Mining Large-Scale, Sparse GPS Traces for Map Inference: Comparison of Approaches

Xuemei Liu*
Shanghai Jiao Tong University
Shanghai, China
xuemeiliu@sjtu.edu.cn

James Biagioni Jakob Eriksson
University of Illinois at Chicago
Chicago, IL, USA
{jbiagi1,jakob}@uic.edu

Yin Wang George Forman
Hewlett-Packard Labs
Palo Alto, CA, USA
{yin.wang,george.forman}@hp.com

Yanmin Zhu
Shanghai Jiao Tong University
Shanghai, China
yzhu@sjtu.edu.cn

ABSTRACT

We address the problem of inferring road maps from large-scale GPS traces that have relatively low resolution and sampling frequency. Unlike past published work that requires high-resolution traces with dense sampling, we focus on situations with coarse granularity data, such as that obtained from thousands of taxis in Shanghai, which transmit their location as seldom as once per minute. Such data sources can be made available inexpensively as byproducts of existing processes, rather than having to drive every road with high-quality GPS instrumentation just for map building—and having to re-drive roads for periodic updates. Although the challenges in using opportunistic probe data are significant, successful mining algorithms could potentially enable the creation of continuously updated maps at very low cost.

In this paper, we compare representative algorithms from two approaches: working with individual reported locations vs. segments between consecutive locations. We assess their trade-offs and effectiveness in both qualitative and quantitative comparisons for regions of Shanghai and Chicago.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design methodology; H.2.8 [**Database Management**]: Applications—*Data mining; Spatial databases and GIS*

General Terms

Algorithms, Experimentation, Performance

Keywords

spatial data mining, GPS, map inference, road maps

*The work was performed while visiting HP Labs, Palo Alto.

KDD'12, August 12–16, 2012, Beijing, China.

1. INTRODUCTION

Accurate road maps are crucial for travel efficiency and safety. Existing maps are mostly drawn from geographical surveys. Such maps are updated infrequently due to the high cost of surveys, and thus lag behind road construction substantially. This problem is most prominent in developing countries, where there is often a combination of inferior survey quality as well as booming growth. Even for developed countries, frequent road reconfigurations and closures often confuse GPS navigation, and have caused fatal accidents even with experienced drivers [4]. As an alternative, OpenStreetMap (OSM) has attracted an online community to manually draw street maps from aerial images and collected GPS traces [2]. Though less expensive, this volunteer-driven process is extremely labor intensive. It also suffers from significant variability in both the skill level and availability of volunteers in different parts of the world.

The past decade has witnessed considerable interest in building road maps automatically from GPS traces. Given sequences of GPS coordinates from instrumented probe vehicles, the problem is to extract the roads covered by these probes. Existing methods can be largely divided into three categories [7]: *k-means*, which finds clusters along roads and links their centers to produce the road centerlines [13, 21, 25, 5]; *kernel density estimation* (KDE), which builds a 3D density map from traces and uses its contour lines to calculate the road centerlines [12, 10, 22]; and *trace merging*, which clusters traces by roads and approximates their centerlines using various fitting methods [9, 20, 26]. Recently, the accuracy of these three categories of methods were compared using GPS traces from shuttle buses in Chicago [7]. The above methods were largely developed for and evaluated against GPS data that is sampled with high frequency, e.g. every second. Where a slightly slower sampling rate was used, e.g. every ten seconds, probe data was only used to update an existing base map [21, 26]. Further, the traces studied typically have good accuracy, with a standard deviation less than five meters [9]. In this paper, we use opportunistically collected GPS traces with coarser recorded precision and sampling.

With the ever increasing use of GPS-enabled smart phones and telematics systems, vehicles equipped with these devices cover road networks around the globe. Often large-scale GPS traces are readily available from these vehicles. For

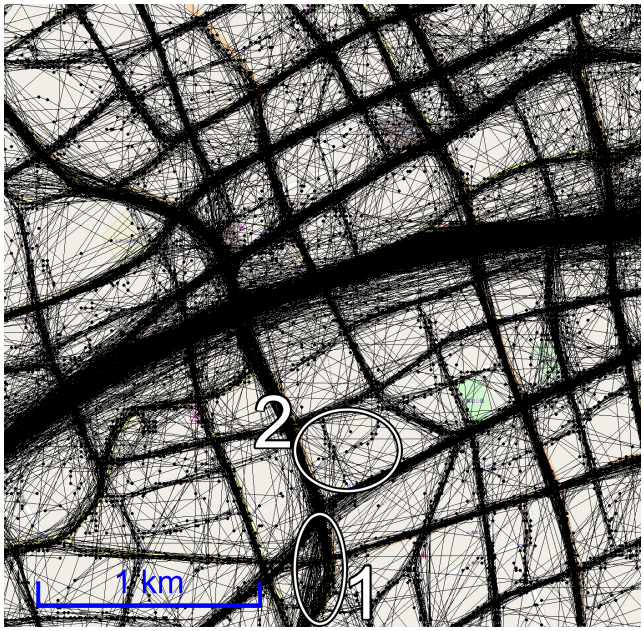


Figure 1: Sample raw input data from a dataset of 2,300 taxis in Shanghai emitting GPS samples every 16 to 61 seconds for six hours. Lines connect consecutive samples, represented by small dots.

example, taxi dispatching that utilizes GPS technology has been around for more than a decade [15], and real-time taxi traces are available in many major cities [16, 8, 6, 24]. These traces enable an unprecedented opportunity to build and dynamically update large-scale maps.

However, the nature of these GPS traces differs significantly from those used in the literature. Most importantly, due to bandwidth and storage cost concerns, sampling rates much lower than 1 Hz are common [18, 24]. In addition, precision and accuracy may sometimes be degraded due to further cost and bandwidth conserving measures. Although the much larger volume of probe data may well compensate for these limitations, none of the above-mentioned algorithms are designed for this type of data. For example, a crucial design tradeoff in map building is whether to use probe coordinates alone [13, 21, 25, 10, 22, 5] or the edges that connect consecutive probe samples [12, 20, 9]. Using individual coordinates alone, sparsely sampled roads may be fragmented or missed entirely. Using segments, one may recover sparsely sampled roads. Here, the drawback is that segments may pass over areas where there is no actual road. For example, two samples taken on either side of a right-angle turn may suggest a non-existent diagonal street.

Fig. 1 illustrates this tradeoff with sparse taxi data from Shanghai [3], overlaid on top of OSM map tiles. We use six hours of data from 8AM to 2PM on Feb. 18, 2007. There are 2,300 taxis reporting their locations. The sampling rate is 16 seconds when vacant, and 61 seconds when occupied [24]. The area shown in the figure is between latitude [31.21, 31.235] and longitude [121.43, 121.46]. We draw each sample as a black dot, and connect consecutive samples by line segments. For better clarity, the figure does not include samples whose straight-line speed exceeds 180 km/h. Most dots and lines are well aligned with roads. The thick horizontal curve in the center corresponds to an elevated freeway. Its greater thickness is due to several factors, including road width, number of samples, and lines connecting these samples that widen the curve inward. The latter issue is most prominent in area 1. Area 2 illustrates the problem with uneven distribution of samples. There are two local roads that intersect orthogonally inside the area, and which are covered by samples with a density much lower than major roads, almost on par with noise density. We cannot visually detect such roads without connecting consecutive samples or obtaining a great deal more data.

In light of these issues, we investigate two new techniques for map inference from sparse data. First, the existing KDE-based algorithm by Davies et al. [12] was previously shown to be robust against high GPS noise [7]. We describe a sparse-data adaptation of this algorithm. Second, we evaluate TC1 [17], a map inference method specifically designed for sparse data. For completeness, we also include the less competitive K-means algorithm [13] in our evaluation.

Below we offer a qualitative and quantitative comparison of these map inference methods on sparse data. As a first step, we use densely sampled trace data from shuttle buses at the University of Illinois at Chicago, together with a hand-verified ground truth map. Using this data we study the performance of the three algorithms for varying sampling intervals, amounts of data, and control parameters, to learn about algorithm characteristics and parameter sensitivity.

However, this dataset is relatively small both in terms of the number of probe vehicles and the number of road segments traversed, as well as being a somewhat controlled environment. For a large-scale evaluation, we use up to one month's worth of opportunistically collected GPS traces from 2,300 taxis in Shanghai. We tune each algorithm on a training region of Shanghai and evaluate its performance on a separate test region to get a picture of the overall map generation performance of these algorithms.

The main contribution of this paper is to provide the first evaluation of existing map building methods on sparse, passively collected data. We also contribute a simple variant of KDE-based map inference for sparse GPS traces.

The remainder of this paper is structured as follows. Section 2 provides background and related work, summarizing the three main categories of map building algorithms. Section 3 describes the key algorithms evaluated here. Section 4 describes the evaluation methodology, with the results and discussion of findings in Section 5. Finally, Section 6 concludes with an eye toward future work.

2. RELATED WORK

In a recent survey of the existing literature [7], three categories of map inference algorithms are identified:

K-means style algorithms begin by performing a traditional k-means style clustering of the GPS samples, where the distance measure may involve both the geometric distance to the cluster center, as well as its GPS bearing. A second phase links appropriate cluster centers to find roads, based on the traces that pass through the clusters. Representative algorithms in this class include work by Edelkamp and Schrödl [13], Schrödl et al. [21], Worrall and Nebot [25], and Agamennoni et al. [5].

Kernel density estimation (KDE) methods begin by transforming the individual samples or traces into a discretized image representing the density of samples or segments at each pixel. Using a binary threshold, they produce

(a) Trace segment selection (b) Clustering (c) Y-split separation

Figure 2: The three steps of the trace clustering algorithm, TC1.

a binary image of the roads in the region, and proceed to find road centerlines by a variety of methods, e.g. via Voronoi partitioning. Algorithms in this class include work by Davies et al. [12], Chen and Cheng [10], Shi et al.[22], and Steiner and Leonhardt [23].

Trace merging algorithms generally accumulate traces into a growing road network, where each addition meets the location and bearing constraints of existing roads so far. A post-processing phase removes relatively unsupported road findings. Representative algorithms in this class include works by Cao and Krumm [9], and Niehofer et al. [20].

In addition to the above references, Zhang et al. [26] use GPS traces to continuously refine existing road maps, with a method similar to trace merging. With assumptions on GPS sample coverage and error bounds, as well as road geometry, Chen et al. [11] formulates road inference as a geometric problem. They propose a map inference algorithm with accuracy guarantees based on finding shared sub-paths from the input traces.

Experiments using controlled GPS traces are common, e.g., a probe vehicle following a certain route. The GPS data produced by these probes is typically of high quality, e.g., with a standard deviation of 4.07 m [19] or 3.3 m [14], and sampled at 1 Hz. By contrast, we address the sparse, low-quality data currently available in large quantities.

3. MAP INFERENCE USING SPARSE DATA

Until now, the map inference literature has focused primarily on dense trace data. In this section, we describe two map inference methods for sparse trace data. One is a modification to an existing KDE algorithm [12]. The other is a new trace merging algorithm [17], designed specifically for sparse data. Finally, we briefly discuss problems encountered when applying dense-data algorithms to sparse data.

3.1 Kernel Density Estimation

We implement two variants of the KDE map inferencing algorithm in [12]: point-based KDE and segment-based KDE. In both cases, the map is represented by a fine grid of cells. A 2-dimensional histogram is produced by counting, in the case of point-based KDE, how many samples are within each cell, or in the case of line-based KDE, how many segments pass through each cell. Here, the line-based KDE is the original method as proposed in [12], and the point-based method is our modified version targeted at sparse data.

The histogram produced above is then convolved with a Gaussian smoothing function to produce a density map. With a small enough cell size, our discretized density map closely approximates the real density. Next we find the contour line based on a given threshold, and the road centerlines are extracted from the Voronoi diagram of sample points evenly distributed along the contour line. A post-processing step removes centerlines outside the contour, as well as any dead ends or "spurs" in the map.

Besides the choice of using either points or line segments, the fundamental control parameter is the global threshold used to find the contour lines. A high threshold may miss roads with few samples, and a low threshold may produce spurious roads due to noise. This problem is prominent with the taxi data, illustrated by Fig. 1. We evaluate these design choices in detail in Section 5. To ensure portability of thresholds between datasets, we express thresholds in terms of density percentiles, rather than absolute values. For example, grid cells with a density above the 80th percentile may be considered roads, independent of the size of the map, or number of sample points.

3.2 Trace Clustering Algorithm (TC1)

For sparse GPS traces, the straight line between successive samples does not necessarily correspond to the actual road driven. Consider a car turning a corner. With one sample well before the turn, and one sample well after the turn, the straight line between samples would be likely to cut straight through a building. There is a natural precision-recall tradeoff between using the sample locations alone vs. also using the line segments of the trace paths: If an algorithm uses the samples alone, it risks having lower recall for roads less traveled. For example, consider area 2 labeled in Fig. 1. Such a road may be better detected by an algorithm that takes into account the line segment between samples. However, due to turns and curvy roads, this is likely to result in lower precision, as illustrated by area 1 of Fig. 1.

To explore the use of line segments for sparse GPS traces, we have previously designed a trace clustering algorithm, TC1 [17]. It operates in four main steps, as described below. Prior to this, all data is pre-processed to discard segments over 1.5 km long or that imply more than 120 km/h in straight-line speed.

Step 1: Segment Selection. As a first step, segments that are likely to fall on an actual road, as opposed to across a building, are selected using a simple heuristic. Intuitively, for a given pair of GPS points (a, b), if a has roughly the same bearing as b and is consistent with the segment's orientation, then the vehicle likely followed the straight line from a to b. Fig. 2a demonstrates the effect of segment selection on the dataset illustrated in Fig. 1.

Step 2: Clustering. After segment selection, the remaining segments are clustered using a method based on single-linkage clustering. Initially, each segment represents its own cluster, and clusters are iteratively merged that are highly similar in both orientation and geographic distance. Two clusters A and B are merged if there exists a pair of segments $a \in A, b \in B$ such that their difference in orientation and geographic distance are below their corresponding thresholds. The segments of Fig. 2a have been replaced by their end-points in Fig. 2b, and clustered (represented by their color). After this step, segments are no longer used, and therefore we display each cluster by samples alone.

Step 3: Y-split separation. Step 2 performs well for orthogonal roads, but for roads that have Y-splits of a sufficiently narrow angle, multiple roads become merged into one cluster. To separate these Y-splits, the major axis of each cluster is traversed, building a polyline incrementally with nodes at least 100m apart, selected so that pairs of adjacent segments do not turn excessively. The process is repeated with any remaining samples of the original cluster. For example, the big cluster that covers the horizontal freeway in Fig. 1, shown in blue (navy) in Fig. 2b, is separated into several sub-clusters representing distinct roads, as shown by different colors in Fig. 2c.

Step 4: Centerline fitting. The centerline of each road is computed using *B-spline fitting* [21]. The end result is that the road centerlines curve naturally to fit the actual GPS probe points, which we display later in Fig. 9c. Similar to the contour threshold used by KDE, a global threshold determines the minimum support needed to fit a cluster, called the *support threshold* hereafter.

3.3 Other Map Inference Algorithms

Map inference algorithms designed for high-density, high-accuracy data typically work poorly on low-density or low-accuracy data. For example, our implementation of a K-means algorithm [13] and a trace merging algorithm [9] produce numerous spurious roads when applied to high-error samples obtained near high-rise buildings [7].

These algorithms perform even worse when both errors and sampling intervals are high. For example, we applied the K-means algorithm described in [13] to our Shanghai taxi data set. The clustering step is able to locate many clusters near road centers after several iterations, but the topology discovery step produces an almost completely connected graph among all cluster centers. This is because a vehicle may travel through multiple clusters between two sparse samples, and the algorithm simply connects the two end points. Finding the correct sequence of clusters it passed through is non-trivial.

4. EVALUATION FRAMEWORK

To evaluate the quality of a result from a road inference algorithm, we perform both qualitative evaluation as well as quantitative measurements of performance. So far, the literature has largely relied on the former ("eyeball" tests) for a variety of reasons: it is simple to draw the inferred roads on top of a ground-truth background image, it does not require having a ground-truth map that accurately captures the road centerlines, and it gives a good, high level perspective that may not be captured by various evaluation methods (or these may not capture salient differences in an intuitive way). But, going forward it will be increasingly important to also perform a quantitative measurement of performance to enable: (a) measurement of progress in the field, (b) comparison of many algorithms in one plot, and (c) use of machine learning techniques to derive parameter choices from training data. As ground-truth data, we use manually verified portions of OpenStreetMap.

We perform our quantitative evaluation by measuring the precision and recall of each inferred road map M with respect to the ground truth map $Truth$. To do this, we determine the *true positive length* $tp = M \bigcap Truth$, as a measure of common road length, which we define shortly. Then we compute

$$recall = \frac{tp}{||Truth||}, \quad precision = \frac{tp}{||M||}$$

where $||\cdot||$ measures the total length of the roads in the set. As is ubiquitous in information retrieval evaluation, we combine these two measurements into a single, scalar value by computing their harmonic average:

$$F\text{-}measure = \frac{2 \times precision \times recall}{precision + recall}$$

F-measure has a nice *AND*-like property: it only gives a high score when both precision and recall are high.

It remains only to explain the computation of the intersection $M \bigcap Truth$. We walk each ground-truth line segment $t \in Truth$ taking samples at every meter along t to determine what fraction of those samples are within a perpendicular distance threshold m to a (nearest) segment $t' \in M$, such that the orientation difference between t and t' does not exceed 60°. The angle restriction avoids matching portions of roads that cross at intersections, but allows for substantial misalignment of the inferred road. (This wide angle margin is needed for the zigzag road centerlines that the Voronoi algorithm sometimes produces in the KDE method.) Note that even if the inferred road has a high frequency zigzag pattern next to a straight ground-truth road, the true positive length tp will not exceed the length of the ground-truth road. Consider a situation where two inferred roads lie close to a single ground-truth centerline (within the distance m). The evaluation will find that 100% of the ground-truth road is accounted for, but the precision for this portion will be 50%, as we would expect, since twice as many kilometers will have been inferred than are actually present.

In order to measure the typical road centerline offsets from ground-truth, we separately compute the histogram of perpendicular road distances from each 1m sample from each segment $t' \in M$ to the nearest road segment $t \in Truth$ (with replacement this time). For an inferred map with accurate centerline positions, we would obtain a sharp histogram within a small distance from the centerline.

Note that the evaluation here tests the geometric accuracy of inferred road centerlines, but does not test the algorithm's ability to detect one-way vs. two-way roads, individual freeway lanes, turn restrictions at intersections, or the presence of an intersection node where two road segments cross (cf. [7]). We are not aware of any map inference method that can detect such topological information in sparse data with reasonable accuracy.

Finally, as a matter of protocol when employing quantitative measurements, it is even more important that the debugging, development, and parameter tuning of each algorithm is *not* performed on the same geographical region as

the testing region. We have observed this in our experiments using separate datasets with distinctive features.

5. EXPERIMENTAL RESULTS

In this section we compare multiple algorithms on multiple datasets, using both qualitative evaluation and quantitative performance measurement. The algorithms under consideration are the variants of KDE and the trace clustering algorithm TC1, discussed in Section 3.

5.1 Datasets and Ground-Truth Map

We use two different datasets for evaluation: 118 hours of traces from the UIC campus shuttles [1], and one month of traces from 2,300 taxis in Shanghai [3].

The Chicago data is sampled at 1 Hz, using commodity, low-cost SiRF-3 GPS receivers installed in 13 vehicles. Spatial resolution of this data is 0.000001 degrees latitude and longitude (less than 0.1m), offset by GPS noise with a standard deviation of 3.3m. Certain portions of the Chicago data contain significant noise (tens of meters) due to nearby high-rise buildings [7]. This dataset represents the best quality data one might expect to receive from passive data collection using non-survey grade equipment. However, the coverage of this dataset is severely limited, due to the small number of vehicles and their regular driving patterns.

The Shanghai data was collected by 2,300 taxis, providing excellent coverage of the freeways and arterials. However, the quality of the recorded data is decidedly less stellar. The taxis report their location at various intervals, the most common being 16 seconds (when vacant) and 61 seconds (when occupied). These longer sampling intervals are quite common among the taxi dispatching systems we are aware of [16, 18, 8, 6]. In addition to this, the spatial resolution of these traces varies between 0.0001 and 0.0002 degrees latitude and longitude, or about 10-20 meters, and bearing is reported at a resolution of 45 degrees [24]. We speculate that the taxis were using an early generation of GPS devices, resulting in these poor specifications. The test-area for the Shanghai dataset is displayed in Fig. 1.

For ground truth, we select portions of OpenStreetMap (OSM) data that we have manually verified. The OSM map typically represents freeways and arterial roads by multiple parallel polylines corresponding to opposite lanes. We manually remove duplicate lanes from the ground-truth for this evaluation. With the Chicago dataset, roads not covered by shuttle routes were removed as well. Curiously, judging by our raw GPS traces, the OSM map in the Shanghai area suffers from a bias to the northeast, which is visible in Fig. 1 viewed at high magnification. We observe that most roads in Shanghai agree with the aerial images and were likely created by manually tracing the shape of the roads on the satellite imagery. This suggests that the aerial images are misaligned; see [24] for detail. Based on the visual comparison with the locations reported by the taxi fleet, we shift the ground-truth map 15 m to the east and 12 m to the north for more faithful evaluation.

As discussed in Section 3, we need to determine the *contour threshold* for KDE methods and the support threshold for TC1. In addition to these map-inference control parameters, we need to pick a *matching distance threshold* for the evaluation, i.e., the maximum distance between generated and ground-truth roads that we consider as a match. Our Chicago dataset is relatively small in terms of the size and the area covered. Therefore, we do not separate training and testing data for parameter tuning. Instead, we set the matching distance threshold to 20 m, based on typical GPS error ranges and road widths [24], and pick the optimal contour threshold and support threshold for the experiments. We use this dataset mostly to evaluate the accuracy of different algorithms under different temporal resolutions. On the other hand, our Shanghai dataset covers the entire metropolis for one month. We first examine each algorithm threshold separately to understand its effect. Then we use data from a different area in Shanghai for training, and apply the tuned parameters for a fair comparison of all algorithms. The Shanghai GPS data and its road network are much more diverse than the Chicago dataset. For example, the Chicago dataset covers only straight roads aligned to the four cardinal directions, while the Shanghai dataset covers curved roads, Y-splits, five-way intersections, and all road types. Moreover, the GPS samples in Chicago are somewhat evenly distributed along shuttle bus routes, while there is a great degree of uneven distribution in Shanghai. In summary, "stress testing" using the Shanghai dataset exhibits many important issues for map inference using sparse data.

5.2 Results for the Chicago Shuttle Bus Data

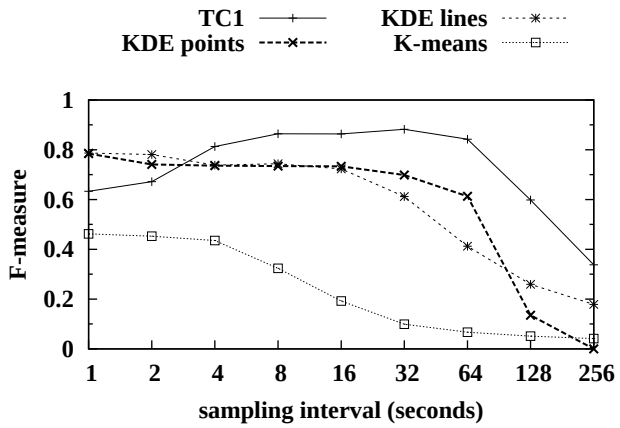
The Chicago dataset in its original form has high resolution in both the spatial and temporal dimensions. To better understand how temporal resolution impacts map inference, we vary the temporal resolution of this data via subsampling. Fig. 3a shows the F-measure of each algorithm at sampling intervals of 1, 2, 4, ..., 256 seconds.

As the sampling interval increases, the performance of KDE-based methods gradually decreases, with the line-based method dropping off considerably when sampled at 32 second intervals and beyond. Intuitively, for larger intervals, the line based method is likely to produce many diagonal segments, cutting across buildings instead of following the roads. This intuition is corroborated by Fig. 3b, where the precision of the line-based KDE method drops off quickly at 32 seconds or more, while recall decays more slowly, as seen in Fig. 3c.

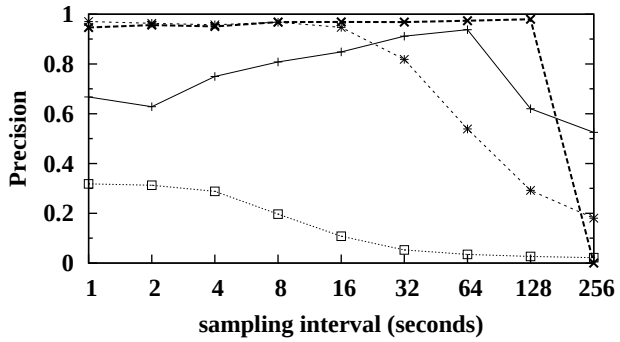
The point-based KDE variant shows somewhat increased tolerance for temporal sparsity. Overall its F-measure is better maintained up to the 64 second interval. Beyond 64 seconds, the decreased density fails to fully cover the streets, resulting in a map with poor recall. However, for the roads it does infer, their precision remains high up until 256, where no roads were found at all. Both KDE methods suffer from a relatively low recall: a large section of the ground-truth map is very lightly traveled compared to the rest of the map, which the global contour threshold discards as noise.

The TC1 algorithm offered the best overall performance, with both high recall and precision. Interestingly, TC1 performance does not consistently improve with higher sampling rates. We discovered that dense data at a turn sometimes links the two perpendicular roads into one cluster, and the centerline extracted from the linked cluster matches neither. Exploring alternatives to single-linkage clustering may lead to further improvements in TC1 performance.

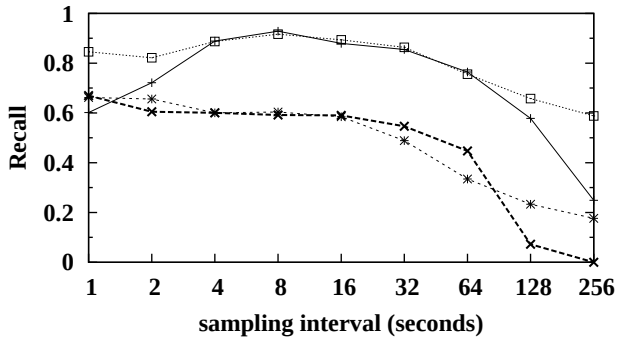
The K-means algorithm produced poor results overall. We believe that this class of algorithms is unsuitable for sparse data, due to its strong tendency to produce spurious segments as data sparsity increases. This is evidenced by the very low precision offered by this algorithm. Due to its poor



(a) F-measure for varying sampling interval



(b) Precision for varying sampling interval

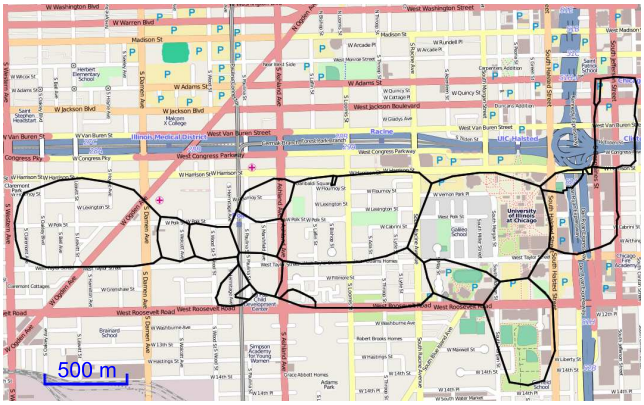


(c) Recall for varying sampling interval

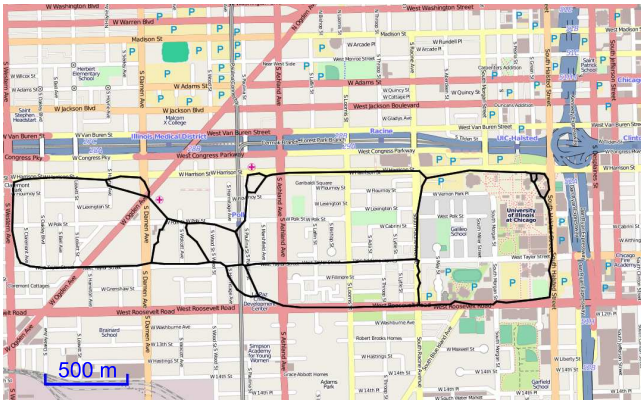
Figure 3: Performance results from various algorithms on Chicago Shuttle Bus dataset.

performance in this initial evaluation, we omit K-means in our comparison for the rest of the paper.

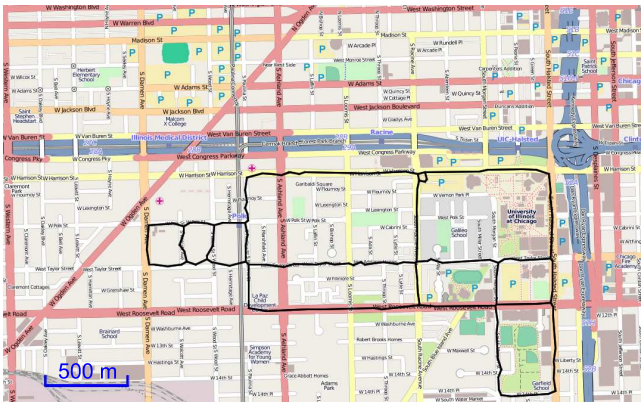
Fig. 4 shows the generated roads overlaid on OSM, using various algorithms at a 60 second sampling interval. Here, it is clear how the line-based KDE method suffers from large numbers of spurious edges, as corners are pulled inward along shuttle routes. A higher contour threshold mitigates this problem, but reduces coverage instead. The point-based KDE does not suffer from the corner problem and produces a high-precision map, but it cannot discover roads that are not *entirely* covered by samples. Since the point-based KDE is less prone to producing spurious density, it tends to produce better results with a low contour



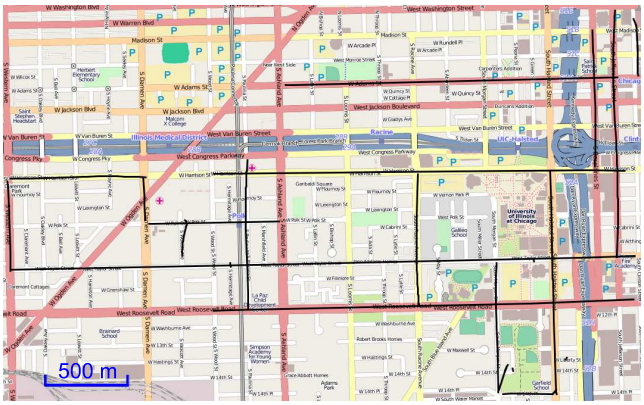
(a) line-based KDE, low threshold



(b) line-based KDE, high threshold



(c) point-based KDE, lowest threshold



(d) TC1

Figure 4: map inference results in Chicago

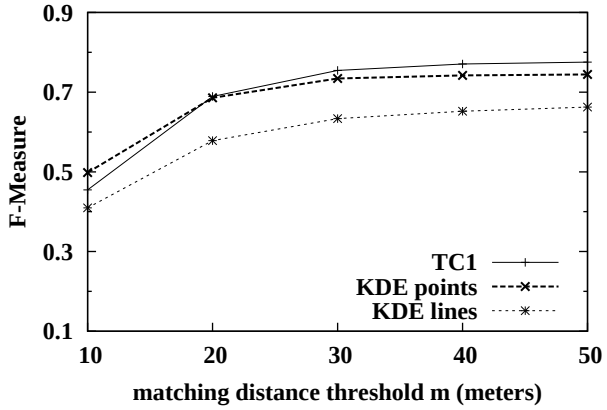


Figure 5: F-measure for each algorithm on a 6-hour dataset, for varying matching distance thresholds. All three algorithms level out around 20 meters.

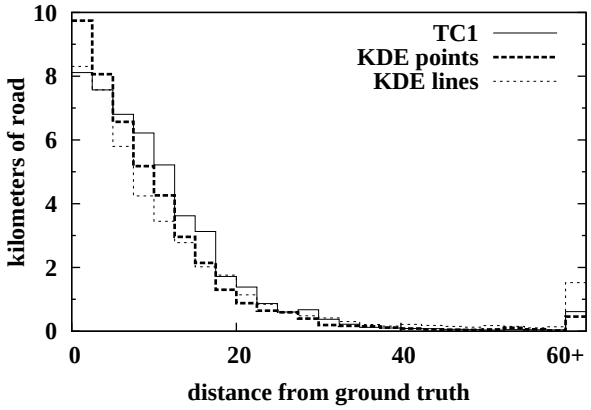


Figure 6: Histogram of distance between generated roads and ground truth.

threshold. Finally, TC1 has good coverage, producing centerlines that match the road shape as the result of B-spline fitting. Qualitatively, we observe a large number of small gaps where line segments meet—an artifact of deliberately discarding turns. This does not impact precision and recall, and simply connecting nearby segments may resolve this issue. Similarly, there are two short vertical road segments on the middle left-hand side that are covered by both KDE methods but not TC1. Here, the bus route turns both immediately before and after the short vertical segment, such that there is typically only one sample on the segment itself. TC1 discards such point pairs, and thus cannot form a cluster.

5.3 Shanghai Taxi Dataset

The Shanghai dataset presents a much larger and more complex problem than the previous Chicago dataset. Here, we first study how the matching distance threshold affects the evaluation results. Recall that the Shanghai dataset represents locations in 0.0001–0.0002 increments of latitude and longitude, resulting in a 10–20 meter granularity.

Fig. 5 shows, for each algorithm, how the F-measure varies with the matching distance threshold. These results are based on a 6-hour dataset from the area shown in Fig. 1. We pick the contour thresholds for KDE methods and the support threshold for TC1 that produce the highest scores. A larger matching distance naturally increases the number of matched roads, allowing both precision and recall to monotonically increase. However, we note that the increase levels off near the 20 meter threshold used elsewhere in this paper.

Based on the same dataset and algorithm parameters, Fig. 6 shows the histogram of distance between generated roads and ground truth for the three algorithms. Again, for all three algorithms, few generated roads fall outside our default 20 meter matching distance threshold. However, there is a relatively heavy tail of distances over 60 meters, suggesting room for further improvement.

Fig. 7 illustrates the trade-off in selecting the contour threshold for KDE and the support threshold for TC1, using the same 6-hour dataset. Here, we see that for the KDE methods, increasing the threshold reduces the number of spurious roads (increasing precision), but also discards roads less frequently traveled (reducing recall). Perhaps counter-

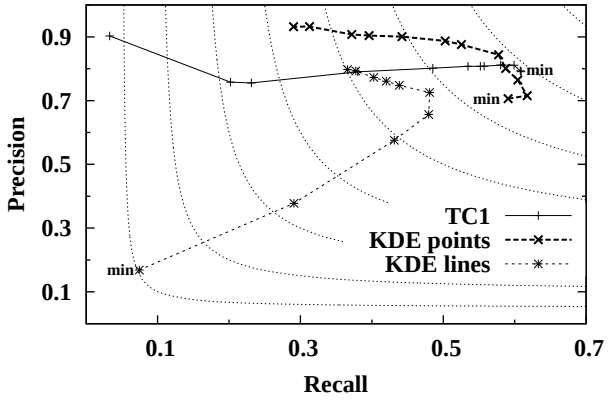


Figure 7: Precision/recall trade-off as we vary the contour threshold for KDE, and the support threshold for TC1. For each algorithm, the threshold monotonically increases from "min" along the line. The background curves show isoclines of equal F-measure, with greatest performance in the top right.

intuitively, KDE-lines is unable to match the recall performance of KDE-points despite drawing the full line between each pair of points. This apparent contradiction is explained by the centerline-finding step in the KDE algorithm: with a low threshold, KDE-lines will produce large areas of solid "road surface," for which the centerline-finding method then produces a single road, significantly limiting recall for this algorithm. A lower support threshold increases the recall of TC1 as well, but it is less correlated with the precision. The pruning step of TC1 removes most of the noise, and the clusters obtained afterwards are usually accurate. As F-measure is the harmonic average of precision and recall, the best F-measure is attained towards the top right corner; the isoclines in the background connect precision-recall points with equal F-measure.

Next we study how each algorithm performs as we vary the amount of input data. We select a different training area of similar size in Shanghai for parameter tuning. We find the contour thresholds for KDE methods and the support threshold for TC1 that produce the best F-measures on the training area. For TC1, the same threshold is then used for

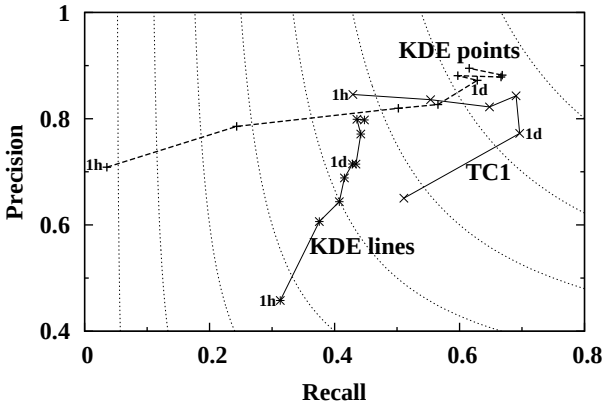


Figure 8: Precision and recall as we vary the number of hours of Shanghai taxi data: 1, 2, 4, 8 hours, 1, 2 days, and (for KDE only) 1, 2, 4 weeks. Isoclines show F-measure.

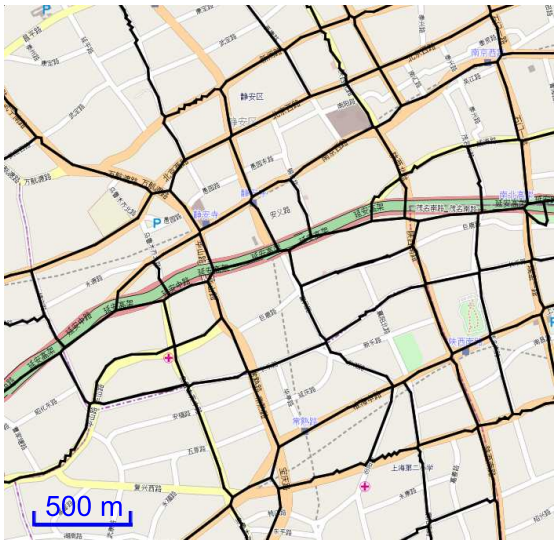
the test area of Fig. 1. For KDE, the density percentile of the threshold is preserved, rather than the absolute value. Fig. 8 shows the precision, recall, and F-measure results as we vary the input data from one hour to one month.

As one might expect, recall initially increases rapidly as trace coverage improves. Overall, increasing the amount of data has relatively little effect on precision: we train the algorithms separately for each data size, which automatically adjusts the thresholds to compensate for the extra data. However, the performance improvement starts to fizzle around one week of data (for KDE-points), and 24 hours (for TC1). For TC1, this can be explained by the use of single-linkage clustering: additional data increases the diversity of points, which raises the probability of nearby clusters merging incorrectly. We did not run TC1 for datasets larger than two days due to its high runtime complexity.

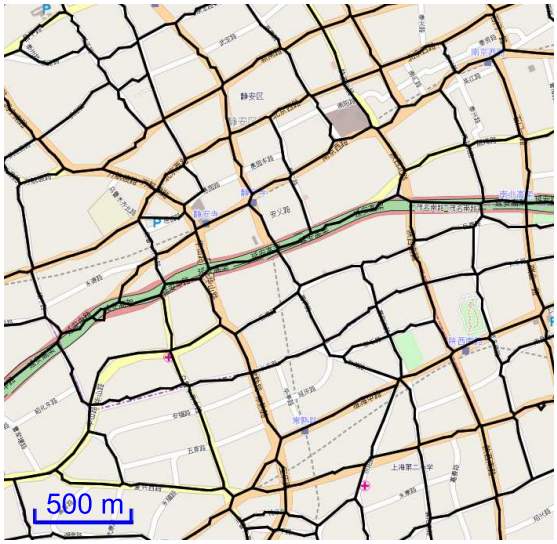
For KDE-points, over-fitting to training data appears to be one culprit: for the larger datasets, a lower threshold than what training produced results in a considerably better map for the test area. This suggests that there is room for improvement in the way the KDE-points classifier is trained. Overall, we find that KDE-points and TC1 exhibit similar numerical performance, with KDE-lines falling far behind.

Finally, Fig. 9 shows the inferred maps from the three algorithms overlaid on OSM map tiles, using the bounding box from Fig. 1. This yields some valuable qualitative insight into the relative performance of the three algorithms. KDE-lines by necessity chooses a high threshold in order to distinguish between roads. This, in turn, reduces recall leading to a clean, but sparse-looking map. KDE-points produces what from a distance appears to be a very accurate map, with very few spurious road segments. This is somewhat surprising, given the merely 90% precision reported in Fig. 8. However, upon closer inspection, the KDE-points map exhibits severe jaggedness in many places, significantly increasing total road length, thus reducing precision according to the definition in Section 4.

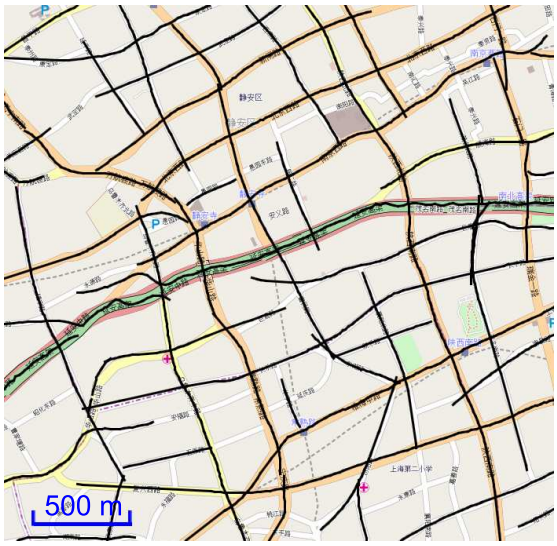
The TC1 map (Fig. 9c) shows very good coverage, but precision is hampered by a number of spurious road segments. This map also exhibits the disconnected road segments discussed in Section 5.2.



(a) line-based KDE (1 month)



(b) point-based KDE (2 weeks)



(c) TC1 (8 hours)

Figure 9: Inferred maps overlaid on OSM map tiles.

6. CONCLUSIONS AND FUTURE WORK

We have highlighted the problem of inferring road maps from inexpensive, large-scale, *coarse-grained* GPS trace data, which will increasingly become available as a byproduct of other processes. Given the continuing desire to minimize communication costs and the very large install base of sparsely recording GPS tracking systems, we hold that there will be an ongoing need for inference algorithms that can deal with sparse data. Such robust algorithms may succeed partly by the large volume of data available in this form.

While this paper has assessed existing algorithms, the results show there is ample space for ongoing algorithms research. Future work will need to produce high quality, *navigable* maps, including turn restrictions—all from sparse, coarse-grained data. Furthermore, to obtain data volume large enough to have good coverage over all the roads in a city, future system designers will likely wish to include heterogeneous data sources, such as taxis, public transportation vehicles, emergency vehicles, and (by opt-in) cell phones and touch-pad devices. Even with homogeneous data sources, working with large-scale, real-world datasets always involves a substantial effort in data cleaning [24]. Thus, another avenue of future research is to automate the data cleaning processes, especially for heterogeneous data.

The ability to automatically and quantitatively evaluate inferred maps opens the door to machine learning approaches, which require goodness-of-fit measures. It is hoped that by exposing this work to the KDD community, we may encourage future research in this direction.

7. ACKNOWLEDGMENTS

We acknowledge HP Open Innovation Program, Shanghai Pu Jiang Talents Program (10PJ1405800), and NSFC (No. 61170238, 60903190, and 61027009) for the support of Xuemei Liu and Dr. Yanmin Zhu. Also, this material is based upon work supported by the U.S. National Science Foundation under Grants CNS-1017877, CNS-1149989, and DGE-0549489.

8. REFERENCES

[1] BITS lab trace data. `http://www.cs.uic.edu/Bits/Software`.

[2] Open StreetMap. `http://www.openstreetmap.org`.

[3] SUVnet-trace data. `http:/wirelesslab.sjtu.edu.cn`.

[4] More than 50 crashes on Bay Bridge curve. `http://www.usatoday.com/news/nation/2009-11-18-bay-bridge_N.htm`, 2009. USA Today.

[5] G. Agamennoni, J. I. Nieto, and E. M. Nebot. Robust inference of principal road paths for intelligent transportation systems. *IEEE Trans. on Intelligent Transportation Systems*, 12(1):298–308, 2011.

[6] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. In *MobiSys*, 2011.

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

[8] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. N. Koutsopoulos, and C. Moran. IBM infosphere streams for scalable, real-time, intelligent transportation services. In *ACM SIGMOD*, 2010.

[9] L. Cao and J. Krumm. From GPS traces to a routable road map. In *ACM SIGSPATIAL GIS*, 2009.

[10] C. Chen and Y. Cheng. Roads digital map generation with multi-track GPS data. In *Int'l. Workshop on Geoscience and Remote Sensing*, 2008.

[11] D. Chen, L. J. Guibas, J. Hershberger, and J. Sun. Road network reconstruction for organizing paths. In *SODA*, 2010.

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

[13] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective*, pages 128–151, 2003.

[14] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys*, 2008.

[15] Z. Liao. Real-time taxi dispatching using global positioning systems. *Comm. ACM*, 46(5):81–83, 2003.

[16] K. Liu, T. Yamamoto, and T. Morikawa. Feasibility of using taxi dispatch system as probes for collecting traffic information. *J. Intel. Trans. Sys.: Technology, Planning, and Operations*, 13(1):16–27, 2009.

[17] X. Liu, Y. Zhu, Y. Wang, G. Forman, L. M. Ni, Y. Fang, and M. Li. Road recognition using coarse-grained vehicular footprints. Technical Report HPL-2012-26, HP Labs, 2012.

[18] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *ACM SIGSPATIAL GIS*, 2009.

[19] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL GIS*, 2009.

[20] B. Niehoefer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In *1st Int'l. Conf. on Advances in Satellite and Space Communications*, 2009.

[21] S. Schrödl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson. Mining GPS traces for map refinement. *Data Min. Knowl. Discov.*, 9(1):59–87, 2004.

[22] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive GPS, vehicle trajectories. In *Intelligent Transportation Systems, International IEEE Conference on*, 2009.

[23] A. Steiner and A. Leonhardt. A map generation algorithm using low frequency vehicle position data. In *Transportation Research Board, 90th Annual*, 2011.

[24] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li. Challenges and opportunities in exploiting large-scale GPS probe data. Technical Report HPL-2011-109, HP Labs, 2011.

[25] S. Worrall and E. Nebot. Automated process for generating digitised maps through GPS data compression. In *Australasian Conference on Robotics and Automation*, 2007.

[26] L. Zhang, F. Thiemann, and M. Sester. Integration of GPS traces with road map. In *2nd Int'l. Workshop on Computational Transportation Science*, 2010.