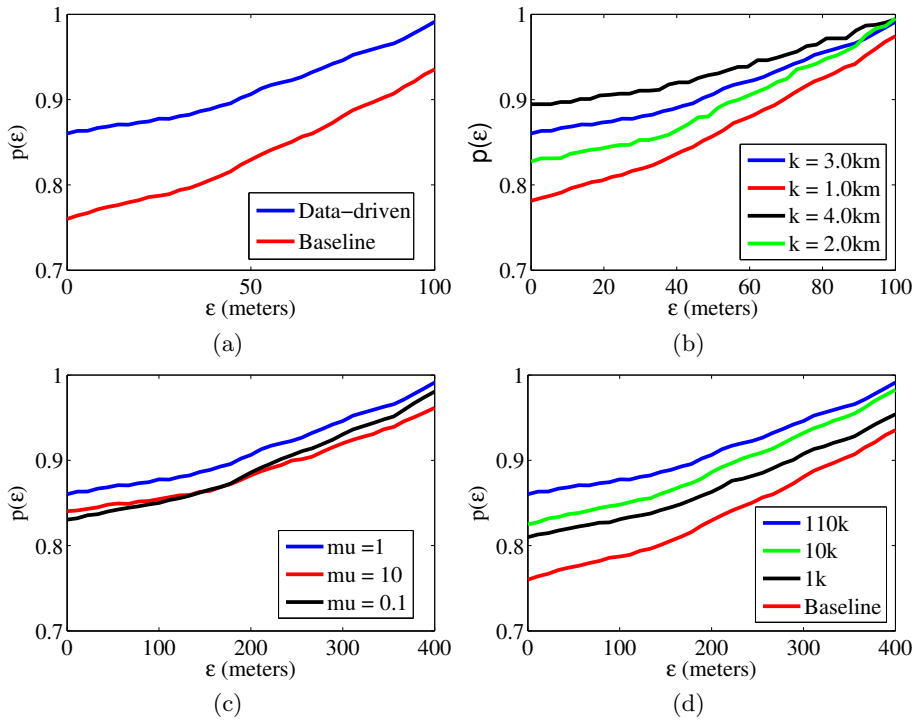


Figure 12: (a) Comparison of map matching accuracy in reference to the baseline method, with default parameters. (b) Segment order k and map matching accuracy. (c) Parameter μ and map matching accuracy (d) Number of input trajectories and map matching accuracy

1.73% of the original segment count. This verifies that the number of selected segments is indeed small.

Segment order. Figure 12(b) compares the results of setting k to 1-4 kilometers at 1km increment.We can see that using longer segments yields better results. On the other hand, the number of segments increases drastically as k increases, which leads to higher computation costs.

Coefficient μ. The parameter μ controls the data-driven effects. Figure 12(c) compares the results of setting $\mu = 0.1$ and $\mu = 10$ with the default value of $\mu = 1$. We find that when $\mu = 0.1$, the performance of our algorithm is close to that of the baseline algorithm. This is expected since the selected segments are determined for each input trajectory independently. On the other hand, when $\mu = 10$, the performance also drops. This is due to the fact of over-smoothing, i.e., where individual features of each trajectory are filtered.

Data helps. More input trajectories should help improve the quality of the extracted popular segments. We have tested our algorithm when only using 1K and 10K trajectories, as shown in Figure 12(d). In most cases, the benchmark trajectories are included in the test set. We found that the performance increases gradually with more input trajectories available.

Timing. The total running time on the whole dataset with 8 iterations is 8 hours and 21 minutes on a machine with 3.2GHz quadcore CPU. It took approximately 0.021 seconds on a single core to generate the candidate segments per trajectory point. The map matching process at each iteration takes 0.005 seconds.

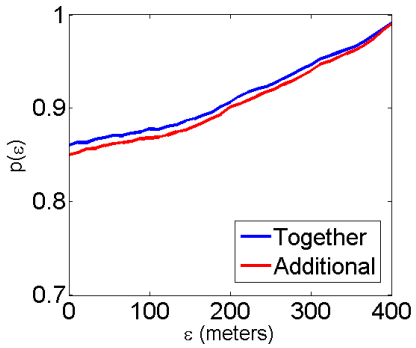


Figure 13: Additional trajectories and map matching accuracy.

5.3 Matching Additional Trajectories

Figure 13 shows the result of using the learned k-segments to match additional trajectories. In our experiment, we remove the benchmark trajectories from the input trajectory set and apply the proposed algorithm to obtain the set of selected segments as well as their optimized values of y_s. Then we perform single-track map matching for each trajectory in the benchmark set using the learned segments, as described in Section 4. Compared with the standard setting, we found that the performance of matching the labeled trajectories as additional trajectories is very similar to that of including them into training phase. Moreover, both of them are bet-

ter than the baseline algorithm. This shows the proposed algorithm has good generalization performance. Time-wise, the map matching process of one additional trajectory takes 0.03 seconds. This makes it appropriate for real-time applications.

6. CONCLUSIONS

We have presented a multi-track map matching algorithm that discovers and exploits regular structures exhibited in large collections of GPS traces. The algorithm jointly selects a set of k-segments that minimizes both the matching error and the number of unique segments used in all matchings. It then merges the selected segments into paths. We have applied the algorithm on a collection of over 120,000 taxi trajectories, and tested the matching accuracy on a manually labeled benchmark.

Here are a few important aspects that we aim to address in future work.

Temporal information. The current algorithm does not use any temporal information of the trajectories. In reality the popularity of segments can vary a lot during different times of day, and different days of a week. For instance, drivers may choose driving on the side roads during peak hours to avoid traffic, and the traffic pattern in a city may be different between weekdays and weekends.

Curve-to-point distance In Section 3 we gave a heuristic definition of the curve-to-point distance d_{curve} that works reasonably well in our tests. Nevertheless we need a more meaningful distance metric for d_{curve}, from which we theoretically justify the effectiveness of data-driven map matching. One candidate measure is the Fréchet distance that uses both position and order information of the curve.

Acknowledgments

The authors acknowledge support of ARO grant W911NF-07-2-0027, NSF grants CCF-1011228 and CCF-1161480, a Google Research award and by the Max Planck Center for Visual Computing and Communication.

7. REFERENCES

[1] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proc. 13th International conference on Ubiquitous computing (UbiComp '11)*, pages 89–98, 2011.

[2] P. S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Pervasive Computing*, volume 7319 of *Lecture Notes in Computer Science*, pages 57–72. Springer Berlin Heidelberg, 2012.

[3] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *Proc. 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pages 669–677, 2012.

[4] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 637–646, 2009.

[5] A. Javanmard, M. Haridasan, and L. Zhang. Multi-track map matching. In *Proc. 20th International Conference on Advances in Geographic Information Systems (GIS '12)*, pages 394–397, 2012.

[6] M. C. González, C. A. Hidalgo, and AL. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779 – 782, 2008.

[7] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st International Conference on Very Large Databases (VLDB '05)*, pages 853–864, 2005.

[8] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proc. 13th Meeting on Algorithm Engineering and Experiments (ALENEX '11)*, pages 75–83, 2011.

[9] P. Newson and J. Krumm. Hidden Markov map matching through noise and sparseness. In *Proc.17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*, pages 336–343, 2009.

[10] R. Raymond, T. Morimura, T. Osogami, and N. Hirosue. Map matching with hidden Markov model on sampled road network. In *Proc. 21st International Conference on Pattern Recognition (ICPR '12)*, pages 2242–2245, 2012.

[11] T. Hunter, R. Herring, P. Abbeel, and A. M. Bayen. The path inference filter: model-based low-latency map matching of probe vehicle data. *Computing Research Repository*, abs/1109.1966, 2011.

[12] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proc. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*, pages 352–361, 2009.

[13] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and GZ. Sun. An interactive-voting based map matching algorithm. In *Proc. 11th International Conference on Mobile Data Management (MDM '10)*, pages 43–52, 2010.

[14] C. White. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1-6):91–108, 2000.

[15] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. Online map-matching based on hidden Markov model for real-time traffic sensing applications. In *Proc. 15th IEEE International Conference on Intelligent Transportation Systems (ITSC '12)*, pages 776–781, 2012.

[16] S. S. Chawathe. Segment-based map matching. In *IEEE Intelligent Vehicles Symposium (IV '07)*, pages 1190–1197, 2007.

[17] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *Proc. 28th IEEE International Conference on Data Engineering (ICDE '12)*, pages 1144–1155, 2012.

[18] Taxi Trajectory Open Dataset. http://sensor.ee.tsinghua.edu.cn, Tsinghua University, Beijing, China, 2009.

[19] Open Street Map. www.openstreetmap.org.