City-Scale Map Creation and Updating using GPS Collections

Chen Chen
Stanford University
cchen86@stanford.edu

Cewu Lu
Stanford University
lucewu@stanford.edu

Qixing Huang
Toyota Technological Institute at Chicago
huangqx@ttic.edu

Qiang Yang
Hong Kong University of Science and Technology
qyang@cse.ust.hk

Dimitrios Gunopulos
University of Athens
dg@di.uoa.gr

Leonidas Guibas
Stanford University
guibas@cs.stanford.edu

ABSTRACT

Applications such as autonomous driving or real-time route recommendations require up-to-date and accurate digital maps. However, manually creating and updating such maps is too costly to meet the rising demands. As large collections of GPS trajectories become widely available, constructing and updating maps using such trajectory collections can greatly reduce the cost of such maps. Unfortunately, due to GPS noise and varying trajectory sampling rates, inferring maps from GPS trajectories can be very challenging. In this paper, we present a framework to create up-to-date maps with rich knowledge from GPS trajectory collections. Starting from an unstructured GPS point cloud, we discover road segments using novel graph-based clustering techniques with prior knowledge on road design. Based on road segments, we develop a scale- and orientation-invariant *traj-SIFT* feature to localize and recognize junctions using a supervised learning framework. Maps with rich knowledge are created based on discovered road segments and junctions. Compared to state-of-the-art methods, our approach can efficiently construct high-quality maps at city scales from large collections of GPS trajectories.

Keywords

Map construction; GPS trajectories; traj-SIFT feature

1. INTRODUCTION

The availability of accurate and up-to-date digital maps is the foundation of many location-based applications, e.g., autonomous driving, real-time route suggestion and navigation, etc. Currently, to build such maps, companies such as Google, Apple, HERE, TomTom, and OpenStreetMap, etc., are spending tremendous amount of efforts to collect data using specially equipped mapping fleets and to create maps manually from the collected data. Unfortunately, real-world road networks are subject to change due to construction, closures, accidents, etc. As a result, newly updated maps soon become outdated, and it requires significant effort to keep the maps up-to-date. For example, in 2015, HERE map reported millions of updates to their map database around the globe every day [HERE 360, 2015]. Similar statistics have also been reported by OpenStreetMap [OpenStreetMap, 2016]. The proliferation of autonomous vehicles increases demands on the timeliness and accuracy of digital maps. Clearly, manually creating and updating maps is too costly and inefficient to meet such a demand, and the development of automatic techniques to create maps that reflect daily changes of the road network is essential.

Today, GPS sensors are widely deployed in mobile platforms such as cars and smart phones. These sensors can record the moving trajectories of their host vehicles. Meanwhile, the increasing bandwidth over cellular and Wi-Fi network makes it much cheaper to transfer data from these mobile platforms. Therefore, it is possible to collect a large amount of up-to-date, sometimes even real-time trajectories. If high-quality maps can be created and maintained automatically from such GPS trajectory collections, the cost can be dramatically reduced.

Unfortunately, inferring the underlying geometries and topologies of road networks from GPS trajectories can be very challenging for two reasons: First, GPS noise and low sampling rate induces both spatial and temporal uncertainties. Spatial uncertainties come from the noise of GPS measurements [Kaplan and Hegarty, 2006]. Meanwhile, in order to minimize energy consumption and data transmission cost, trajectories recorded by mobile platforms are often sampled sparsely in time, which leads to temporal uncertainties. Second, road network is hierarchical (e.g., highway and local roads), non-planar (e.g., overhead crossings), and heterogeneous (e.g., city road network is often more complicated than rural areas) [Eppstein and Goodrich, 2008]. Therefore, there is a large knowledge gap between the map representation and raw GPS trajectories, and discovering knowledge of the map from unstructured GPS collections is a challenging and long-standing problem.

There have been many seminal works on creating and updating maps using collections of GPS trajectories. Most of the existing methods treat this problem as a clustering prob-

KDD '16, August 13–17, 2016, San Francisco, CA, USA.

DOI: http://dx.doi.org/10.1145/2939672.2939833

lem: clustering GPS points or sub-trajectories into roads, and then connecting them to produce a map. Due to the challenges mentioned above, the maps produced by existing algorithms are often poor in quality and contain many unrealistic structures.

In this paper, we present a supervised learning framework to address the aforementioned challenges. The generated maps can be used readily for regions that do not have existing maps, and enable fast updating to existing maps. The main difference between our method and previous ones is that our supervised framework allows us to leverage prior knowledge of real-world road networks, e.g., the most probable shapes of roads and junctions, into the map construction workflow. We introduce an intermediate representation, referred to as *road segment representation*, as a de-noised representation that contains both local and global information of input GPS trajectories. To generate such a representation, we develop a traj-meanshift algorithm to denoise the GPS points, and a novel graph-based road segment clustering algorithm with smoothness prior of real-world roads. For detecting and classifying junctions, we develop a traj-SIFT feature resembling SIFT feature (i.e., Scale-invariant Feature Transform) in the field of computer vision for scale and rotation invariant junction localization and classification. Our traj-SIFT feature keeps the spirit of image SIFT, but are significantly different due to the large differences between trajectory and image data. We then train a discriminative model using labeled samples from regions where we have ground-truth maps. Finally, we combine the discriminative classification score based on traj-SIFT feature and the likelihood of generative prior junction models for robust junction recognition.

The major contributions of this paper include:

- a graph-based road segment tracing algorithm to extract smooth road segments from unstructured GPS point cloud;
- the introduction of *traj-SIFT* feature to detect and classify road junctions into prior models;
- demonstration of efficient city-scale map construction from collections of GPS trajectories.

2. RELATED WORK

Map construction from GPS trajectories is an active research topic. Existing works can be classified into two categories: (1) clustering GPS points based on their local similarities and (2) clustering sub-trajectories based on path based distances. Existing methods do not take knowledge of existing maps into consideration.

Among the first category (i.e., clustering GPS points), [Edelkamp and Schrödl, 2003] applied k-mean algorithm to cluster GPS points into cluster centers, which were later connected to generate road centerlines. [Davies et al., 2006] computed 2D histogram from GPS points and produced a map using images rendered from a 2D histogram. [Biagioni and Eriksson, 2012b] adopted a similar kernel density estimation approach and used multiple density thresholds to produce the output map. The authors also tried to prune their results by projecting GPS trajectories to their output map. [Cao and Krumm, 2009] clustered GPS points by moving the points according to simulated attraction and repelling forces. Later, a map was built incrementally from the clustered GPS points. [Chen et al., 2010] presented a computational geometry based algorithm that clustered GPS points into "good" segments and links among the segments with theoretical guarantees. [Wang et al., 2015a] introduced an efficient topological method that constructed maps from local density field of GPS points. Although inferring maps from unstructured GPS points can be efficient and robust to GPS sampling rate, due to the lack of global information, this line of approaches suffer from GPS noise, leading to inaccurate structures in the output maps. In addition, they cannot handle non-planar structures such as crossover and underpass, which are common in real road networks.

In contrast to point-based clustering in a local neighborhood, methods in the second category try to cluster trajectories or sub-trajectories based on path-based similarities. For example, [Ahmed and Wenk, 2012] proposed an incremental map construction algorithm using sub-trajectory clustering based on path Fréchet distances. [Fathi and Krumm, 2010] trained a junction detector based on sub-trajectories features to detect junctions and connected them into a map. Compared to the first line of approach, sub-trajectory clustering brings global information into map construction. However, it usually requires more computation and is fragile when trajectory sampling rate is low.

Our method has significant advantages compared to state-of-the-art approaches: (i) we use an intermediate representation, i.e., road segment, to capture both local and global information in GPS trajectories; and (ii) we use prior knowledges of existing road network, such as road smoothness and junction types to infer the correct map representations.

On a broader scale, our work falls into the category of knowledge mining from spatial-temporal data. We are inspired by many seminal works in this area. For example, travel time estimation [Wang et al., 2014]; constructing popular routes [Wei et al., 2012]; temporal skeletonization on sequential data [Liu et al., 2014]; location prediction using mobility data [Wang et al., 2015b]; among others.

3. DATA AND PROBLEM STATEMENT

Raw GPS Trajectories: A GPS trajectory t is a sequence of GPS measurements $\{x_1, \cdots, x_n\}$. In this paper, we assume each GPS measurement x_i has five attributes: *latitude, longitude, timestamp, speed,* and *heading* (Table 1). *Speed* is the magnitude of the device's velocity, and *heading* is the clock-wise angle of the device's moving direction with respect to the earth's true north direction. GPS speed and heading are usually NOT included in previous works. However, we include them since most GPS systems record speed and heading, and in case of absence, they can be estimated from the trajectory data.

Figure 1 shows an example of GPS trajectories as points (left) and as line segments (right). Typically, GPS measurements are very noisy: while most of the GPS measurements are accurate within 15 meters, occasionally the noise level can exceed 50 meters. In addition to GPS noise, GPS trajectories are often sampled sparsely in time to save battery and reduce data transmission cost. As a result, consecutive GPS points in a single trajectory can be far away from each other (Fig 1 (right)). For example, when sampled at 30-second intervals, two consecutive GPS measurements for a car moving at 30 miles per hour can be 400 meters apart.

Despite of GPS noise and varying sampling rate, with the technical advances in mobile platforms, a large num-

ber of GPS trajectories can be collected with relatively low cost. For example, for our San Francisco dataset, by collecting cellphone GPS trajectories, we accumulated more than 26,000 GPS trajectories with ~ 4.5 million GPS points over a 3-month period. Large amount of data provides opportunities for map inference and updates using noisy and sparsely sampled GPS trajectories. However, it also poses significant challenges to the efficiency of our algorithms.

Table 1: Attributes of each GPS reading x_i

Attribute	Symbol	Description
Latitude	lat_i	Latitude.
Longitude	lon_i	Longitude.
Timestamp	$time_i$	UTC time when the GPS measurement was taken.
Speed	$speed_i$	The magnitude of the device's velocity.
Heading	$\mathbf{h}_i$	The clock-wise angle between the moving direction and the earth's true north.

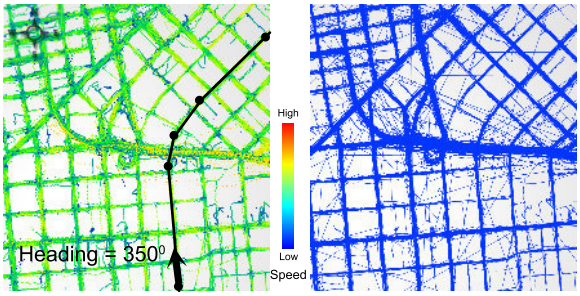

Figure 1: A collection of GPS trajectories. Left: GPS points color-coded with speeds. Right: GPS trajectories.

Trajectory Collection ($\mathcal{T}$). We use $\mathcal{T} = \{t_1, t_2, \cdots, t_m\}$ to denote a collection of GPS trajectories. For a GPS point x_i, we use $T(x_i)$ to denote its corresponding trajectory.

Map Representation ($\mathcal{G}$). A map is typically represented by a directed geometric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V}$ correspond to geo-locations in the road network and the directional edges $\mathcal{E}$ represent the connections between different vertices. Figure 2 shows an example map of a 4-way junction. Although such a graph representation is good enough for navigation, it loses high-order information such as the junction structures. For example, it is hard to tell from $\mathcal{G}$ which subset of nodes corresponds to a 4-way junction. In addition to a map represented as a graph, our method labels the junctions according to their junction types.

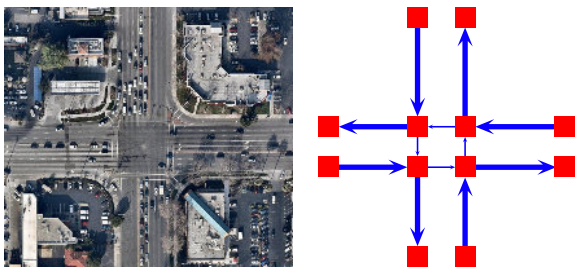

Figure 2: Map representation of a 4-way junction.

4. APPROACH

There are two major steps in our framework: (1) generate a "road segment" representation from GPS trajectories; and (2) detect and classify junctions. The junctions are later connected together to generate the output map. The following subsections present the details for each of the steps.

4.1 Road Segment Representation

Since raw GPS trajectories are noisy and redundant, it is more convenient to work with road segments extracted from raw GPS data. As an abstract representation, road segments reduce uncertainties from raw GPS data and are closer to our desired map representation. In this section, we introduce a *Road Segment Representation* which summarizes both local and global information of trajectory collections.

To trace road segments from GPS trajectories, one might think of a naïve approach that treats each trajectory as a road segment. Unfortunately, this does not work because even with very accurate GPS measurements, GPS trajectories are often sampled sparsely in time. Therefore, consecutive GPS points in a single trajectory can be far away from each other (Fig. 1 right). Such large distance gaps prevent us from using trajectories as road segments directly.

Another line of approach adopted by many existing works traces road segments by clustering unstructured GPS point cloud. The basic observation is that even though each trajectory is sparse, because a road can be traversed by many cars over a period of time, the collective GPS point cloud is usually dense. In addition, clustering GPS points from multiple trajectories in a local neighborhood also removes extreme noisy outliers. However, real road networks often have structures that are close to each other (i.e., within GPS noise range). Therefore, tracing road segments by clustering points often fail to distinguish such detailed structures.

In this work, we adopt road segment tracing from unstructured GPS point cloud. However, different from previous works, we observe that roads cannot have arbitrary curves. When designing road network, roads are often designed to be as smooth as possible to minimize sharp turns. With this prior knowledge, we introduce traj-meanshift and a graph-based clustering algorithm to trace more realistic road segments from GPS point cloud. Our algorithm encourages the samples to lie on road centerlines and generate road segments that are as smooth as possible. Finally, since tracing road segments from local GPS points loses global information from raw GPS trajectories, we add links that reflect transitions among road segments by mapping input GPS trajectories to the road segments.

4.1.1 *Traj-Meanshift Sampling*

Looking at raw GPS points, many points deviate from road centerlines due to GPS noise. Since our goal is to trace road segments, we would like to keep only the samples that lie on road centerlines as much as possible. Typically, road centerlines have higher point density as shown in Fig. 3. Hence, we design our traj-meanshift algorithm that borrows the idea of mean-shift filter to extract a subset of samples from the original GPS point cloud according to GPS location, speed and heading.

More specifically, we denote the original GPS points cloud as $\mathcal{X} = \bigcup_i \{x \mid x \in t_i, t_i \in \mathcal{T}\}$, and extract a set of samples from $\mathcal{X}$, denoted by $\mathcal{S}$, to summarize the raw GPS point cloud. Similar to a GPS point, each sample s_i has the fol-

lowing attributes: location, average speed, and heading. In addition, each sample $s_i \in \mathcal{S}$ also has a weight w_i, recording the number of nearby similar GPS points (in terms of speed and heading) covered by s_i. To obtain such $\mathcal{S}$, we generate new samples iteratively in order to "cover" the original GPS point cloud, i.e., each GPS point x_i is associated with a sample s_j in $\mathcal{S}$. At each iteration, we randomly select an uncovered GPS point x_i and search for a new sample s_i around x_i that most likely lies on a road center line. To find the location of s_i, we project nearby GPS points of x_i into a histogram that is perpendicular to x_i's moving direction, and the road center of x_i corresponds to the peak location of such 1D histogram. This 1D histogram is computed by collecting votes from x_i's neighbor points $x_j \in \Omega(x_i)$ (Fig. 3b). The voting value from each neighbor point x_j should reflect the similarity between x_i and x_j. For example, ss shown in Fig. 3b, for bin k, the vote from x_j will be computed by the following equation:

$$g(x_i, k) = \frac{\sum_{x_j \in \Omega(x_i)} I(\text{Bin}(x_j), k) e^{-\frac{\Delta^2_{\text{heading}}(x_i, x_j)}{\sigma_h^2}} \cdot e^{-\frac{\Delta^2_{\text{speed}}(x_i, x_j)}{\sigma_s^2}}}{\sum_{x_j \in \Omega(x_i)} e^{-\frac{\Delta^2_{\text{heading}}(x_i, x_j)}{\sigma_h^2}} \cdot e^{-\frac{\Delta^2_{\text{speed}}(x_i, x_j)}{\sigma_s^2}}}, \quad (1)$$

where σ_h and σ_s are two parameters and $I(\text{Bin}(x_j), k)$ is an indicator function that outputs 1 if x_j falls in k^{th} bin and 0 otherwise. The corresponding sample s_i of x_i is located at the peak location of x_i's 1D histogram with respect to x_i's heading direction. We set s_i's heading equal to $h(x_i)$, and covers consistent GPS points in $\Omega(s_i)$ and add s_i into $\mathcal{S}$.

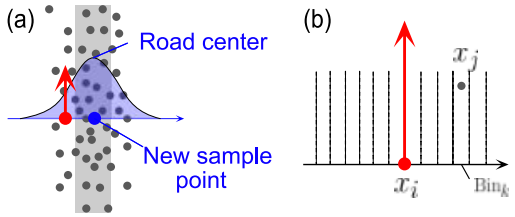

Figure 3: Illustration of GPS point cloud sampling. (a) Determine sample location around a randomly selected GPS point x_i. (b) Nearby GPS point x_j votes to the corresponding histogram bin of x_i according to Eqn. (1).

4.1.2 Graph-based Road Segments Clustering

As stated earlier, tracing road segments can be considered as clustering the samples $\mathcal{S}$. In this section we develop a graph-based clustering method which leverages the prior knowledge that road segments should be locally smooth.

With the sample set $\mathcal{S}$, we initialize an undirected graph $\mathcal{M}$ with vertices being $\mathcal{S}$. We add edges incrementally to $\mathcal{M}$ to connect the vertices into segments. Each output road segment corresponds to a connected component of $\mathcal{M}$ that is longer than a minimum length L_{min}.

The following describes how we add edges to the undirected graph $\mathcal{M}$. We process one vertex (i.e., sample point s_i) at a time in descending order according to their weights. When processing s_i, we pick two best vertices s_p and s_q near s_i and add edges (s_p, s_i) and (s_q, s_i) into $\mathcal{M}$ such that $s_p \to s_i \to s_q$ forms a smooth local segment (Fig. 4).

We pick s_p from the neighbors of s_i based on the following conditions:

- $d(s_p, s_i)$ is smaller than a threshold R
- $\Delta_{\text{heading}}(L(s_i) - L(s_p), \mathbf{h}(s_i))$ and $\Delta_{\text{heading}}(s_p, s_i)$ are both smaller than a threshold θ_h
- $\text{degree}_{\mathcal{M}}(s_p) < 2$

The first condition is to make sure s_p is close to s_i. In the second condition, $L(s_i) - L(s_p)$ is the vector from sample location s_p to s_i. Therefore, the second condition ensures vehicles to move from s_p to s_i without abrupt turning. The third condition ensures no vertex in $\mathcal{M}$ can have more than three edges such that each connected component of $\mathcal{M}$ is a line segment.

When more than one candidates of s_p are available, we pick the one with maximum score computed by the following equation:

$$score(s_p \mid s_i) = ||L(s_i), L(s)_p|| \cdot \Delta_{\text{heading}}(L(s_i) - L(s_p), s_i) \cdot \Delta_{\text{heading}}(s_p, s_i). \quad (2)$$

This scoring function favors smooth connection between s_i and s_p according to their heading directions. If there is no candidate available, no edge will be added to $\mathcal{M}$.

We use a similar procedure to find s_q, only in this case we use $\Delta_{\text{heading}}(L(s) - L(s_i), \mathbf{h}(s_i))$ in the second condition. Finally, each connected component of $\mathcal{M}$ that is longer than a minimum length L_{min} corresponds to a road segment. This algorithm can find road segments from $\mathcal{S}$ in $O(\mathcal{S})$ time.

Smoothing Road Segments. Due to GPS noise and artifacts introduced by sampling, a smoothing step is needed to make the previously extracted road segments more realistic. Since each previously extracted road segments is a line segment with multiple sample points: $\{s_1, s_2, \cdots\}$, and each sample has location and heading attributes, we use the following optimization to smooth each road segment:

$$\min_{s_i'} \sum_i \{||s_i' - s_i||^2 + \frac{1}{2}[(\text{dot}(\mathbf{n}(s_i), \mathbf{s}_i' - \mathbf{s}_{i-1}'))^2 + (\text{dot}(\mathbf{n}(s_i), \mathbf{s}_i' - \mathbf{s}_{i+1}'))^2]\},$$

where $\mathbf{n}(s_i)$ is the normal direction of s_i (i.e., rotate $\mathbf{h}(s_i)$ by 90° counter-clockwise), $(\mathbf{s}_i' - \mathbf{s}_{i-1}')$ and $(\mathbf{s}_i' - \mathbf{s}_{i+1}')$ are two vectors of adjacent samples on the road segment. This quadratic objective function has closed-form solution, and can be easily obtained by solving a small linear system.

4.1.3 Inferring Links among Road Segments

The previously extracted road segments are computed entirely base on unstructured GPS point cloud. As a result, global information, such as the transitions among road segments, is left out from the input GPS trajectories. In order to capture such high-order information, we infer links among previously extracted road segments using input GPS trajectories. More specifically, for each trajectory t_i, we search its nearby road segments $\mathcal{R}(t_i)$. From $\mathcal{R}(t_i)$, we compute a minimum sequence of road segments $r_{i_1} \to r_{i_2} \to \cdots r_{i_k}$ that "covers" t_i. As illustrated in Fig. 5a, each nearby segment covers a portion of t_i and we can select a minimum sequence of road segments that covers t_i. Computing such a minimum sequence is very similar to matching GPS trajectories to an underlying map, and one can find an efficient dynamic programming algorithm in [Goh et al., 2012].

After each trajectory is mapped to a sequence of road segments, we add a link for each pair of consecutive road

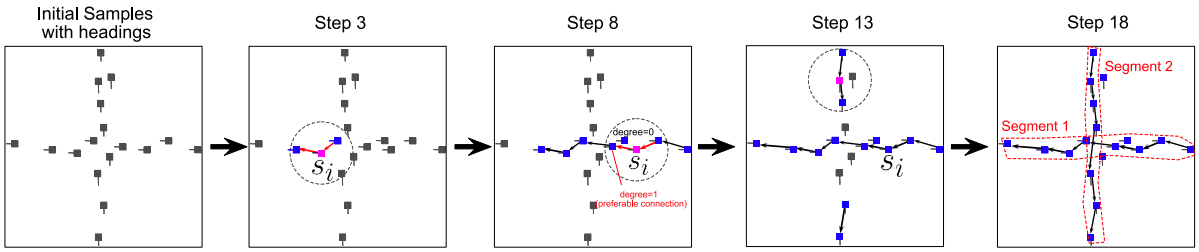

Figure 4: Illustration of our graph-based road segment tracing algorithm.

segments. For example, if t_i is mapped to $r_{i_1} \rightarrow r_{i_2} \rightarrow \cdots r_{i_k}$, we add link $l_k : r_{i_k} \rightarrow r_{i_{k+1}}$ for each consecutive pair of segments in this sequence. Collecting links from all trajectories, we get a collection of links $\mathcal{L} = \{l_1, l_2, \cdots\}$, and each link is of the form $l_i = (r_s \rightarrow r_t : w_i)$, where r_s and r_t are the source and destination road segments, and w_i is the weight of l_i, recording the number of trajectories that support this link. Figure 5b shows an example of such link collection with each link marked in red.

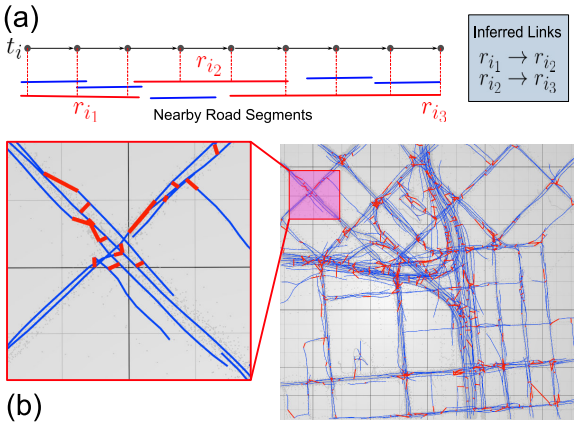

Figure 5: Computing links among road segments. (a) Mapping trajectory t_i to a minimum sequence of nearby segments ($r_{i_1} \rightarrow r_{i_2} \rightarrow r_{i_3}$). (b) The computed links among road segments.

4.2 Junction Detection and Recognition

Junction detection and recognition are crucial for road map construction. Detecting and recognizing junctions is a problem similar to object recognition in computer vision and faces similar challenges: junctions in a road network can have different size, orientation, and deformation, making them hard to be detected and classified. Inspired by the famous Scale-invariant Feature Transform (SIFT) feature [Lowe, 2004], which is widely used and proves to be robust in computer vision, we develop the trajectory SIFT (*traj-SIFT*) method that automatically detects junction scale and generates an orientation-invariant descriptor for junction classification. To make the junction classification more robust, we build a set of prior junction models for each junction class, and compute the likelihood of each junction model by fitting nearby road segments to the junction model. Our unified model combines both discriminative classification score and prior model likelihood for more accurate classification.

Setting. Most road junctions are designed in standard shapes such as Y-shape, T-shape, Cross-shape, and Star-shape. Complicated road junctions, such as local-highway junctions, can be decomposed into several of these simple junction shapes. In this paper, we use four model classes: Y-shape, T-shape, Cross-shape and Star-shape junctions. However, to adapt to their local environments, junctions often have deformed geometry, making it difficult to determine which class the junction belongs to. Fortunately, we can extract training data with ground-truth label from regions that have ground-truth map.

4.2.1 *Traj-SIFT for Discriminative Learning*

In this section we describe our novel traj-SIFT feature for junction classification. The traj-SIFT feature is extracted from the previously described intermediate representation, i.e., road segment ($\mathcal{R}$) and links ($\mathcal{L}$). Since trajectory data is different from image data, the standard SIFT is not useful in our case. Our traj-SIFT extends the idea of image SIFT to trajectory data. We observe that road segment headings resemble image gradients. The design of traj-SIFT feature uses this observation; we describe it in the following subsections.

Key Point Localization. Prior to junction recognition, we need to locate potential junction regions as key points. In image SIFT [Lowe, 2004], key points are extracted at locations with non-uniform gradient orientations, since points inside uniform gradient regions (e.g., inner points of a line or a smooth surface) do not convey discriminative information. We develop a trajectory bilateral filter to detect candidate junction locations. More specifically, for each road segment point, we compute a score that summarizes the heading differences between the road segment point and its neighbors:

$$\text{score}(r_i) = \frac{\sum_{||r_j - r_i|| \le R} \exp(-\frac{\Delta_h^2(h(r_i), h(r_j))}{\sigma_h^2}) \cdot \exp(-\frac{||r_i - r_j||^2}{\sigma_d^2})}{\sum_{||r_j - r_i|| \le R} \exp(-\frac{||r_i - r_j||^2}{\sigma_d^2})}.$$

Intuitively, this score is a weighted average of the heading differences between the selected road segment point r_i and its neighbor road segment points within radius R. Hence, potential junction locations with non-uniform heading distributions should have smaller score. We extract such locations by detecting the local minimum from the entire set of road segment points based on their scores (Fig. 6d).

Automatic Scale Estimation. As stated earlier, road junctions can have different sizes depending on their local environments. Therefore, we need to estimate the scale for each detected key point. In analogy to Difference of Gaussian (DoG) in image SIFT, we design a set of concentric bins with increasing radius (Fig. 6a) and compute the heading distribution of road segment points inside each bin. Within a junction region, the difference of the heading distributions between two consecutive bins (measured by L2-norm) should be small; and when a bin approaches the boundary of a junction, there would be a big jump in the heading distribution difference. In practice, we use $N(= 10)$ concentric bins,

and the radius of the circles increases from $R_{min}(= 50\text{m})$ to $R_{max}(= 150\text{m})$. We compute the heading histogram of road segment points inside each bin using 36 angle bins. The difference vector between consecutive heading histograms can be computed as $D(i) = \|\text{hist}(i) - \text{hist}(i-1)\|_2$. The scale of a key point can be determined as the center of the bin corresponding to the first maximum value of D (Fig. 6b, d).

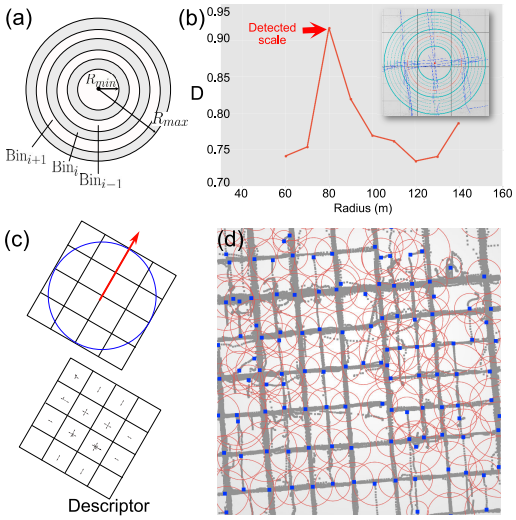

Figure 6: Automatic scale estimation using concentric bins (a) to detect scale by monitoring the heading distribution differences (b). (c) Illustration of traj-SIFT descriptor. (d) Interesting point detection using a bilateral filter. The SIFT scale for each interesting point is marked with a red circle.

traj-SIFT Descriptor. Once the scale is determined, we compute the heading distributions for all nearby road segment points within this scale. We pick the peak heading direction as the canonical direction to make extracted descriptor orientation-invariant. Similar to image SIFT feature, we divide the region near the query location into $4 \times 4 = 16$ subregions, and within each subregion, we compute heading distributions of road segment points (aligned with the canonical direction) using a 8-bin histogram (Fig. 6c). We train a multi-class SVM based on their traj-SIFT descriptors to classify them into corresponding junction classes (i.e., Y-shape, T-shape, Cross-shape and Star-shape junctions).

4.2.2 *Junction Fitting by Generative Prior*

The traj-SIFT descriptor is computed based on local road segment heading distributions. We find that it cannot effectively distinguish models that have similar shapes. For example, a T-shape junction and a Cross-shape junction may have very similar shapes, and our traj-SIFT descriptors have difficulties in distinguishing similar junctions. To make the classifier more accurate, we introduce generative prior junction models. For each junction type, we build a set of candidate models of various shapes cropped from existing OpenStreetMap as prior junction models. In our implementation, we use 20 prior models for Y-shape junctions, 10 for T-shape, 10 for Cross-shape, and 7 for Star-shape junctions (Fig. 7). Of course, more comprehensive prior junction models will lead to more accurate classification results, but will increase the computation cost due to model fitting. Given a junction key point (with estimated scale), we fit each prior junction model to the nearby road segment points. The links among nearby road segments determine the topology of the junction model. Fitting error ε is measured by the average geometric distances between road segment points and their closest junction model points. The likelihood of a prior model being the correct one is computed by $e^{-\varepsilon/\sigma_d}$, where σ_d is a parameter.

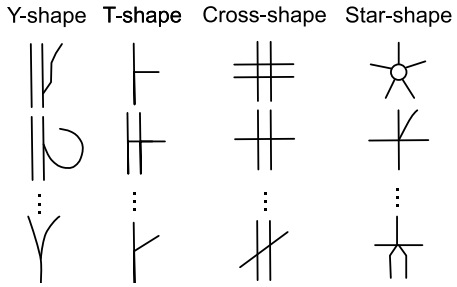

Figure 7: Prior junction models for the four junction classes.

4.2.3 *Unified Model.*

Finally, our unified model combines the score of the discriminative model (learned from training samples based on traj-SIFT descriptors) and the likelihood of prior junction model. For the discriminative model, we train a multi-class SVM based on the traj-SIFT descriptors using training samples with ground-truth junction type labels. For each junction key point, we also fit all potential prior junction models and compute the likelihood of each prior junction model. The final class of the junction key point is determined by:

$$\arg\max_i \quad \{d(J=i) + \lambda g(J=i)\}, \tag{3}$$

where i corresponds to the junction class type. $d(J=i)$ is the predicted probability from the trained multi-class SVM that current junction belongs to class i. $g(J=i) = \max_j p(J = \text{model}_{ij})$ corresponds to the maximum likelihood for all the prior models within class i. λ is a tuning parameter, which is determined by cross-validation.

5. EMPIRICAL EVALUATION

In this section, we verify the effectiveness of our method in map creation and update.

5.1 Dataset and Settings

We introduce three datasets for evaluation. The first two datasets are used for evaluating our constructed maps and comparing with previous works. The last dataset is for demonstrating the usefulness of our generated maps in updating existing maps. We use OpenStreetMap as our reference map in our evaluations due to its public availability ([OpenStreetMap, 2016]).

San Francisco Dataset. Our San Francisco dataset was collected by our industry collaborators, which consists of a large number of cellphone GPS trajectories collected when the users were using their cellphones for navigation. The dataset was collected in 2013. The average trajectory sampling rate of this dataset is $\sim 1pt/sec$. There are a total of $26,316$ trajectories, and ~ 4.5 million GPS points.

Chicago Dataset. This is a relatively small dataset with 889 bus trajectories and $117,672$ GPS points introduced by ([Biagioni and Eriksson, 2012b]). Due to the lack of speed and heading information in this dataset, we infer speed and

heading by differentiating consecutive GPS points in each GPS trajectories.

Map Update Dataset. Our constructed maps can also be used for detecting changes in the road network to update existing maps. Unfortunately, since our San Francisco dataset was collected in 2013, we couldn't detect latest road changes since 2013. However, the principle of map updating stays the same: by comparing our generated map and the current map, we can identify locations the two maps do not match. If up-to-date GPS trajectories are used for map generation, such conflation results can be used to generate map updates. Luckily, we found Google earth satellite images in San Francisco captured around the time period our trajectory was collected. We "pretend" that Google Earth to be the ground-truth map and see if our generated map can be used to detect the required changes to the OpenStreetMap we have. To do so, we overlay the most recent OpenStreetMap on top of Google earth satellite images, and identified five regions where there are inconsistencies. We randomly sampled 100 locations from each region, and manually determined whether OpenStreetMap is consistent with Google Earth at each sample location. The final map update dataset contains 500 sample locations, and each sample location has a label of 1 if OpenStreetMap is inconsistent with Google Earth satellite image, and 0 if otherwise.

5.2 Evaluation of Output Maps

In this section, we evaluate the quality of the maps generated by our framework using both San Francisco dataset and Chicago dataset.

5.2.1 *Chicago Dataset*

Since this is a small dataset, visual comparison is enough to tell the quality differences of the output maps. Figure 8a shows the result produced by [Biagioni and Eriksson, 2012b]. The authors of [Biagioni and Eriksson, 2012b] generated the map using an image-based skeleton extraction method from blurred GPS point images. As shown in Fig. 8a, due to GPS noises, the output map from [Biagioni and Eriksson, 2012b] contains many unrealistic roads and junctions. Our method produces a much more realistic map (Fig. 8b).

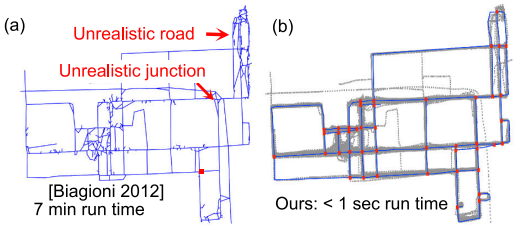

Figure 8: Comparison of our method with [Biagioni and Eriksson, 2012b] using Chicago dataset.

5.2.2 *San Francisco Dataset*

We use three metrics to evaluate the accuracy of our generated map on San Francisco dataset: (i) Precision and Recall (as proposed by [Biagioni and Eriksson, 2012a]), (ii) Trajectory Explanation Rate, and (iii) Trajectory Realization and Haustorff distances. We also give a visual comparison between our method and [Biagioni and Eriksson, 2012b] on a small region in San Francisco.

Fig. 9 shows our generated map using our San Francisco data. The map produced by our algorithm is visually close to OpenStreetMap. The only difference is that in our map, a two-way road has separable two moving directions (as reflected by GPS data). However, since OpenStreetMap is built manually, the two directions of a two-way road are usually merged as a single bi-directional edge to simplify geometry. In addition to the output map as a graph, our method also extract high-order knowledge of each junction type, which is color-coded in Fig. 9.

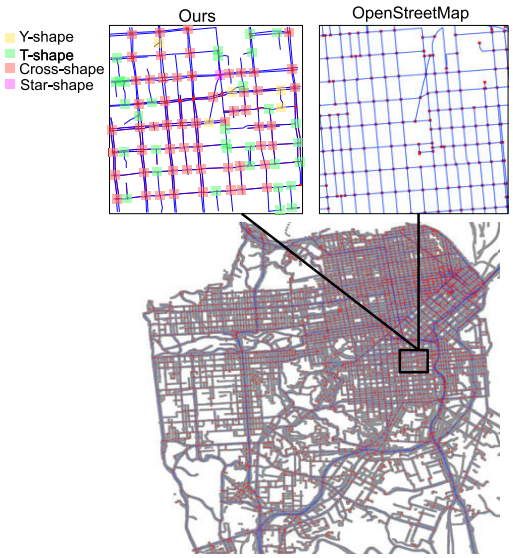

Figure 9: Example constructed maps using San Francisco dataset.

Precision and Recall. We evaluate our constructed map using the method introduced by [Biagioni and Eriksson, 2012a] to compute precision, recall and F-score. The method in [Biagioni and Eriksson, 2012a] evaluates both the topologies and geometries of the generated maps. Given our generated map $\mathcal{G}$ and the corresponding OpenStreetMap $\mathcal{G}_0$, multiple random locations are selected as starting points. At each starting point p, a DFS search from p is conducted on both $\mathcal{G}$ and $\mathcal{G}_0$. "marbles" are dropped at fixed intervals when traversing $\mathcal{G}$, and "holes" are dropped at fixed intervals when traversing $\mathcal{G}_0$ until a maximum radius r is reached. If a "marble" is close to a "hole" within a matching threshold d, the "marble" matches to the "hole." The precision, recall and F-score are defined as follows:

$$precision = \frac{\text{matched marbles}}{\text{total marbles}}$$

$$recall = \frac{\text{matched holes}}{\text{total holes}}$$

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

We randomly pick 1% of the GPS points as the starting locations. We fix $r = 100m$ and vary d from $5m$ to $30m$. Fig. 10 shows the results on our San Francisco dataset. The high precision and recall score proves that our generated maps are very close to the reference OpenStreetMap.

Trajectory Explanation Rate. Since our output map is generated by a collection of input GPS trajectories, we expect to be able to project each input trajectory to a path on our generated map. However, in real scenarios, our methods could fail to generate road segments for roads that are sparsely traversed. Our method may also fail to infer connections at a junction region when the corresponding turnings

are rarely traversed. As a result, we may have input trajectories that could not be projected to paths on our generated map. To evaluate the coverage of our map on input trajectories, we define *Trajectory Explanation Rate* as the percentage of input GPS trajectories that can be projected as paths on a reference map. We use a simple online map-matching algorithm in [Goh et al., 2012] to project the input trajectories to a map. We compare the trajectory explanation rates of our map to OpenStreetMap. As shown in Fig. 11, on our San Francisco dataset, our constructed map can explain $\sim 89\%$ of the input trajectories.

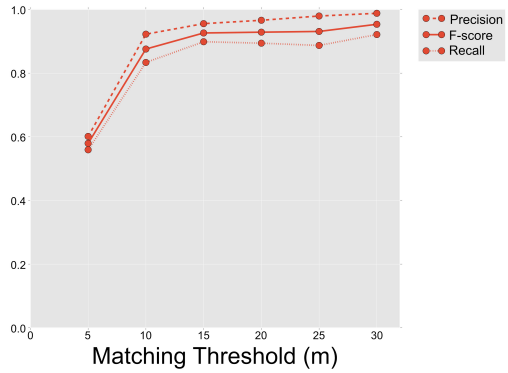

Figure 10: Precision / recall and F-score.

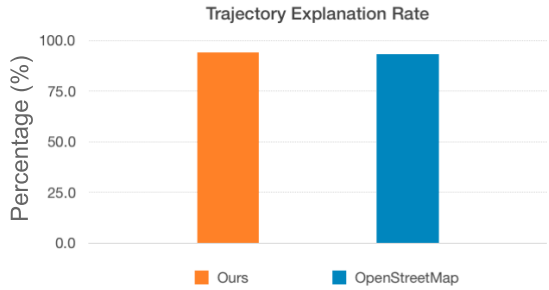

Figure 11: Trajectory explanation rates on San Francisco dataset.

Trajectory Realization and Haustorff distances. We are aware of other path-based map comparison methods. For example, comparing Hausdorff distances between shortest paths among randomly selected source and destination location pairs suggested by[Ahmed et al., 2014]. However, since we are only interested in the map structures that are traversed by GPS trajectories, we compare the Hausdorff distance between map-matched GPS trajectories to our map and to the reference OpenStreetMap. More specifically, for each input GPS trajectory t_i, we project it to both our map and OpenStreetMap as two corresponding paths $p_G(t_i)$ and $p_{G_0}(t_i)$. Hausdorff distances between $p_G(t_i)$ and $p_{G_0}(t_i)$ is computed by

$$d_H(p_G(t_i), p_{G_0}(t_i)) = \max\{\sup_{x \in p_G(t_i)} \inf_{y \in p_{G_0}} d(x,y), \sup_{x \in p_{G_0}(t_i)} \inf_{y \in p_G} d(x,y)\}.$$

We only consider trajectories that can be explained by both maps (which corresponds to $\sim 80\%$ of the input trajectories). The comparison results are shown in Fig. 12. For those trajectories that can be explained by both maps, the Hausdorff distances between their map-matched paths to our map and OpenStreetMap are less than 50 meters, which are smaller than the majority GPS noise level.

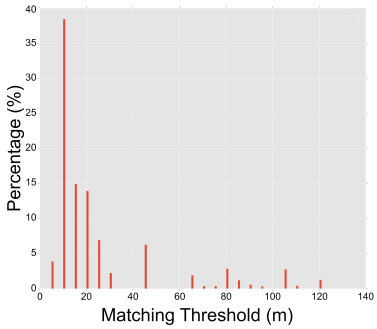

Figure 12: Distribution of Hausdorff distances between map-matched paths to our map and to the reference OpenStreetMap.

Visual Comparison with Previous Works. We select a small region in San Francisco, and compare our result with that produced by [Biagioni and Eriksson, 2012b] (Fig. 13). Since this is a larger region with more complicated road structures compared to Chicago dataset, there are more unrealistic structures in the output map using method from [Biagioni and Eriksson, 2012b]. Our method is stable and generates high-quality results.

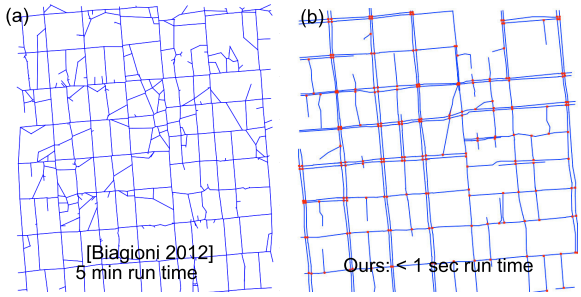

Figure 13: Comparison of our method with [Biagioni and Eriksson, 2012b] in a small region in San Francisco.

5.2.3 *Map Update Dataset*

As described earlier, our map update dataset contains 500 sample locations with ground-truth being Google Earth. For those locations where OpenStreetMap is inconsistent with Google Earth, we check if our generated map covered the desired updates. Overall, we find that 95% of the inconsistent locations are covered by our generated map. Figure 14 shows two examples of such cases along with the ground-truth satellite images from Google Earth.

5.3 Analysis

We provide in-depth analysis of our method in the following sections. We provide analysis on the effectiveness of our traj-SIFT feature and our unified junction classification model. The stability of our framework with respect to GPS sampling rate and input data is also included.

5.3.1 *traj-SIFT Feature Analysis*

We extracted ~ 300 junctions in the city of San Francisco with ground-truth junction labels. At each junction location, a traj-SIFT feature is extracted from the road segment representation computed from input GPS trajectories.

Without Prior Model Likelihood. We first test the classification accuracy using only the 128-dimensional traj-

SIFT descriptors. More specifically, we use 70% of the samples to train a multi-class SVM classifier [Chang and Lin, 2011]. The remaining 30% samples are used for testing. Table 2 shows the confusion matrix of the multi-class SVM on testing data. The SVM sometimes mis-classifies T-shape to Cross-shape junctions, and Star-shape to Cross-shape junctions. This is because such structures sometimes are difficult to separate when having similar shapes.

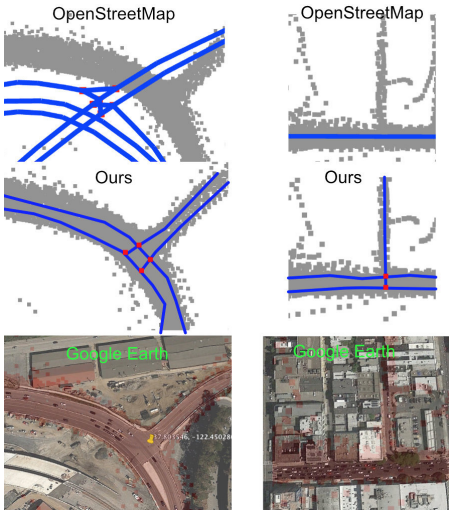

Figure 14: Identified regions where OpenStreetMap does not agree with our generated map. The satellite image aligned with GPS points are shown at the bottom.

Table 2: Multi-class SVM Confusion matrix using only traj-SIFT descriptor.

		Prediction			
		Y	T	Cross	Star
Truth	Y	**0.86**	0.09	0.05	0.0
	T	0.19	**0.34**	0.44	0.03
	Cross	0.03	0.18	**0.78**	0.01
	Star	0.13	0.0	0.47	**0.4**

With Prior Model Likelihood. In addition to the discriminative SVM prediction score, our unified model combines the discriminative prediction score with the likelihood of each prior junction model (Eqn. 3). As shown in Table 3, such an unified model performs much better than using only a multi-class SVM.

Table 3: Confusion Matrix of our unified model (Eqn. 3)

		Prediction			
		Y	T	Cross	Star
Truth	Y	**0.91**	0.09	0.0	0.0
	T	0.08	**0.86**	0.06	0.0
	Cross	0.0	0.0	**0.98**	0.02
	Star	0.0	0.0	0.09	**0.91**

Comparison with Image SIFT. We introduce traj-SIFT feature for junction localization and prediction based on insights from image SIFT feature. An naive approach is to render the GPS trajectories as blurred images, and extract image SIFT feature directly to classify junctions. Table 4 shows the confusion matrix using such an approach. The results are much worse than using our traj-SIFT features.

Table 4: Multi-class SVM confusion matrix using naive image SIFT descriptors.

		Prediction			
		Y	T	Cross	Star
Truth	Y	**0.40**	0.34	0.16	0.10
	T	0.47	**0.26**	0.15	0.12
	Cross	0.23	0.11	**0.37**	0.29
	Star	0.18	0.07	0.27	**0.48**

5.3.2 *Stability Analysis*

Sensitivity to Input Data. To test the stability of our framework under different input data, we extracted a small region from our San Francisco dataset. We further divided the trajectories into two part, and Fig. 15 shows the output maps using only half of the data. As we can see from Fig. 15, the majority of the map structures are stable under different input data. However, sparsely traversed roads are not stable since they may not be covered using just part of the data. If a road is sufficiently covered, our method produce consistent results under different input.

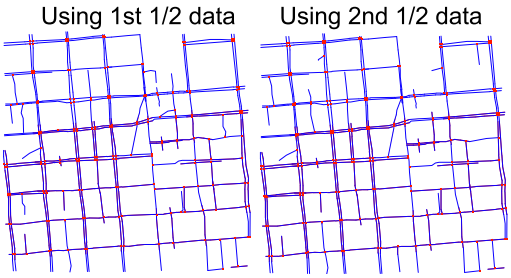

Figure 15: Stability test using different sub-parts of the original input data.

Sensitivity to Sampling Rate. Another type of uncertainties comes from the varying sampling rate of input trajectories. We pick a small 4-way junction, and sub-sample the original trajectories to generate lower sampling rate trajectories. Figure 16 shows the results of using trajectories with different sampling rates. Since we are only subsampling the original GPS trajectories, the total number of GPS points remains the same. Therefore, the main junction structures are stable, as our road segment tracing algorithm mainly relies on GPS points. However, when sampling rate drops to a very low level, we cannot identify some connections in the junction. In the extreme case (Fig 16d), the input is virtually a GPS point cloud. Therefore, our algorithm inferred a non-planar crossing for this junction.

5.4 Running Time

We test our algorithms on a Macbook Pro 2013 with 2.4GHz Intel Core i7 and 8GB memory. Table 5 shows the running time of our algorithm on the San Francisco dataset. The total running time is less than 5 minutes. With optimizations such as parallel computing, we believe our workflow can speed up to near real-time.

6. CONCLUSION

In this paper, we present an efficient framework to create maps form GPS collections. The generated maps can be used in regions that do not have existing maps. They are also helpful in identifying changes and generating updates to existing map database. The key idea of our method is to leverage prior knowledge on the structure of a road net-

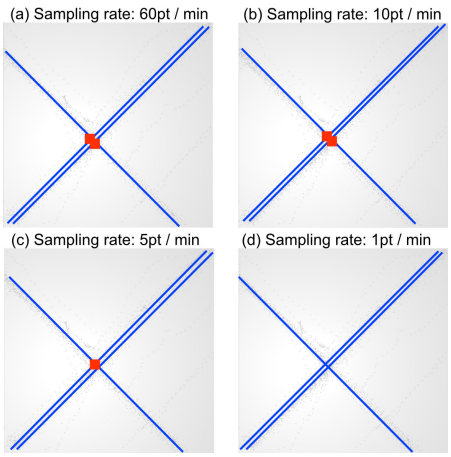

Figure 16: Sensitivity analysis with varying sampling rate.

Table 5: Running Time on San Francisco Dataset ($4,474,167$ GPS points).

Description	Value
#GPS Points	$4,474,167$
#Trajectories	$26,316$
Compute road segment representation	49.7 sec
Interesting point detection	< 2 sec
Junction classification and model fitting	< 3 min
Total running time	< 5 min

work to minimize uncertainties from GPS trajectories. We introduce a novel graph-based algorithm to trace road segments from GPS trajectories as an intermediate representation. Such an intermediate representation captures both local and global information of noisy input GPS trajectories. To detect and classify junctions, we design a scale- and orientation-invariant *traj-SIFT* feature, and introduce an unified classification model that combines the discriminative prediction score and the likelihood of prior junction models. Compared to existing state-of-the-art methods, our method produces high-quality maps, and is scalable to large city-scale dataset with millions of GPS points.

7. ACKNOWLEDGEMENTS

The US authors acknowledge the support of of NSF grants DMS-1228304 and CCF-1514305, a Google Research Award, and the Max Planck Center for Visual Computing and Communication. We also acknowledge support to D. Gunopulos from the European Union Horizon 2020 Programme (Horizon2020/2014-2020), under grant agreement VaVeL n° 688380 and by the INSIGHT FP7-318225 EU funded project.

8. REFERENCES

[Ahmed et al., 2014] Ahmed, M., Karagiorgou, S., Pfoser, D., and Wenk, C. (2014). A comparison and evaluation of map construction algorithms. *CoRR*, abs/1402.5138.

[Ahmed and Wenk, 2012] Ahmed, M. and Wenk, C. (2012). Constructing street networks from gps trajectories. In Epstein, L. and Ferragina, P., editors, *Algorithms ESA 2012*, pages 60–71.

[Biagioni and Eriksson, 2012a] Biagioni, J. and Eriksson, J. (2012a). Inferring road maps from gps traces: Survey and comparative evaluation. In *Transportation Research Board, 91st Annual*.

[Biagioni and Eriksson, 2012b] Biagioni, J. and Eriksson, J. (2012b). Map inference in the face of noise and disparity. ACM SIGSPATIAL '12, pages 79–88.

[Cao and Krumm, 2009] Cao, L. and Krumm, J. (2009). From gps traces to a routable road map. ACM SIGSPATIAL '09, pages 3–12.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*.

[Chen et al., 2010] Chen, D., Guibas, L. J., Hershberger, J., and Sun, J. (2010). Road network reconstruction for organizing paths. SODA '10, pages 1309–1320.

[Davies et al., 2006] Davies, J., Beresford, A., and Hopper, A. (2006). Scalable, distributed, real-time map generation. *Pervasive Computing, IEEE*, 5(4):47–54.

[Edelkamp and Schrödl, 2003] Edelkamp, S. and Schrödl, S. (2003). *Computer Science in Perspective: Essays Dedicated to Thomas Ottmann*, pages 128–151. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Eppstein and Goodrich, 2008] Eppstein, D. and Goodrich, M. T. (2008). Studying (non-planar) road networks through an algorithmic lens. ACM SIGSPATIAL '08.

[Fathi and Krumm, 2010] Fathi, A. and Krumm, J. (2010). Detecting road intersections from gps traces. GIScience'10, pages 56–69, Berlin, Heidelberg.

[Goh et al., 2012] Goh, C., Dauwels, J., Mitrovic, N., Asif, M., Oran, A., and Jaillet, P. (2012). Online map-matching based on hidden markov model for real-time traffic sensing applications. In *ITSC*, pages 776–781.

[HERE 360, 2015] HERE 360 (2015). the official here blog. http://360.here.com/2015/06/04/what-do-android-ios-and-wp-here-maps-update/.

[Kaplan and Hegarty, 2006] Kaplan, E. and Hegarty, C. (2006). *Understanding GPS: Principles and Applications*. Artech House mobile communications series.

[Liu et al., 2014] Liu, C., Zhang, K., Xiong, H., Jiang, G., and Yang, Q. (2014). Temporal skeletonization on sequential data: Patterns, categorization, and visualization. KDD '14, pages 1336–1345.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.

[OpenStreetMap, 2016] OpenStreetMap (2016). http://openstreetmap.org.

[Wang et al., 2015a] Wang, S., Wang, Y., and Li, Y. (2015a). Efficient map reconstruction and augmentation via topological methods. In *Proceedings of the 23th International Conference on Advances in Geographic Information Systems*.

[Wang et al., 2015b] Wang, Y., Yuan, N. J., Lian, D., Xu, L., Xie, X., Chen, E., and Rui, Y. (2015b). Regularity and conformity: Location prediction using heterogeneous mobility data. KDD '15, pages 1275–1284. ACM.

[Wang et al., 2014] Wang, Y., Zheng, Y., and Xue, Y. (2014). Travel time estimation of a path using sparse trajectories. In *KDD '14*. ACM.

[Wei et al., 2012] Wei, L.-Y., Zheng, Y., and Peng, W.-C. (2012). Constructing popular routes from uncertain trajectories. KDD '12. ACM.