Map Inference in the Face of Noise and Disparity

James Biagioni
Dept. Computer Science
University of Illinois at Chicago
jbiagi1@uic.edu

Jakob Eriksson*
Dept. Computer Science
University of Illinois at Chicago
jakob@uic.edu

ABSTRACT

This paper describes a process for automatically inferring maps from large collections of opportunistically collected GPS traces. In this type of dataset, there is often a great disparity in terms of coverage. For example, a freeway may be represented by thousands of trips, whereas a residential road may only have a handful of observations. Additionally, while modern GPS receivers typically produce high-quality location estimates, errors over 100 meters are not uncommon, especially near tall buildings or under dense tree coverage. Combined, GPS trace disparity and error present a formidable challenge for the current state of the art in map inference. By tuning the parameters of existing algorithms, a user may choose to remove spurious roads created by GPS noise, or admit less-frequently traveled roads, but not both.

In this paper, we present an extensible map inference pipeline, designed to mitigate GPS error, admit less-frequently traveled roads, and scale to large datasets. We demonstrate and compare the performance of our proposed pipeline against existing methods, both qualitatively and quantitatively, using a real-world dataset that includes both high disparity and noise. Our results show significant improvements over the current state of the art.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Map inference, map generation, GPS

* This material is based upon work supported by the U.S. National Science Foundation under Grants CNS-1017877, CNS-1149989 and DGE-0549489.

1. INTRODUCTION

Automatically inferring maps from opportunistically collected GPS traces has a range of valuable applications. For example, where no manually-produced maps exist, inferred maps may be the only available option. Elsewhere, inferred maps may be used to detect changes to the road network and to pin-point errors in existing maps. Map inference may also be used to produce custom maps for certain classes of travelers, including pedestrians, bicyclists, transit riders [5], truckers, or tourists, by feeding in data from different sources. The GPS traces needed for producing such maps are becoming increasingly available due to the popularity of smartphones and GPS navigators, which capture large volumes of user location traces on a daily basis.

Inferred maps today however, do not offer the quality we expect of real maps. The existing state of the art in map inference [6, 11, 9] is highly sensitive to disparities in the number of trips made on different roads and to the high levels of GPS noise often encountered in urban areas, typically resulting in maps with either poor coverage or a multitude of spurious and misaligned roads. In this paper, we propose a hybrid map inference method that combines the best aspects of existing algorithms with several new innovations to produce the most accurate map inference method to date. Moreover, in order to accelerate the development of further improvements, we develop an extensible map inference pipeline, make the code and data available, and invite the research community to help bring map inference over the last remaining hurdles.

The primary contributions of this paper may be summarized as follows:

- An extensible, hybrid map inference pipeline with high tolerance to disparities in coverage and GPS noise.
- A gray-scale map skeletonization method for extracting map centerlines from a density estimate.
- A trajectory-based topology refinement technique for edge pruning and intersection merging.
- A trajectory-based geometric refinement technique to estimate intersection geometries.
- A comparative evaluation of the proposed pipeline against several existing map inference techniques from the literature.

Below, an overview of our proposed hybrid map inference pipeline is given in §2. We then discuss each stage of the pipeline in its own section (§3–§7), followed by a brief review of the related literature in §8. We evaluate the performance of our map inference pipeline against that of several algorithms from the literature in §9, and §10 concludes.

2. A HYBRID MAP INFERENCE PIPELINE

In this section, we present a scalable, extensible map inference pipeline, and motivate its design. In addition to creating a high-performance map inference engine, we aim to provide a reusable framework within which improvements to individual components may be made and evaluated by the community. To this end, the source code of the map inference engine, together with example trace data and ground truth maps, are made available on our website [1].

The list below gives a general overview of our pipeline, where each step builds upon the output of the one before it. In this paper, we propose an effective method for each step in turn, but expect future research to offer further improvements for each of the various steps.

1. **Density Estimation (§3).** The full set of GPS traces is passed through a kernel density estimator (KDE) with a Gaussian kernel, to produce a single sample point density estimate for the area in question.

2. **Initial Map Generation (§4).** Road centerlines are extracted using our new gray-scale skeletonization algorithm.

3. **Trace Map Matching (§5).** Viterbi map matching [23, 17] is used to associate each GPS sample point in the original traces with an edge in the initial map, weighted by the mean density beneath each edge.

4. **Topology Refinement (§6).** Here, the map-matched traces are studied to remove low-confidence edges, merge redundant nodes, and infer allowable edge transitions.

5. **Geometry Refinement (§7).** Finally, our topology-invariant geometry refinement step aligns intersections, extracts turn-lanes and fits road segments to transform the resulting topologically-accurate road map into a more geometrically-accurate, finished map.

In a major departure from prior work, we combine initial density processing as in [9, 22, 7, 21] with subsequent trajectory processing [6, 11, 24, 12, 13, 3, 18]. As shown in [4], density processing holds a significant advantage over trajectory processing in terms of robustness to noise and computational complexity, both supremely important considerations as we grow the amount of trace data used. Density estimation's ability to very efficiently consider all traces simultaneously allows us to find an optimal road skeleton, rather than resorting to greedy approximations.

Existing density-based approaches are highly sensitive to density disparities between roads—we address this in §4. More critically, density-based approaches are poorly suited to detecting turn restrictions and grade-separated roadways, as they discard the relationships between points. By preserving these relationships, trajectory-based techniques are better able to perform fine-grained trajectory analysis such as lane detection [11, 19, 8], allowable turn detection [19], and spline fitting of road curvatures [11, 19, 24, 3]. However, the prohibitive complexity of computing globally optimal solutions based on trajectories has led to a variety of greedy solutions in the literature. None of these are robust to noisy GPS data, resulting in poor map quality [4].

By map-matching the original traces to a density-based map in step (3), we recover the relationships between successive samples without the noise sensitivity and scalability problems associated with trajectory-based map inference algorithms. One may interpret map-matching noisy traces to a noise-resilient scaffold as a replacement for the clustering or merging step in existing trajectory-based methods. After matching, we may safely assume that every point matched to an edge is a (potentially noisy) sample from a single road. On rarely traveled roads, however, the noise problem persists: determining whether an underlying road is accurately represented by a single, potentially noisy trace, is an open problem. We now discuss each step in more detail.

3. DENSITY ESTIMATION

In the first stage of the pipeline, the full set of GPS traces is condensed into a single two-dimensional (2-D) density estimate. The area of interest is first discretized into 1x1 meter cells. The number of times a trace passes through each cell is then computed, producing a 2-D histogram.

To account for aliasing problems and GPS errors, the 2-D histogram is convolved with a normal distribution function $N(0, \sigma^2)$, representing the expected GPS error distribution. This is a close (and fast) approximation of a well-known technique in statistics known as Kernel Density Estimation (KDE), with a Gaussian kernel [20]. The choice of σ should be made based on the expected GPS error and road width. We used $\sigma = 8.5$ meters.

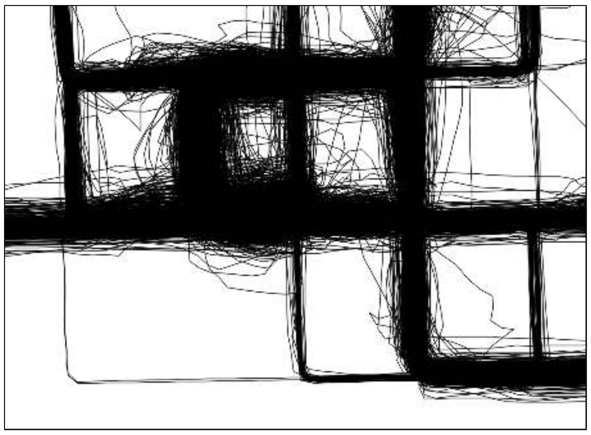
Fig. 1(a) shows a sample of raw GPS traces, and its corresponding kernel density estimate is shown in Fig. 1(b). The density estimate shows clear and smooth peaks along the most heavily traveled roads in our dataset. In the next section, we extract an initial road network based on the density estimate computed here.

4. GRAY-SCALE SKELETONIZATION FOR ROAD CENTERLINE FINDING

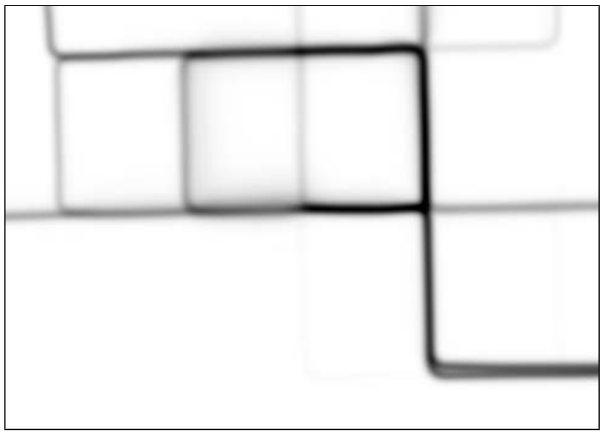
As shown in Fig. 2(a) a simple binary threshold may be used to produce a binary mask of the most popular roads. However, as seen in Fig. 2(b), simply lowering the threshold to include less popular roads also admits a great deal of GPS noise in some areas, creating a difficult dilemma.

The canonical skeletonization algorithm, due to Zhang and Suen [26], produces a characteristic skeleton from a binary image, as illustrated in Fig. 3(a). As mentioned above however, no single threshold can produce a binary image that is both inclusive and accurate in the presence of density discrepancies and GPS noise. Below, we extend the original skeletonization technique to multi-intensity (gray-scale) images, so that we may produce a skeleton without the use of a binary threshold.

At a high level, our algorithm repeatedly performs the binary skeletonization operation, once per integer density level, starting with the maximum density. At each level, new parts are potentially added to the skeleton, but none are ever removed. This process accurately captures the centerlines of high-density ridges. At the same time, it is able to produce centerlines for roads that were only driven once. More formally, the gray-scale skeletonization algorithm proceeds as follows.

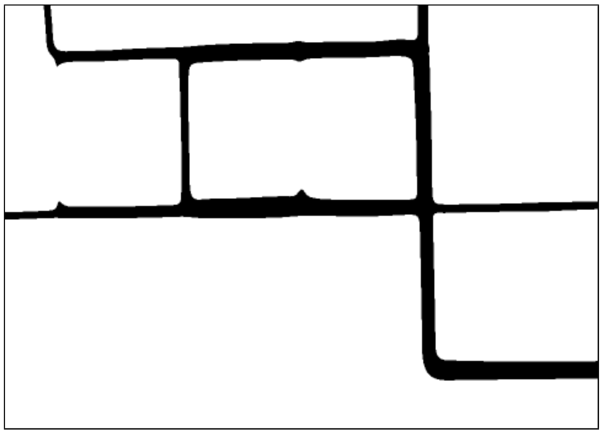


(a) Binary mask of original traces. Coverage density varies by three orders of magnitude, including very significant noise in some areas.

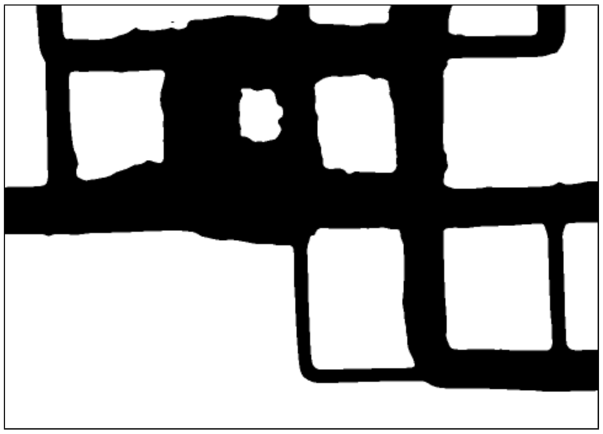


(b) Kernel density estimate. Darker regions correspond to frequently traveled road segments. Some roads are very faint.

Figure 1: As a first step, kernel density estimation produces a continuous distribution out of a noisy set of GPS traces. Note the fuzziness of the density estimate around the high-noise areas.

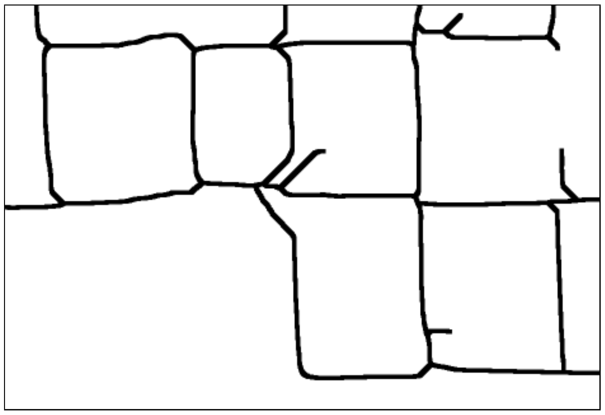


(a) Binary map image derived by applying a high threshold to the KDE.

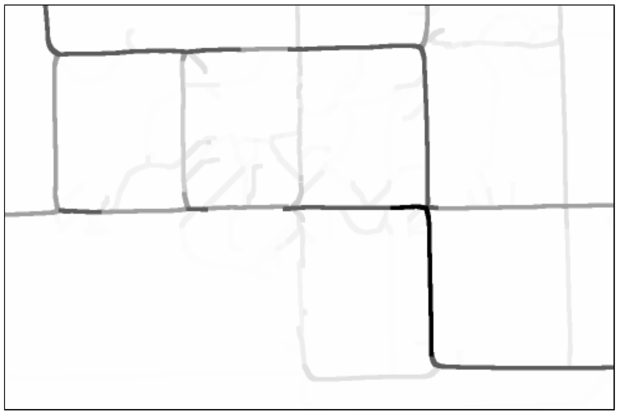


(b) Binary map image derived by applying a low threshold to the KDE.

Figure 2: Simple binary thresholding does not work well, as no single threshold achieves both high accuracy and high coverage. To address this, a threshold-free centerline finding algorithm is needed.



(a) Skeleton from binary image.



(b) Gray-scale skeleton from KDE.

Figure 3: Binary skeletonization vs. gray-scale skeletonization. Gray-scale skeletonization produces more edges, but each edge is more accurate, and annotated by a gray-scale level. The darker the edge, the higher the confidence in its location and existence.

P_9 (i - 1, j - 1)	P_2 (i - 1, j)	P_3 (i - 1, j + 1)
P_8 (i, j - 1)	P_1 (i, j)	P_4 (i, j + 1)
P_7 (i + 1, j - 1)	P_6 (i + 1, j)	P_5 (i + 1, j + 1)

Figure 4: The 8-neighborhood of pixel P_1, used in the process of skeletonization.

4.1 Algorithm Description

Let $\mathbf{D}(x, y)$ be the density at location x, y. For each level $l \in 1 \ldots l_{max}$, let $\mathbf{T}_l$ be a binary (thresholded) image, such that $\mathbf{T}_l(x, y) = 1$ iff $\mathbf{D}(x, y) \geq l$. Our algorithm recursively produces a skeleton image $\mathbf{S}_l$ for level l, such that $\mathbf{S}_l = \mathit{skeletonize}(\mathbf{T}_l + \mathbf{S}_{l+1})$, where $\mathbf{S}_{l_{max}} = \mathit{skeletonize}(\mathbf{T}_{l_{max}})$. Thus, the skeleton pixels at the end of the process have values in the range $1 \ldots l_{max}$ depending on the level at which they were introduced.

The procedure *skeletonize* repeatedly "thins" the image until it converges. Each thinning step consists of applying several local conditions to any pixel of value 1 (unit pixels) in the image, to decide whether to set the pixel to zero. Here, we use the notation from [26], partially illustrated in Fig. 4. For every pixel $P_1 = 1$, the decision to keep it or set $P_1 := 0$ is made based on the values of the neighboring pixels P_2 through P_9. Each major thinning iteration consists of two sub-iterations. In the first sub-iteration, $P_1 := 0$ iff all of the following conditions are satisfied:

$$B(P_1) \geq 2 \quad \text{(a)}$$
$$B(P_1) \leq 6 \quad \text{(b)}$$
$$A(P_1) = 1 \quad \text{(c)}$$
$$P_2 \cdot P_4 \cdot P_6 \neq 1 \quad \text{(d)}$$
$$P_4 \cdot P_6 \cdot P_8 \neq 1, \quad \text{(e)}$$

where $A(P_1)$ is the number of $(0, \geq 1)$-pairs in the ordered set $P_2, P_3, \ldots, P_9, P_2$, and $B(P_1)$ is the number of non-zero neighbors of P_1. In the second sub-iteration, rules d' and e' are substituted for rules d and e:

$$P_2 \cdot P_4 \cdot P_8 \neq 1 \quad \text{(d')}$$
$$P_2 \cdot P_6 \cdot P_8 \neq 1. \quad \text{(e')}$$

Upon convergence, any remaining unit pixels that satisfy the following rule are set to zero:

$$B(P_1) \geq 7 \quad \text{(f)}$$

4.2 Algorithm Intuition

By proceeding level by density level, and preserving any skeleton pixels from higher levels as we progress toward the lower densities, we ensure that high-density pixels are accurately represented in the final skeleton. We now briefly summarize the purpose of each of the conditions above.

Here, Cond. (a)–(e') closely resemble those in [26], adjusted to accept values other than $\{0, 1\}$. Specifically, Cond. (a) preserves unit pixels at the end-points of lines. Cond. (b) forces thinning to progress inward from the edges of contiguous unit pixel areas, consistent with the thinning concept. Cond. (c) preserves any unit pixel that is the sole bridge between two or more non-zero pixels. Here, two or more $(0, \geq 1)$-pairs split the 8-neighborhood into disjoint sets, connected only through P_1. Finally, alternating between Conds. (d–e) and (d'–e') enforces a degree of synchronization in this otherwise highly parallelizable algorithm. This is to avoid a race condition which may thin two-pixel wide lines in a single step, rather than thinning these to a one-pixel line.

In the final step, applying Cond. (f) to the image after convergence is necessary to counter a phenomenon that occurs only in gray-scale skeletonization. Occasionally, an area surrounded by high-density ridges may be entirely filled with cells of non-zero density. Using binary skeletonization, the high density ridges would first be flattened to level 1, and the image thinned until a single line remained through the middle of the low-density area. In gray-scale skeletonization however, the high-density ridges appear early in the process, and cannot be thinned away later. Due to Cond. (b), these ridges then prevent pixels in the surrounded low-density area from being removed. Cond. (f) returns the image to its desired skeleton shape by hollowing out any such surrounded areas in a single step.

4.3 Performance Optimizations

Both the density estimation and skeletonization techniques described are highly parallelizable algorithms that are well suited for GPU or MapReduce implementations. However, as described, the gray-scale skeletonization algorithm runs one binary skeletonization per level. For maps with very high densities, this may grow to an unmanageable number of iterations, even on GPU hardware. In practice, we have found that restricting the levels to powers of two produces largely identical initial maps, with an exponential improvement in execution time.

4.4 Edge Extraction from Skeleton Image

Given a skeleton image, our goal is to produce an initial road map consisting of nodes and edges. We use the combustion technique described in [21] to associate each pixel with an edge, and the Douglas-Puecker algorithm [10] to produce the edges that make up the shape of each road segment.

5. DENSITY-AWARE MAP MATCHING

We now match our original traces to a contiguous sequence of edges from the initial map produced by the skeletonization algorithm above. This accomplishes two things. First, it sets an upper bound on the number of nodes and edges. Later steps may prune and shift nodes and edges, but may not add to the topology. This avoids the tendency of trajectory-based techniques to produce spurious edges. Second, by assigning each point to an edge, it reduces the computational complexity and improves the parallelism of downstream methods that can now operate on each edge independently.

Our map-matching technique uses Viterbi's algorithm, and is based on [23]. Relative to [23], we make the following modifications. To enforce a speed limit, every edge is represented by several consecutive fixed-length states, each of which must be traversed before transitioning to a new edge. This replaces the speed limit heuristic in [23]. Moreover, transition probabilities, which were uniform in [23], are instead assigned values proportional to the edge weight, computed as the mean level of the pixels that make up the grayscale skeleton. Weighting transitions based on edge weight encourages the matcher to use popular roads when the trace allows it, effectively reducing the number of traces that traverse spurious roads.

6. TOPOLOGY REFINEMENT

In the topology refinement step, the initial map produced through density processing is updated based on the the map-matched traces. All edges with zero or one traversals are discarded, pairs of intersections are merged when trace evidence supports it, and statistics are produced to determine the allowable transitions between edges. Below, we describe each step in more detail.

6.1 Pruning Spurious Edges

Through edge pruning, edges that see less than two well-matched traversals are removed. More formally, for each map-matched trace t and edge e, we compute n_e^t, the number of traversals of e such that $RMSD(\tau, e) < RMSD_{max}$, where τ is a traversal of e and

$$RMSD(\tau, e) = \sqrt{\frac{1}{|\tau|} \sum_{p \in \tau} dist(p, e)^2}.$$

Here $p \in \tau$ are GPS points, and *dist* is the distance between p and the nearest point on e. We only consider traversals with $RMSD(\tau, e) < RMSD_{max}$ good matches; these are used as evidence of a road segment's existence.

Aggregating n_e^t across all traces, we have $n_e = \sum_t n_e^t$, the number of well-matched traversals for each edge. Intuitively, any edge with $n_e = 0$ is unlikely to represent an actual road. Such an edge may, for example, have been created by a noisy GPS trace, which was later matched to a more popular nearby road by our weighted map-matching technique in §5.

We argue that the same is true for edges with $n_e = 1$. With a single traversal, the trace will naturally fit the map edge perfectly. After all, the edge was most likely drawn based on that single trace. Thus, the $RMSD(\tau, e)$ is pointless when $n_e = 1$. For this reason, we require two traces to support the existence of any given road segment.

Fig. 5 shows our map before (a) and after (b) this pruning process, clearly illustrating the effectiveness of this technique in reducing the number of spurious road segments.

6.2 Collapsing Nodes into Intersections

While the multi-level skeletonization algorithm in §4 produces an accurate road skeleton for a given density estimate, the generated topology does not necessarily correspond well to typical road designs. One common case of this, caused by an uneven density distribution in the intersection, is illustrated in Fig. 6(a). Here, a single four-way intersection is represented as two adjacent three-way intersections.

To address this common problem, we first sort all pairs of adjacent intersection nodes in order of increasing distance. We then consider collapsing each pair in order, replacing the pair's two m, n-degree intersections with a single (up to) $m + n - 2$-degree intersection at their mean location. If this refinement does not reduce the total number of well matched traces, it is made permanent. Fig. 6(b) shows the result after collapsing. This process effectively transforms the map into a topologically accurate representation of the underlying road network.

6.3 Map Matching Do-Over

After completing the pruning and collapsing steps above, all traces are map-matched once more, this time using the actual number of traversals rather than edge densities to compute transition probabilities. Any traces initially matched to now deleted edges are re-matched to a more likely route, and edge pruning is performed once more. Here, the first pruning round breaks a number of spurious cycles, and the second round removes the remaining spurs after map-matching re-routes traces to more probable routes. The final result from this step is shown in 14(d).

6.4 Detecting Allowable Edge Transitions

Finally, for each trace we compute a list of all adjacent pairs of distinct edges $e : d$, in order. We then compute the number of occurrences of each pair, *count*$(e : d)$ across all traces. To enforce allowable edge transitions we use a strict interpretation of this data: a transition from edge e to edge d is allowed iff *count*$(e : d) > 0$.

7. GEOMETRY REFINEMENT

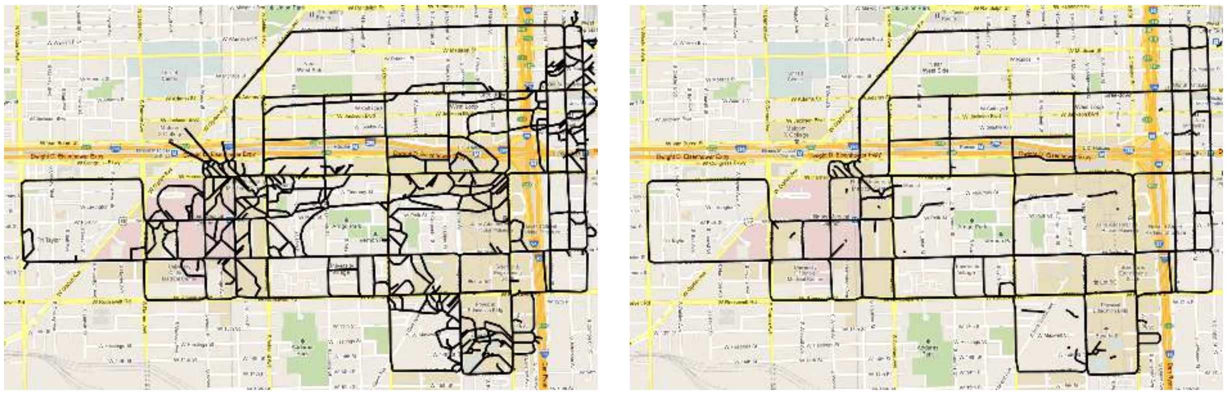
In the geometry refinement step, the map is updated to model intersections in more detail, and to improve the alignment of the inferred map using its original traces. Here, the segment-level topology does not change—segments may shift, but they are not added or deleted. One minor exception to this are turn-lanes, which do contribute additional detail to the topology of an intersection, but do not add new road segments.

7.1 K-means Refinement

Our goal here is to simultaneously refine the entire map, including intersections, rather than refine each segment piecemeal. We propose a geometry refinement technique based on the k-means algorithm. The input to the k-means algorithm is the k initial means and a set of sample points to be clustered. We adapt k-means to the geometry refinement problem by (a) creating an initial estimate based on the input map, and (b) restricting clustering eligibility—which sample points may be assigned to what means, based on the map matching results from §5.

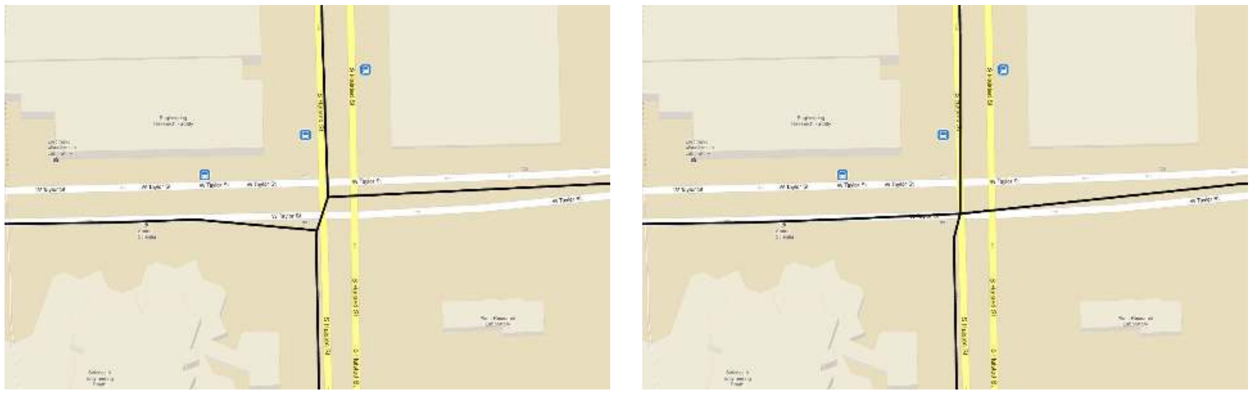
7.1.1 Initialization

Key to a successful application of the k-means algorithm is a good initial estimate. Since the goal here is to refine an existing map, we base our estimate on this map. We produce two classes of means: intersection means and segment means. For each intersection and end-point (i.e., dead-end) in the input map, we add one intersection mean to our initial estimate. Each intersection mean is associated with all segments incident on the intersection. For each road segment in the input map, treating each direction separately, we produce $\lceil \frac{L}{m} - 2 \rceil$ means, where L is the length of the segment, and m is the maximum distance between means. The first and last points of the segment are excluded, as these are already represented by the intersection means. The rest are uniformly distributed along the length of the segment. These means are associated only with the segment from which they originated.


(a) Map before pruning.

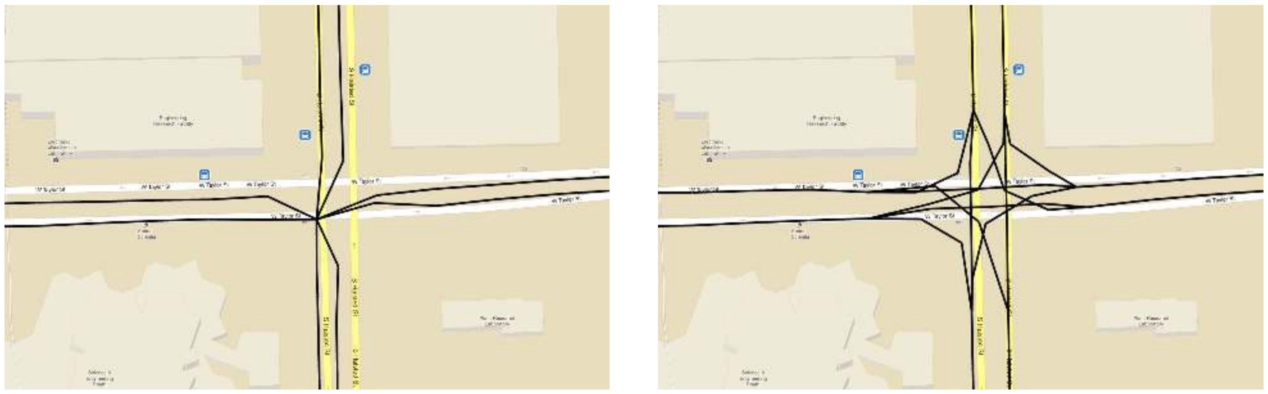
(b) Map after first round of pruning.

Figure 5: Density-based centerline finding produces a number of spurious edges. A trajectory-based pruning process, using the original traces map-matched against the initial map, removes most spurious edges.


(a) Intersection before collapsing nodes.

(b) Intersection after collapsing nodes.

Figure 6: By collapsing nearby nodes into intersections, the final map topology is produced. Nodes are merged only when the original traces support this change.


(a) Same intersection with directional street segments.

(b) Same intersection with turn-lanes added.

Figure 7: Geometry refinement, separating directions and adding turn lanes.

7.1.2 *Assignment*

Each GPS sample is then assigned to its nearest eligible mean. The set of eligible means include those from the segment that the sample was matched to, as well as the intersections or end-points that delimit the segment at each end. Note that intersection means are eligible for GPS sample assignment from all segments incident on that intersection. This optimizes intersection alignment, taking all neighboring segments into account. Simultaneously, segment-based means automatically find the shape of each road segment.

7.1.3 *Update*

In the update step, each mean is moved to reflect its new sample membership. The typical update function simply takes the mean location of all member samples as the new mean location. This can be further refined by first removing points that lie too far from the mean (outliers), and by taking into account the location of neighboring means.

This produces a map with correct lane separation, but with intersections still represented as points in the map topology. Where bi-directional segments exist, this leads an hourglass-shaped intersection geometry, see Fig. 7(a). To produce a correct intersection geometry, we need to estimate each segment transition separately.

7.2 Estimating Transition Trajectories

As a naïve solution, simply replacing the intersection node with direct edges between segments produces a significantly improved map; the hour-glass shape is removed, while the topology is preserved. This approach however, does not accurately capture the individual shape of each lane, often leading to crooked-looking intersections. Below, we describe a solution that separately estimates the transition between each pair of segments, producing a complex, but accurate, intersection geometry.

For each pair of road segments, taken as a single "transition", we prepare a separate set of means according to the initialization method described above. We then perform k-means again, this time using the transition means. Here, a given mean is eligible for assignment only if the current sample came from a matching transition. The generated transition segments and the original street segments are then merged. In this paper, we use a simple constant set-back from the intersection as the merge-point. Fig. 7(b) shows the final result, after adding turn-lanes.

Note how the location of the intersection is initially offset to the south-west in Fig. 7(a). This intersection is highly asymmetric in density, with a large majority of traces in the south-west corner. Density-based processing finds this peak, and places the intersection here. After breaking out the turn-lanes however, this density asymmetry no longer has an effect as each turn-lane has an independent set of traces matched to it. This example emphasizes the power of combining density- and trajectory-based processing for map inference.

8. RELATED WORK

The existing literature concerning the problem of map inference from GPS traces can be organized into three broad categories, surveyed in [4]:

K-means algorithms begin by performing variants on k-means clustering over the set of GPS samples, where the distance measure is defined as a combination of the Euclidean distance to, and compass heading of, a given cluster. In order to extract the road map, a second pass is made through these clusters, connecting with road segments those clusters that fall along the path of the raw GPS traces. Methods that belong to this class include algorithms described by Edelkamp and Schrödl [11], Schroedl et al. [19], Worrall and Nebot [24], Guo et al. [12], Jang et al. [13], and Agamennoni et al. [3].

Trace-merging algorithms incrementally construct a road map by merging together incoming GPS traces that are located nearby in terms of distance and bearing. As a noise-reduction step, those roads with relatively few corroborating GPS trace samples are not included in the final road map. Algorithms belonging to this class include work by Niehofer et al. [18] and Cao and Krumm [6].

Kernel density estimation (KDE) algorithms begin by discretizing the geometric space covered by the set of GPS points into a grid of pixels, recording the approximate density of samples that fall within and across each cell. A global threshold is applied to the resulting density map in order to produce a binary representation of the road network, and the centerline is extracted using a variety of image-processing methods. Representative algorithms here include work by Davies et al. [9], Chen and Cheng [7], Shi et al. [21], and Steiner and Leonhardt [22].

A representative example from each of these categories is included in our evaluation (§9).

8.1 Gray-scale Skeletonization

There is existing work in gray-scale skeletonization by Li et al. [14], and Yim et al. [25]. Li et al. look at the problem of skeletonizing gray-scale images that lack a contiguous contour. Starting from the boundary segments of the gray-scale image, their algorithm develops a so-called Skeleton Strength Map (SSM) from which they are able to extract the skeleton. Yim et al. look at the skeletonization of gray-scale images from magnetic resonance angiography, as a means to identify the paths of small vessels and their tree structures. The technique employed in this work starts by modeling the image as a directed acyclic graph, and then applies selection and pruning methods to isolate the most salient features in order to extract a final skeleton.

8.2 Road Centerline Finding Algorithms

Prior work on kernel density estimation (KDE) algorithms have employed a variety of road centerline finding methods. In Davies et al. [9], a form of Voronoi tessellation is used within the boundaries of the binary road network. By pruning away short "dead-end" segments, they are able to recover a contiguous spline along the approximate road centerline. In Chen and Cheng [7], morphological "dilation" followed by "closing" operations are used to fill gaps in the binary road network representation, and then morphological "thinning" is used to extract the centerline of the resulting shape. Lastly, in Steiner and Leonhardt [22], the watershed transform [16] is used to extract the centerline. This transform suffers from two major weaknesses making it unsuitable for our purposes. First, it depends on a method for identifying a set of local minima from which to initiate the flooding process, and reliably choosing minima that produce a good set of ridges is non-trivial. Second, the watershed transform is fundamentally unable to detect dead-end roads, since these cannot form a separate flooding basin.

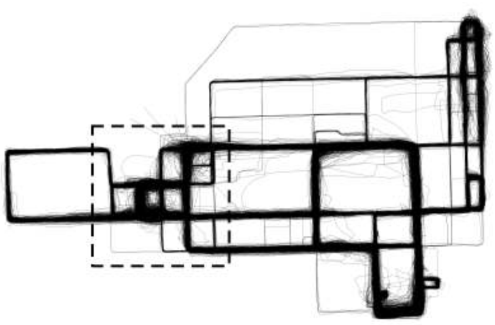


Figure 8: 7 months of raw traces. Data displays both high disparity and high GPS noise. Dashed square indicates the "hospital area."

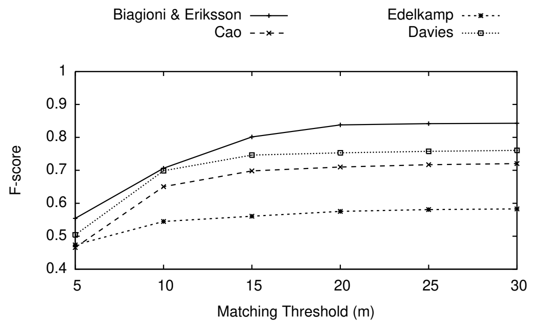


Figure 9: GEO F-scores of our method vs. existing methods on the 1-month dataset.

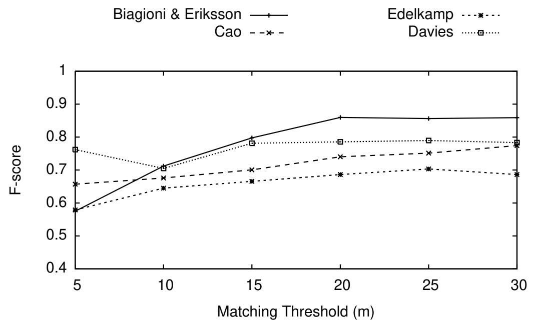


Figure 10: TOPO F-scores of our method vs. existing methods on the 1-month dataset.

9. EVALUATION

In this section, we perform a quantitative and qualitative evaluation of our map inference pipeline. Where possible, we compare our results to those of three previous map inference algorithms, demonstrating dramatically improved performance both quantitatively and qualitatively.

9.1 Dataset

For our evaluation, we use up to 7 months of data collected from 13 university shuttle buses serving the University of Illinois at Chicago campus. A mask of the full set of traces is shown in Fig. 8. These shuttles serve several regular routes with dozens of trips per day, two commuter shuttle routes with a handful of trips per day, as well as occasional chartered trips. This combination of occasional, infrequent, and frequent trips provides a high degree of density disparity.

In addition, the area served contains a mix of low-rise residential buildings, mid-rise campus buildings, and high-rise office and hospital towers. In the hospital area (highlighted by the dashed square in Figure 8), urban canyons are created where GPS reception is highly error-prone, with some traces showing consistent errors well over 100 meters.

9.2 Other algorithms

In addition to our proposed map inference pipeline, we evaluate three existing algorithms for comparison purposes: the KDE-based method by Davies et. al [9] (Davies), and two trace-based methods by Edelkamp et. al [11] (Edelkamp) and Cao and Krumm [6] (Cao). For this comparison, we limit our data to a 1-month subset for two reasons. First, and most importantly, the inherent scalability problem of [6] makes it infeasible to evaluate this algorithm with a larger dataset. Second, due to noise sensitivity, the performance of [11] declines as the amount of data exceeds the 1 month mark.

9.3 Evaluation Methodology

The purpose of our evaluation is to determine the accuracy with which each inference algorithm represents the underlying road network. As ground truth, we use a manually-verified section of OpenStreetMap [2] for our region of interest. Because not all streets in this map are actually visited by our vehicles, we make one modification to the ground truth map: all traces are map-matched to the ground truth map, and any road segment that is not traversed by at least one trace is removed from the ground truth map.

We use two different evaluation methods from previous work to compare our inferred maps to the ground truth map. The first method (GEO) is taken from [15], and evaluates map geometry only. Here, the connectivity of the map is ignored entirely, but every segment of both maps is taken into account. In brief, the full set of segments of each map are first sampled at 5-meter intervals. Then, a bi-partite, one-to-one matching is computed between the two sets, up to a maximum *match distance threshold*. Remaining samples from the inferred and ground truth maps are termed *spurious*, and *missing* respectively.

The second method (TOPO) is taken from [4], with one modification. This sampling-based method evaluates the topology of the map as well as the geometry. We modify the method to ignore parts of the map where no correspondence could be found between inferred and ground truth maps. Thus GEO partially evaluates the entire map, whereas TOPO fully evaluates those parts where the maps overlap. See [4] for more details. For both methods we use the F-score, the geometric mean of *precision* and *recall*, as our primary evaluation metric.

For the results below, we use the topology refinement output only. While geometry refinement produces a more descriptive map, our ground truth does not contain this level of detail, making it infeasible to quantitatively evaluate the performance of the geometry refinement step. Visual samples of the geometry refinement output are provided in 7(a) and 7(b).

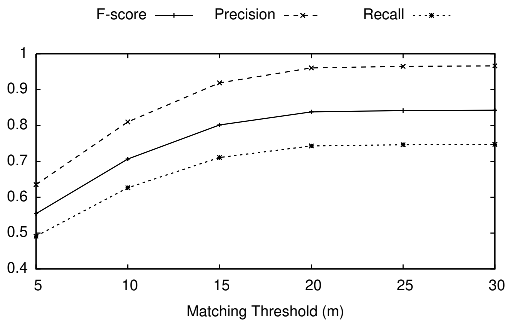


Figure 11: Precision/recall and F-score on 1-month dataset. GEO evaluation metric.

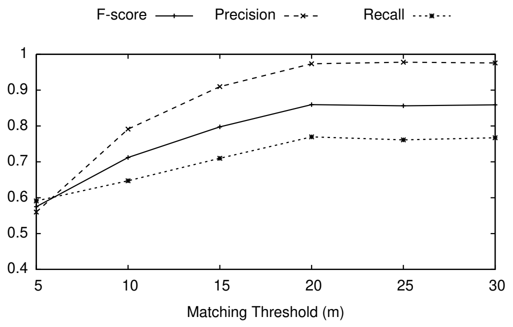


Figure 12: Precision/recall and F-score on 1-month dataset. TOPO evaluation metric.

9.4 Results

Our main result is shown in Figs. 9–10. Our new method offers a significant improvement over the previous state of the art in both GEO and TOPO evaluations.

Figs. 11–12 offer further insight into the limitations of our proposed pipeline. While precision falls in the 0.9–1.0 range for matching thresholds of 15 meters and above, recall falls below 0.7. Visually inspecting the generated maps, the explanation behind this is clear: many roads in the ground truth were traversed only once, and in the current topology refinement step, roads with a single traversal are pruned. In future work, we hope to address this problem to further improve performance.

Finally, Fig. 13 shows the GEO performance on 7 months of data. The slightly sharper bend of the curve suggests that additional data was helpful in producing more accurate centerlines. However, when using a higher matching threshold, performance is essentially unchanged. In this case, additional data simultaneously improves the map quality and introduces additional ground truth edges. With more evenly distributed data, we expect to see a monotonic performance increase with larger data sets.

Figs. 14(a)–14(d) visually illustrate the performance of each algorithm across the entire region of interest. Our proposed method produces a significantly more complete map, with very few spurious edges. Note that here, we show only the 7-month dataset for our proposed pipeline, due to space limitations. The 1-month map has somewhat fewer edges but is otherwise similar.

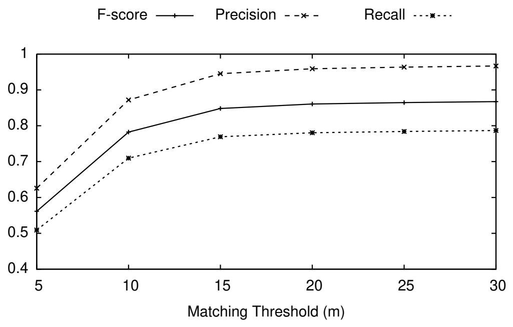


Figure 13: Precision/recall and F-score on 7-month dataset. GEO evaluation metric.

Focusing our attention on the "hospital area" which has high density, disparity, and GPS error, we also see that our method produces a considerably higher quality map, with better coverage and improved alignment.

10. CONCLUSION AND FUTURE WORK

We have presented a hybrid map inference pipeline, which significantly advances the state of the art when considering noisy and disparate datasets. Key to our work is the combination of initial density processing, with its ability to consider all traces in aggregate, followed by trajectory processing, with its capacity for capturing topological and geometric details.

While the results on this dataset are very good, much work remains to validate our approach on different datasets, and tune each step for optimal performance. By releasing the source code and data used in this paper, we invite the community to contribute to this evolving project. These resources are available on our website [1].

11. REFERENCES

[1] Bits Networked Systems Laboratory, http://bits.cs.uic.edu/. Accessed July 22, 2012.

[2] OpenStreetMap, http://www.openstreetmap.org/. Accessed July 22, 2012.

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

[4] J. Biagioni and J. Eriksson. Inferring Road Maps from GPS Traces: Survey and Comparative Evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2012.

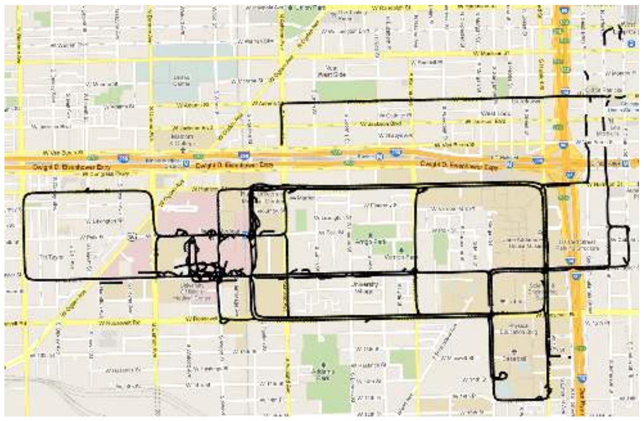
[5] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. In *SenSys*, pages 68–81. ACM, 2011.

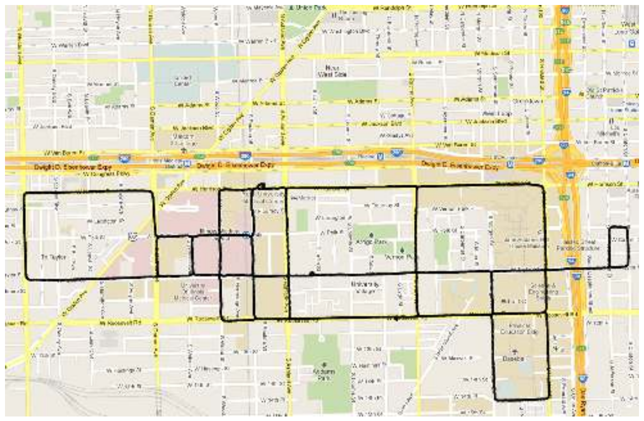
[6] L. Cao and J. Krumm. From gps traces to a routable road map. ACM GIS, pages 3–12, 2009.

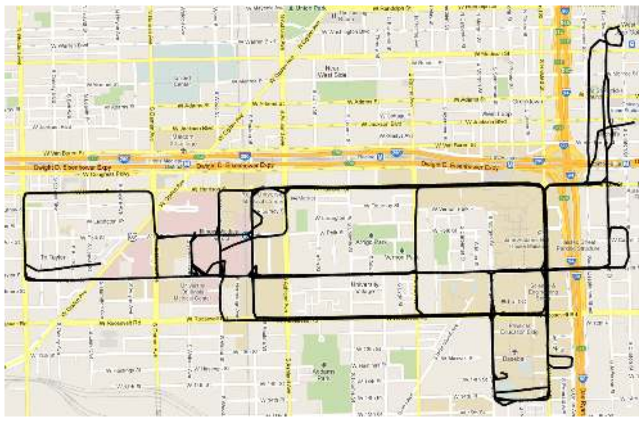
[7] C. Chen and Y. Cheng. Roads digital map generation with multi-track gps data. In *ETT and GRS 2008.*, volume 1, pages 508–511, December 2008.

[8] Y. Chen and J. Krumm. Probabilistic modeling of traffic lanes from gps traces. GIS, pages 81–88, New York, NY, USA, 2010. ACM.

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


(a) Full map generated using Cao method with 1 month of raw data, overlaid on a map.


(b) Full map generated using Davies method with 1 month of raw data, overlaid on a map.


(c) Full map generated using Edelkamp method with 1 month of raw data, overlaid on a map.

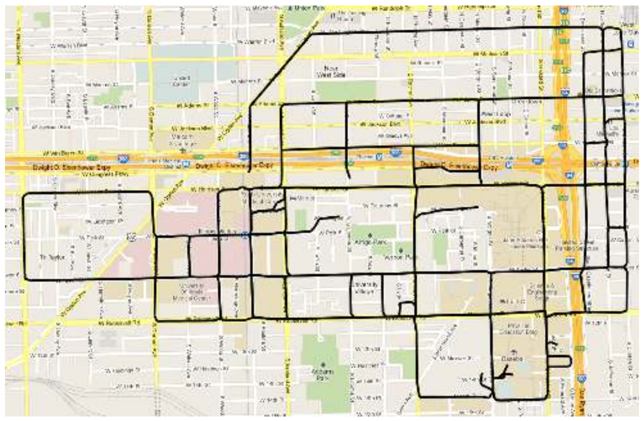

(d) Full map generated using our method with 7 months of raw data, overlaid on a map.

Figure 14: Visual illustration of the performance of each algorithm across the entire region of interest.

[10] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[11] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In R. Klein, H.-W. Six, and L. Wegner, editors, *Computer Science in Perspective*, volume 2598 of *LNCS*, pages 128–151. Springer, 2003.

[12] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive gps traces data. In *IGARSS 2007*, pages 667–670, 2007.

[13] S. Jang, T. Kim, and E. Lee. Map generation system with lightweight gps trace data. In *ICACT*, volume 2, pages 1489–1493, February 2010.

[14] Q. Li, X. Bai, and W. Liu. Skeletonization of gray-scale image from incomplete boundaries. In *ICIP 2008*, pages 877–880, oct. 2008.

[15] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining Large-Scale, Sparse GPS Traces for Map Inference: Comparison of Approaches. In *KDD*. ACM, 2012.

[16] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, 1994.

[17] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. GIS, 2009.

[18] 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 *SPACOMM 2009*, pages 156–161, July 2009.

[19] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson. Mining gps traces for map refinement. *Data Mining and Knowledge Discovery*, 9:59–87, 2004.

[20] D. W. Scott. *Kernel Density Estimators*, pages 125–193. John Wiley and Sons, Inc., 2008.

[21] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive gps, vehicle trajectories. In *ITSC '09*, pages 1–6, October 2009.

[22] A. Steiner and A. Leonhardt. A Map Generation Algorithm using Low Frequency Vehicle Position Data. In *90th Annual Meeting of the Transportation Research Board*, 2011. 17 pages.

[23] A. Thiagarajan, L. Sivalingam, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan. Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, 2009.

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

[25] P. Yim, P. Choyke, and R. Summers. Gray-scale skeletonization of small vessels in magnetic resonance angiography. *Medical Imaging, IEEE Transactions on*, 19(6):568–576, June 2000.

[26] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, Mar. 1984.