CrowdAtlas: Self-Updating Maps for Cloud and Personal Use

Yin Wang
HP Labs, Palo Alto
yin.wang@hp.com

Xuemei Liu[*]
Baidu Inc., Shanghai
liuxuemei01@baidu.com

Hong Wei
Shanghai Jiao Tong University
keith.collens@sjtu.edu.cn

George Forman
HP Labs, Palo Alto
george.forman@hp.com

Chao Chen
Shanghai Jiao Tong University
chaochen@sjtu.edu.cn

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

ABSTRACT

The inaccuracy of manually created digital road maps is a persistent problem, despite their high economic value. We present CrowdAtlas, which automates map update based on people's travels, either individually or crowdsourced. Its mobile navigation app detects significant portions of GPS traces that do not conform to the existing map, as determined by state-of-the-art Viterbi map matching. When there is sufficient evidence collected, map inference algorithms can automatically update the map. The CrowdAtlas server aggregates exceptional traces from users with the navigation app as well as from other, large-scale data sources. From these it automatically generates high quality map updates, which can be propagated to its navigation app and other interested applications. Using CrowdAtlas app, we mapped out a 4.5 km^2 street block in Shanghai in less than half an hour and built a walking/cycling map of the SJTU campus. Using taxi traces collected from Beijing, we contributed 61 km of missing roads to OpenStreetMap, the first set of completely computer-generated roads for this large, open-source map community.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; H.2.8 [**Database Management**]: Database Applications—*Data mining; Spatial databases and GIS*; H.4 [**Information Systems Applications**]: Miscellaneous; I.5.4 [**Pattern Recognition**]: Applications

Keywords

mobile systems; spatial data mining; GPS; map matching; map inference; road maps

[*]The work was performed during an internship at HP Labs.

1. INTRODUCTION

Year 2012 marks the 500th anniversary of cartographer Gerardus Mercator's birth, who introduced the term "atlas" for maps and invented the Mercator projection widely used today. Nautical charts and terrain maps have long been precisely plotted with modern cartography technologies. However, digital road maps remain surprisingly inaccurate and incomplete. Both keynote talks of the recent ACM GIS conference mentioned persistent map accuracy problems with Google, Apple, etc., and emphasized auto map correction as one of the top open problems [35, 39]. Since people increasingly rely on navigation using smart phones for everyday activities, inaccurate maps can have substantial economic consequences, as evidenced by the recent iOS map incident [19]. A British insurance survey found 26% of drivers had been directed by their GPS to go through no-entry signs or prohibited areas [33], and the news periodically reports car accidents caused by or related to digital maps [29, 52, 54].

The U.S. maps of Google, Waze, and OpenStreetMap all stem from the TIGER products of the U.S. Census Bureau, which have various quality issues such as road disconnections and misalignments [1, 36, 56]. In addition to correcting existing maps, updating them promptly and consistently can be more challenging. TomTom estimates that up to 15% of roads change each year in some way [5]. The map situation is worse in developing countries where there is more new construction and roads more often degrade beyond use. We show later that commercial maps of central Shanghai have missing roads and incorrect road types. While inaccurate maps rarely create problems for daily commuting, they cause plenty of frustration for travelers in unfamiliar areas, especially in foreign countries because of the additional language barrier.

In addition to heavy demand for updating road maps, there are also growing needs for additional types of maps for non-motorist traffic. Fueled by the green transportation movement, many cities are ramping up the rate of bike path development [25]. Unfortunately, most cycling maps rely on crowdsourcing and are far from complete [22]. In addition, personalized maps are increasingly popular. There are many applications and online services that customize maps to include personal points of interest. Customized road networks can help navigate private or personal routes not on existing maps, e.g., off-road driving or cycling, and backcountry hiking or skiing.

Existing commercial and public maps are created manually, which is costly, error-prone, and cannot keep up with the aggregate rate of change. For example, NAVTEQ (now Nokia) employs more than 7,000 employees worldwide in its Location Content team to update its maps [2]. Google, TomTom, and Waze also encourage users to submit map edits online, pending manual review by their teams of content editors. The largest crowdsourced mapping project is OpenStreetMap (OSM), which has more than 600,000 registered contributors. However, due to the skills and effort needed for map creation, user-contributed maps on OSM are limited in detail, freshness and accuracy. For example, its street-level maps in China are currently limited to a few tourist locations and have various quality issues, as we will illustrate. There is no automated map update system.

Automated road inference using GPS probes has gained much attention recently [12,15,17,20,23,32], partly because large-scale GPS traces are increasingly abundant from systems that track user locations, such as fleet management, telematics systems, and online navigation apps [9,10,37,55]. Most existing map inference methods rely on low-noise, densely-sampled, and uniformly distributed GPS traces. With high-noise, sparsely-sampled, and highly-disparate data, calculating road geometry, inferring road connectivity, and gaining high map coverage all become very challenging. A few recent studies try to address some of the challenges but not all of them simultaneously [12, 32]. In addition, tunnels, plane-separated roads, and road metadata cannot be inferred from longitude/latitude data alone. By contrast, completing and updating a reasonably accurate base map is much more practical than generating a new map from scratch. For example, some work has demonstrated refining the road geometry in a base map using densely sampled GPS traces [44,45].

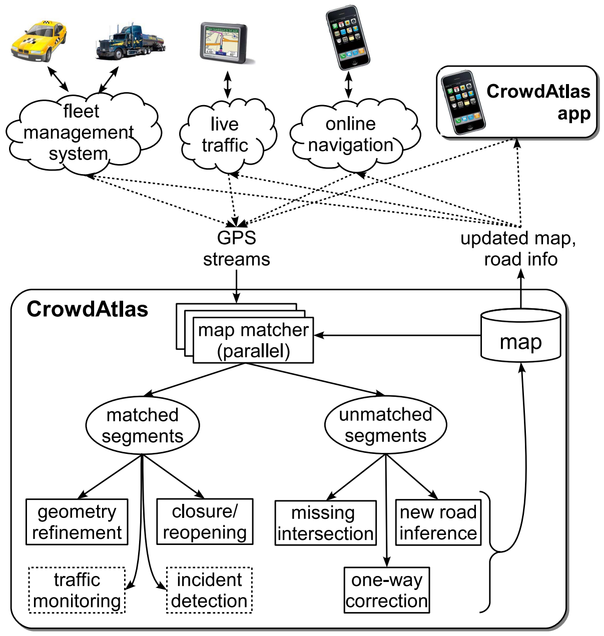
Leveraging the latest GPS map inferencing techniques, we built CrowdAtlas, a system that automatically and continuously rectifies existing maps using data collected from mobile probes. CrowdAtlas employs a *Hidden Markov Model* (HMM)-based *map matching* algorithm [40,56] to detect the discrepancies between GPS samples and roads, and applies a clustering-based map inferencing algorithm [32] to the discrepant traces to update maps. CrowdAtlas can utilize large-scale GPS data from existing location tracking systems and provide reciprocal benefits back to such systems. We also built CrowdAtlas app, a *map-inaccuracy*-aware navigation app, for users without access to these systems or even without internet connection, and for users who want to create and use customized maps.

We make the following contributions.

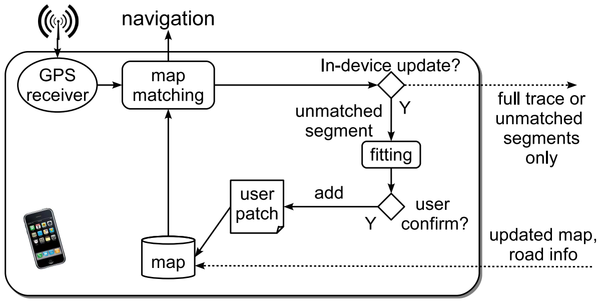
- An automatic map update system that consists of a server and/or a navigation app. The server attains coverage and accuracy by leveraging large-scale, noisy GPS streams from existing location-tracking systems. The app can add new roads in a standalone mode.

- The novel combination of map inference with navigation map matching for practical, highly-accurate, and real-time map update, including heavy customization of our prior state-of-the-art map inference and map matching algorithms.

- Implementation and evaluation of CrowdAtlas, completing roads in a 4.5 km^2 street block in the Shanghai Pudong district, creating a cycling/walking campus map of Shanghai Jiao Tong University, and contributing 61 km of roads for the Beijing map on OSM. The latter is based on 8 days of GPS data from 70 taxis, obtained from a fleet management system. To the best of our knowledge, this is the first set of computer-generated roads uploaded to OSM. We make our datasets available for public use [1].



(a) CrowdAtlas system and the overall context



(b) CrowdAtlas navigation app

Figure 1: CrowdAtlas architecture

Section 2 gives an architectural description of CrowdAtlas. Sections 3 and 4 describe the algorithms for map matching and map updates, respectively, and Section 5 gives implementation details. Section 6 contains evaluations for standalone and server updates. Section 7 discusses related work, and Section 8 concludes.

2. OVERVIEW

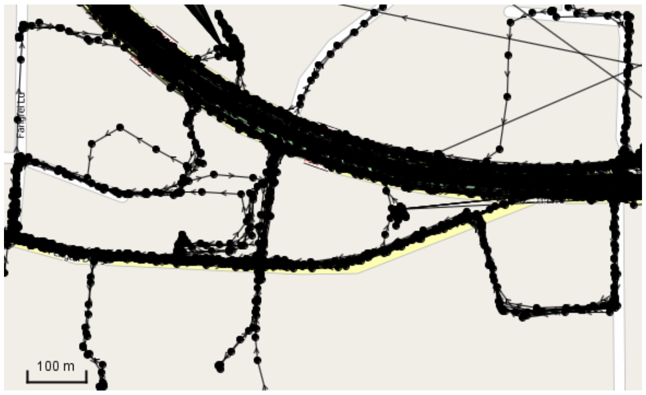
We discuss the CrowdAtlas server and its mobile navigation app in turn.

2.1 CrowdAtlas Service

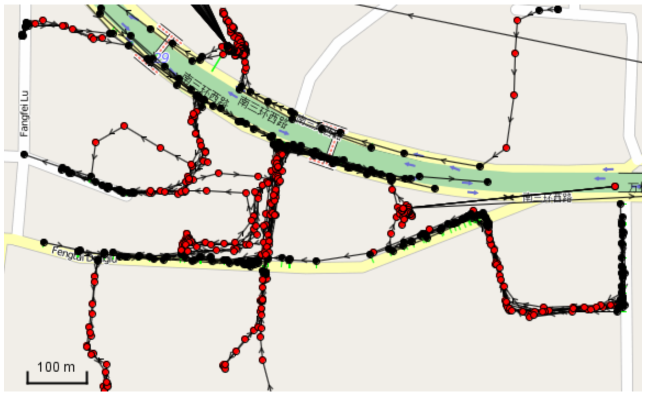
Figure 1a shows the overall system. At a high level, CrowdAtlas consumes GPS streams from various sources and offers map services with real-time updates. The idea is like *crowdsourcing*; however, compared with other crowdsourced map services, updates can be automated without requiring skilled users or user attention. Because of this automation, it can also leverage a variety of large-scale GPS trace sources from existing services or applications that do location tracking, e.g., fleet management systems for service or delivery [13, 55], live traffic estimation such as by TomTom or Garmin, and online navigation such as by Apple or Google maps.

CrowdAtlas consists of a series of stream processing tasks. First, the GPS traces go through a *map matching* procedure using the current map data. This is a common procedure used in location tracking and navigation, which matches a series of GPS coordinates to the most likely passable route on the given map. We customized a reliable map matching algorithm [40, 56] so it can recognize map inaccuracy during the matching process, and thereby we separate matched and unmatched trace segments. Map matching is the most time-consuming step. Fortunately, it can be run independently for different traces. CrowdAtlas parallelizes map matching using multithreading for city- or state-scale map update, or distributed computing for global-scale map update. After map matching, CrowdAtlas infers new roads from unmatched segments, using a streaming variant of our prior map inference algorithm [32]. We also extend this algorithm to infer other missing map features, including intersections, new turn possibilities, and one-way road directions and corrections. In addition to inferring new map features, CrowdAtlas uses the matched segments to refine road geometry and to detect road closures and reopenings. The scalable and reliable map matcher also enables other GPS-based analysis tasks, such as traffic monitoring [27, 59] and incident detection [49]. For example, each GPS sample typically includes the travel speed measurement, or we can estimate it from the time interval and the map matched path between consecutive samples. The median speed of samples matched to a road segment is the current travel speed of the road. While CrowdAtlas can provide these additional information services to users, we focus on inference of missing roads in this paper, which is the most challenging task among all updates CrowdAtlas provides.

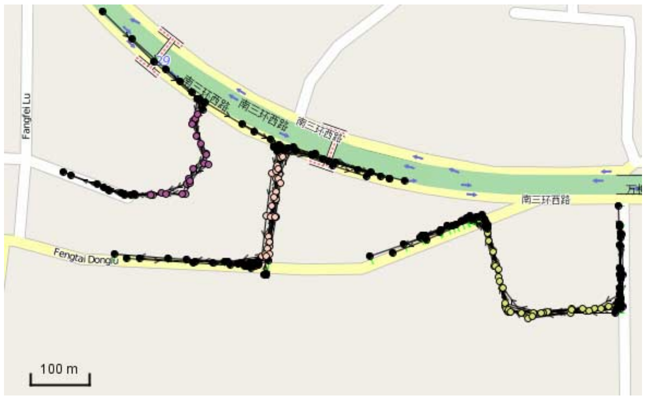
Figure 2 illustrates the overall stream processing flow for new road inference on the server. Figure 2a shows 8 days of data from 70 taxis in Beijing, with a sampling interval of 10 seconds. This area has few high-rise buildings, but noise still exists. After map matching, Figure 2b shows unmatched sub-traces (red) with a few matched samples (black) before and after. (The short green line segments in the figure connect matched samples to their road locations.) The unmatched sub-traces are then aggregated with a stream clustering algorithm. Once a cluster exceeds a support threshold, CrowdAtlas invokes a *polygonal principal curve* algorithm [31] to extract the road centerline from the cluster. With one week of data and a threshold of four sub-traces, there are three clusters in the area, shown with distinct colors in Figure 2c, and their three inferred roads are shown in Figure 2d, overlaid on a Bing aerial image. Judging from the image, these line segments are well aligned with the centerlines of the actual roads. The server can infer other missing roads in the area as more unmatched GPS traces accumulate over time. One could lower the support threshold to achieve greater recall of missing roads, but this



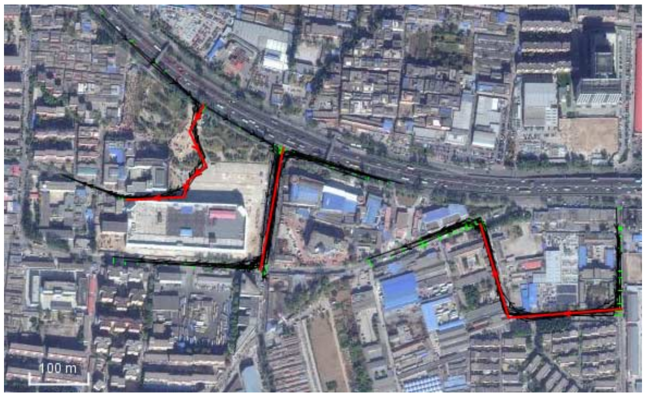
(a) Input GPS traces (70 taxis in Beijing, 8 days)



(b) Extracting unmatched segments (red) after map matching



(c) Extracting clusters with sufficient unmatched segments



(d) Fitting new roads to clusters (red, arrows for one-way)

Figure 2: Stream processing flow on CrowdAtlas server

increases the possibility of inferring roads from noise and leads to greater position error.

2.2 CrowdAtlas App

CrowdAtlas can leverage existing GPS data from location tracking systems and services. In addition, we built our own navigation app that can either upload GPS traces to the server and receive aggregated map updates, or else update in-device maps based on personal travel habits using our software in a standalone mode. The app has several key benefits over other navigation apps or services that could use CrowdAtlas as a map provider. First, it can build personalized maps for an individual, e.g., trail, cycling, or off-road driving maps. Second, its navigation is aware of map-inaccuracy and is optimized for the purpose of map update. It dynamically adjusts the sampling rate transmitted to the server to reduce the data volume for known roads, while providing dense sampling for unmatched traces. Communication cost is of great economic concern to location tracking systems, and many use low sampling rates for this reason [55], which impair map update efficiency and accuracy. Using CrowdAtlas app guarantees high-quality map updates while incurring negligible extra communication—proportional to the number of missing roads on the map, not the size of the fleet. This provides dense sampling only where the server needs it, which is far better for map update than sending all traces at a uniformly low sampling rate. The user can also opt to send only unmatched segments to the server, reducing privacy exposure somewhat but losing the opportunity to correct existing roads. Standalone operation further avoids data upload and internet connection altogether.

Figure 1b shows the architecture of the navigation app. From the user's perspective, it can be viewed as a navigation system integrated with a mini CrowdAtlas server for standalone use. It has an interactive mode of operation to enable users to map out an area or a road system themselves. The clustering step is omitted, so that one need not retrace a road multiple times to generate sufficient data. Instead, it fits a road right away after detecting an unmatched subtrace and asks for user confirmation. Standalone updates can only add new roads and other map features; it would be risky to reposition roads based on a single user's trace. Updating known roads is only enabled on the server, as it should require evidence from multiple users to justify.

3. MAP MATCHING

We first introduce the basics of the Viterbi map matching algorithm and then discuss our customization for the purpose of map update.

3.1 Background

Almost all map matching algorithms consist of two steps: i) finding a set of candidate road match locations within the error radius of a sample, and ii) selecting a sequence of candidates to form the matched path maximizing some criterion. Various algorithms differ in the second step on the selection criteria, which can be largely divided into two groups: incremental methods [38, 53, 57], where the algorithm picks the current best candidate based on previous observations only, and global methods [34, 40, 51], where the algorithm picks a whole sequence at once, typically using Viterbi dynamic programming. In general, real-time (online) map matching requires incremental algorithms, while global algorithms are more accurate for offline map matching.

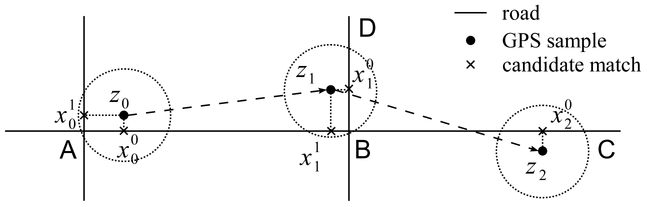


Figure 3: Map matching illustration

Figure 3 shows an example. GPS samples z_0, z_1, and z_2 have candidate sets $\{x_0^0, x_0^1\}$, $\{x_1^0, x_1^1\}$, and $\{x_2^0\}$ in their error circles, respectively. Upon observing z_1, the most likely drive path is ABD since they are closest to the samples. However, after observing z_2, the most likely path should be ABC.

In our prior comparative study [56], global algorithms based on *Hidden Markov Models* (HMM) are the most accurate for offline map matching. In particular, the best performers are the HMMs whose transition probability incorporates the length of the shortest path between two consecutive candidates [34, 40]. These models are especially robust against long sampling intervals, even up to a minute between samples.

The CrowdAtlas server requires only offline map matching. We use the HMM-based Viterbi dynamic programming algorithm described in [56], which improves upon [40] and is the most accurate in our prior comparative study. In a nutshell, taking a sequence of GPS samples as input, the Viterbi algorithm calculates the probabilities for all candidate matches of each sample in a forward loop. Next, starting from the candidate with the maximum probability of the last sample, a backward loop recovers the most likely path taken by the host generating the GPS sequence. CrowdAtlas server breaks incoming GPS streams by time windows and matches the streams section by section. By contrast, the CrowdAtlas navigation app requires real-time map matching. In this case, we simply let Viterbi output the candidate match with the highest probability immediately during the forward loop. The result is identical to a *Bayes filter* applied to the same HMM formulation. Map matching using a Bayes filter applied to a different HMM formulation has been proposed previously [38]. After a mismatch between the map and the GPS is detected, as discussed in the next subsection, the CrowdAtlas app invokes the backward loop of the Viterbi algorithm to more accurately recover the unmatched segment. For this purpose, it keeps a window of previous observations in case it needs to go back for a mismatch.

3.2 Extracting Unmatched Segments

The HMM-based map matching algorithm discussed in the previous subsection can detect two types of mismatches between GPS traces and road maps. Type I mismatch is when a sample's error radius does not intersect any road. Type II mismatch is when two consecutive matches are disconnected on the existing map or the shortest path between them is too long to be traversed by the host within the sampling interval. In CrowdAtlas we set the maximum travel speed to 180 km/h; therefore, any consecutive samples matched to locations beyond $50t$ meters from each other are considered a mismatch, where t is the sampling interval.

GPS signals are inherently noisy. In addition, roads vary in width, and maps can be misaligned. Ideally, we want to match GPS samples to the right roads even in these cases, but not miss any unmatched segments. To achieve these two opposing goals, we develop a novel technique of applying different error radius thresholds to matched and unmatched segments. The error radius for candidate search in the map matching algorithm is 50 m, which doubles the maximum error in a Seattle dataset [56] and is used by other map matching algorithms [40]. On the other hand, when mismatches are detected, we extend the mismatched segment both ways in a postprocessing step until the samples are within 15 m of roads, a threshold needed to cover road width based on our visual inspection as well as empirical evaluation in Section 6.

Applying different error radii to matched and unmatched segments separately, CrowdAtlas can accurately differentiate new roads from noise. On one hand, noisy GPS samples within 50 m of the road are always matched, which well covers Gaussian GPS noise and map misalignment problems. On the other hand, a new road is not missed as long as a section of it is more than 50 m away from existing roads (because GPS samples generated from that section are Type I mismatches, and we extend them to cover the rest of the new road until within 15 m of the connecting roads). Even if the entire new road is within 50 m of existing roads, there is often a Type II mismatch detected, because connecting the start and end locations using existing roads would be very circuitous.

In our experimental study, CrowdAtlas never missed any new road traveled by the probe, but there are outlier samples and traces well beyond 50 m to the path taken, e.g., the long line segments in Figure 2a. Compared to normal GPS noise or noise due to misaligned maps, these outliers are very sparse and never formed a cluster.

3.3 Map Matching in the CrowdAtlas App

As discussed in Section 3.1, CrowdAtlas app employs the same Viterbi algorithm for map matching and unmatched segment extraction, but it also outputs the most likely candidate immediately after each observation for navigation.

Another important feature we implement in CrowdAtlas app is dynamic sampling, which applies a high sampling rate to unmatched segments and a low sampling rate to matched ones. Without sacrificing map update efficiency and accuracy, dynamic rates significantly reduce the data communicated over the network, saving both energy and communication fees. The latter is a crucial concern for many location tracking services, which typically sample at one minute or longer just to save cost [55]. The extra cost CrowdAtlas incurs is proportional to the number of new roads generated (modulo noisy samples) and is not affected by fleet size or drive distance if server updates propagate frequently.

We apply dynamic rates only to the data transmitted to the CrowdAtlas server. For the purpose of navigation, low sampling rates would degrade the user experience and would not save energy since existing mobile phones do not turn off their GPS units for short duty cycles [42, 51]. However, many people use smart phones for navigation already. In this case, the additional power needed is for new road inference in standalone mode or data transmission in server mode. The former is negligible since the inference typically takes less than half a second. The latter is also immaterial because dynamic sampling significantly reduces the data to be transmitted, eliminating the bulk of the points on existing roads.

Most GPS-equipped phones provide the accuracy estimate of the sample, e.g., `getAccuracy()` in Android returns the standard deviation of the GPS measurement in the current location, assuming Gaussian noise. We use this information to better adjust the error radius dynamically. More specifically, instead of using two fixed error radii as in the server, our app sets the radius to be twice the standard deviation returned by `getAccuracy()`, with a lower bound of 15 m to cover road width, the same used in the server. If twice of the standard deviation is beyond 50 m, we simply discard the sample since it is likely noise. This dynamic error radius feature is crucial for detecting walking/cycling trails, which often closely parallel drive roads.

4. MAP UPDATE

We describe new road inference, other map updates, and challenges and limitations of CrowdAtlas.

4.1 New Road Inference

CrowdAtlas infers new roads using our prior clustering-based map inference algorithm, which has been shown to achieve good road coverage even for noisy and sparse data [32]. In this algorithm, each cluster contains line segments of consecutive samples that are likely generated from one road, judged by their location proximity and direction similarity. The road centerline is extracted from each cluster using B-spline fitting. We customize this algorithm for the goal of map update rather than new map inference, described below.

Trace clustering: In CrowdAtlas, the element in each cluster is an entire unmatched segment, i.e., a sequence of unmatched samples, rather than a line segment of a consecutive pair. The distance between two unmatched segments is the *Hausdorff distance* of the two trajectories, i.e., the longest distance between a sample of one segment and the trajectory of the other. Clustering whole unmatched segments filters out noise effectively, since multiple traces have to follow the same trajectory to form clusters, which does not happen for erratic, noisy samples. In our prior work, as well as other popular *Kernel Density Estimation* (KDE) based approaches, portions of noisy traces sometimes locate together and lead to false positive roads [12, 32]. Hausdorff distance considers only the longest distance between two traces, which is rather aggressive in noise filtering. A softer distance measurement that considers percentile or probability may include traces with a few noisy samples in the cluster, in which case we would need more robust centerline fitting algorithms. This is a typical tradeoff between road coverage and precision. We prefer the latter for the purpose of automated road update. We apply a streaming variant of the *single linkage clustering* algorithm [48]. Each newly extracted unmatched segment is compared with existing segments to calculate the Hausdorff distance. For those segments within a distance threshold to the new one, their respective clusters are merged and augmented with the new segment.

Centerline fitting: CrowdAtlas fits a road to a cluster once its number of traces exceeds a support threshold. We apply a *polygonal principal curve* algorithm to fit the set of samples in the cluster [31], which generates a polyline to minimize its mean square error to the samples. In our experiments using real taxi traces from Beijing, this algorithm is more accurate and more compatible with existing maps than alternative methods such as B-spline fitting [45] and KDE [12, 20]. B-spline fitting models roads by smooth polynomial functions, which is often overly smooth for straight roads or sharp turns. Even for curved freeways, B-spline fitting is overkill since almost all digital maps approximate curves by polygonal lines, and CrowdAtlas must be compatible with these maps in order to update them. KDE is good at approximating road centerlines from samples with Gaussian noise. However, it requires sufficient samples evenly distributed over the road, which requires substantially more data for the same degree of road coverage [32]. After polyline fitting, we apply the Douglas-Peucker algorithm [21] that is widely used in *cartographic generalization* and other map inference work [12] to remove unnecessary intermediate nodes.

Connection: After fitting a centerline to a cluster, the connectivity between the new road and existing ones is determined by the matched samples immediately before and after the unmatched segments in the cluster. These samples indicate which roads the new road connects to, called *anchor roads*, but not the exact intersection location unless the sampling rate is sufficiently high, e.g., 1 Hz. Instead of using the matched locations of these samples, we find the intersection via a set of rules. First, if there is already an intersection near the end node of the new road, CrowdAtlas connects the new road to this existing intersection. If there is no intersection nearby, CrowdAtlas extends the last line segment of the new road until it intersects with the nearest anchor road. We use the nearest anchor road because there can be multiple anchor roads at low sampling rates, e.g., the trace completely skips a short road after the unmatched segment. However, it is also common that a new road connects to a major road represented by two one-way roads of opposite directions on map. Therefore, if there is an anchor road parallel to the nearest one, CrowdAtlas further extends the last line segment to intersect the parallel road of opposite direction.

Iteration: Once a new road is generated and added to the map, future GPS traces following the road will be matched and no new unmatched segment will be extracted. However, existing unmatched segments in other clusters may partially overlap with the new road. We rematch and re-cluster these segments for iteratively more road inference. Figure 4 shows an example, where a whole block of roads is missing from OSM in an urban canyon area. The cluster corresponding to Road 1 first exceeds the support threshold of three and the road is added. Unmatched segments overlapping Road 1 are rematched and re-clustered. Road 2 is generated next. Upon its inference, Road 3 is immediately added after rematching and re-clustering because it exceeds the threshold. The unmatched segments corresponding to Road 3 did not form a single cluster earlier because they do not follow the same road beyond Road 3. Road 4 is added next. The rest of the unmatched segments do not form clusters of sufficient support because they do not fully overlap or are too noisy to be within the Hausdorff distance to another trace, which

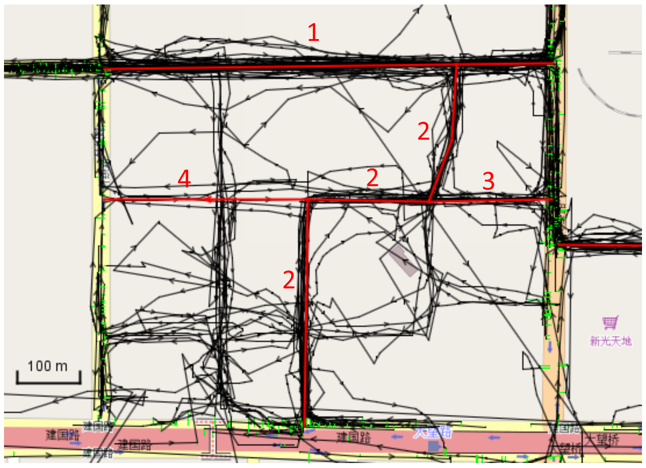


Figure 4: Iteratively inferring a block of missing roads

we set to 20 m based on our empirical study. All missing roads in the block will be added eventually with more data.

Road attributes: CrowdAtlas infers the directionality of the new road by checking whether all unmatched segments of the cluster follow the same direction. It infers the type of the road (e.g., freeway, primary, or residential) by comparing the speed profile and traffic frequency of the new road with those of the existing roads, and picks the most similar type. The number of lanes could be inferred from GPS traces as well [18]. Finally, there are computer vision techniques that could extract road names and traffic signs from street views [36], which could complement CrowdAtlas to produce fully-fledged digital maps.

Standalone mode: CrowdAtlas app follows a similar procedure to infer new roads in its standalone mode. However, instead of aggregating multiple GPS traces for high confidence, this app adds each new road immediately, given user confirmation. The road type (e.g., driveway, cycling way, or footpath) is determined by the travel mode set by the user in the app. There are techniques that infer road types in more comprehensive ways [61], which CrowdAtlas could exploit.

4.2 Other Map Updates

In addition to new roads, CrowdAtlas infers other missing map features from unmatched segments. Even in commercial-grade maps, missing intersections are quite common [56]; i.e., the map data fails to connect two roads with a connecting node, preventing navigation aids from considering any route from one to the other. These errors are not visually detectable on the rendered raster map if the roads overlap or come extremely close. Fortunately, CrowdAtlas reliably detects missing intersections as Type II mismatches in its map matching process. When the number of such mismatches between two close roads exceeds a threshold, CrowdAtlas connects them at the corresponding location, typically using the existing intersecting point of the two road polylines. Similarly, new turn possibilities of an existing intersection can be inferred. Incorrectly marked one-way roads also incur Type II mismatches. If the inferred road does not overlap with an existing one-way road—indicating that the two opposite ways are well separated as typical of major roads—then we add the inferred road. Otherwise we simply change the one-way mark of the existing road to two-way.

In order to update the geometry of existing roads, CrowdAtlas keeps for each a limited buffer of recently matched traces. Periodically (e.g. daily), if the buffer has enough newly accumulated traces and the median signed offset from the road centerline exceeds a threshold, a revised geometry is generated by the same centerline fitting algorithm as for new road inference. Whether to refine an existing road or to generate a new road is determined by the error radius in map matching, discussed in Section 3.2. A road constantly within the radius to the correct location is refined, otherwise a new road is generated.

A closure of a road segment between intersections is detected by a surprising lack of newly matched traces. Each segment establishes an appropriate timeout proportionate (3x) to the maximum time between vehicles in a training window. No timeout is set for a segment until it has accumulated at least a week of data and at least five traces. Thus, most residential roads have no timeout established. A reopening event is detected by a matched trace arriving for a road segment believed closed.

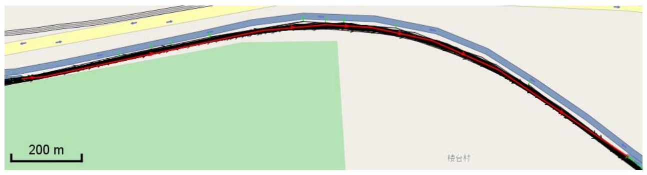
Merging new roads generated by both standalone operation and the server could result in near-duplicates. To avoid this, CrowdAtlas app employs the iterative rematching and fitting process used by the server. Roads created in the app save the original unmatched trace that generated the road; later these are matched against the updated map from the server. Standalone updates can be removed or updated if these traces fully or partially match in the new map.

4.3 Challenges and Limitations

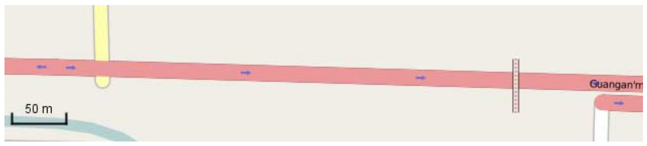
Inferring new roads in an existing map is significantly easier than building from an empty map, since we only need to focus on unmatched trace segments, which are relatively short and easy to locate. However, CrowdAtlas still faces several challenges and limitations. First, the base map must have a reasonably accurate skeleton of arterial roads. Without a base map or if the base map is too sketchy, unmatched segments become lengthy, and it takes a long time for them to form clusters with sufficient support. These clusters tend to cross multiple roads in a real map, which reduces readability. Road 2 in Figure 4 is an example; it would likely be covered by three different roads in a manually created map.

In the introduction, we listed tunnels and plane separated roads as inherent limitations for map inference using only latitude and longitude. Hopefully these road features are already covered in the base map, since they are typically on major roads. In practice, accurate maps of arterial roads are relatively easy to obtain. For example, in OSM, developed countries are relatively well covered by public or out-of-copyright data, and a commercial donation from Automotive Navigation Data covers major truck roads in China and India.

Another major challenge for GPS-based map inference is user-friendly map presentation. Road names and address numbers cannot be added automatically. A more serious problem is whether to represent a major road by one two-way polyline or two one-way polylines in the map data. Four or more separate polylines are common for parallel expressways and freeways. Existing map inference work always makes a single representation choice for all roads.



(a) CrowdAtlas infers a missing segment of a divided freeway



(b) A double-polyline major road with both wrong direction marks and missing segments on OSM

Figure 5: Double-polyline roads are a challenge

However, map updates must be compatible with existing manually created maps, which commonly contain different choices in different places. Figure 5a shows an example where CrowdAtlas correctly inferred a missing segment in one direction of a divided freeway. Figure 5b, however, is a difficult case because it is missing a segment in one direction, the direction mark of the other is backwards, and the two opposite ways are much closer. Without both high-precision and high-frequency data CrowdAtlas may connect the inferred eastbound missing segment to the existing upper polyline since it is consistent in direction, and it is closer than the lower road stub. Fortunately, direction errors of arterial roads are rare and easy to detect because they quickly incur many unmatched traces. Figure 5b is the only error of this type we found on the Beijing OSM map. Using CrowdAtlas app for trace collection can also solve the problem with its dynamic error radius feature, discussed in Section 3.3, which can emit unmatched segments covering the whole missing eastbound segment like Figure 5a, under normal GPS measurement conditions (small error radius).

5. IMPLEMENTATION

We implemented CrowdAtlas and its app in Java. Our implementation leverages many open source tools. We use OSM and its tool suite for the base map and its relevant read/write, database import/export operations. We also use JMapView, a GUI of OSM, to visualize and verify the results, as well as capture the screenshots in this paper. Our road fitting algorithm leverages the implementation of the polygonal principal curve algorithm by permission of its author [31]. Our software development focused on the algorithms, described in Sections 3 and 4, and scalability, described below.

Map matching is the bottleneck of CrowdAtlas, because it involves the Viterbi dynamic programming algorithm with a complexity of $O(nm^2)$, where n is the number of observations and m is the number of states of the HMM. This algorithm further invokes the costly shortest-path calculation for each match candidate of each sample. We have heavily optimized our implementation in our prior work [56], and it is one of the best performers in the 2012 GIS Cup competition for both speed and accuracy [7]. Nevertheless, it is still orders of magnitude slower than the other processing steps of CrowdAtlas. Furthermore, map

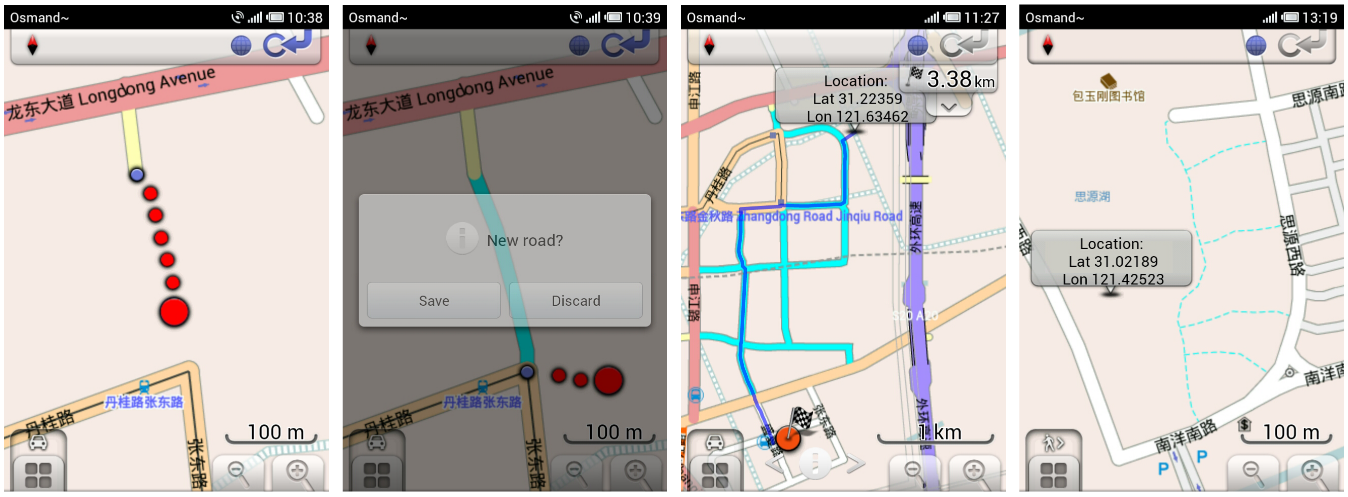


Figure 6: Screenshots of CrowdAtlas app: i) OsmAnd map missing roads in an area of Shanghai Pudong (plus unmatched GPS track overlaid); ii) user confirmation after traversing the new road (cyan, new red dots are shown as the user continues on another missing road); iii) map after generating multiple roads, plus navigation path (dark blue) showing the use of these new roads; iv) building a walking/cycling map of SJTU campus, where the lower left icon indicates the transportation mode.

matching must scale as more GPS data becomes available, while the other tasks only need to scale with the map, which is relatively stable.

Fortunately, map matching can be easily parallelized by partitioning the input GPS streams using a spatial grid or political boundaries. We implement both multithreaded and distributed map matching for different scales of data. The latter is based on Storm [4], a scalable distributed stream processing platform. By contrast, only a single workstation is needed for all the map update nodes in Figure 1a. Having only a single map writer simplifies our implementation.

Each map matcher node caches the map in memory for fast spatial indexing and shortest-path calculation. The cache in map matcher can be slightly outdated because of the delay between the inference of a new road and its propagation back to map matcher through the map database. Outdated caches may only produce extra unmatched segments for already inferred roads. CrowdAtlas uses the road version number in OSM to detect possibly outdated unmatched segments, and rematches them using the latest map in the new road inference node. When a new road is inserted, the version numbers of its anchor roads on both ends are incremented. An unmatched segment is rematched if at least one of its anchor roads is not current. This rematching step piggybacks on the existing iterative fitting process of new road inference, discussed in Section 4.1, which indexes new roads and unmatched segments for fast computation.

CrowdAtlas does not store its input GPS streams in persistent storage, and data loss is possible upon failure, which includes both the input and intermediate results. However, each node can always restart independently, and the map updates will eventually occur as new evidence builds up over time.

CrowdAtlas app is based on OsmAnd, an open source navigation app for Android using OSM maps [3]. There are many map editing tools available for OSM map formats, allowing users to add road attributes and other information. Based on the dynamic sampling feature discussed in Section 3.3, when the trace is matched to the map, CrowdAtlas app samples sparsely and sends each one to the server right away to facilitate real-time applications such as traffic monitoring. This is similar to many fleet management systems where the vehicle location is transmitted periodically through SMS. On the other hand, unmatched trace segments are densely sampled and buffered until they again match to an existing road, triggering batch transmission. Alternately, it could buffer all samples and transmit only when a 3G connection is available. In standalone mode, a popup window asks for user confirmation when a new road is detected. We simply set a timeout for the popup window, in which case the new road is discarded.

6. EVALUATION

We evaluate the standalone and server map updates of CrowdAtlas separately.

6.1 Standalone Map Update

Figure 6 shows a series of screenshots from the CrowdAtlas app. The first three were captured during an experiment in a recently developed area of the Pudong district where a block of roads are largely missing. We took a drive to complete the map systematically. The first screenshot was captured while driving a missing road. The second screenshot is the confirmation pop-up after generating the road (cyan). The third screenshot is the final result with all the roads completed. We picked a pair of locations for navigation to confirm that the roads generated are navigable, as shown by the dark blue line. There are nine roads inferred in total, with lengths ranging from 171 m to 1534 m. The inference and map save time added together for each road is between 110ms and 459ms, a minor delay for a smooth user experience, and negligible extra energy consumption for those who use navigation apps already. While waiting for user confirmation, CrowdAtlas app continues to infer new roads if there are more unmatched segments, as illustrated in the second screenshot. Therefore, the user can drive around

Figure 7: Photo of the shortcut footway from the entrance to the library in SJTU.

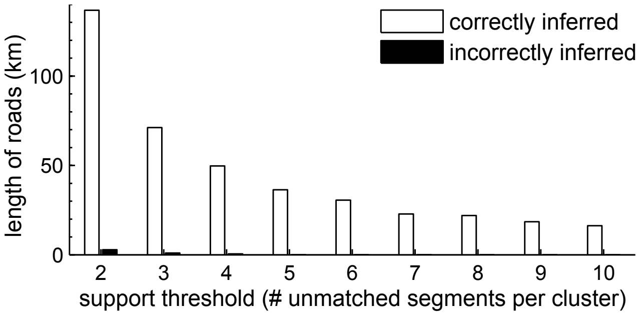
an area to map out missing roads and then confirm them in a batch.

The whole process took only 27 minutes. In comparison, other offline navigation apps or devices typically have an annual map update cycle, and there is no guarantee you get the update you want in your area. Google map in the area has a few quality issues. Its satellite view is misaligned, and one of the roads we generated is shown as only a road stub. It was probably surveyed before construction was complete. The Baidu map is more consistent with our result. Interested readers can verify using the lat/lon provided in the third screenshot.

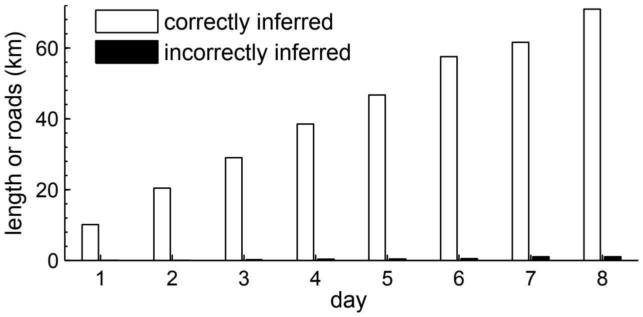
In another experiment we aimed to build a cycling/walking map for the campus of Shanghai Jiao Tong University. The campus has an area of 3.2 km^2, and it is gated to limit motorist traffic. However, there is no cycling/walking map, and almost every student has been asked for directions by visitors numerous times. We mapped the campus on foot by setting the travel mode to *pedestrian* in OsmAnd, indicated by the icon in the lower left of the rightmost screenshot of Figure 6. There is a large amount of work on automated travel mode detection [43,61], which OsmAnd could leverage to automatically change the travel mode. The screenshot shows a small part of the campus, from its main entrance (southern end) to the library (northern end). The OSM map includes only the driveway connecting the entrance to the library, which is a detour compared to the shortcut footway used by most students, shown in Figure 7. The screenshot shows this vertical shortcut, which saves walking time by 34%, as well as another four horizontal footways connecting it to the vertical driveway on the right. CrowdAtlas app inferred these roads correctly after we walked them. The samples along all horizontal footways are within 50 m to either the driveway or the shortcut inferred earlier. Without the dynamic error radius feature, samples would have always matched to either road and no new footway would be detected. Comparing with commercial maps, Google map is also missing these footways. While Baidu map contains some of these footways, they are incorrectly marked as driveways that can be used for vehicle navigation.

6.2 Server Map Update

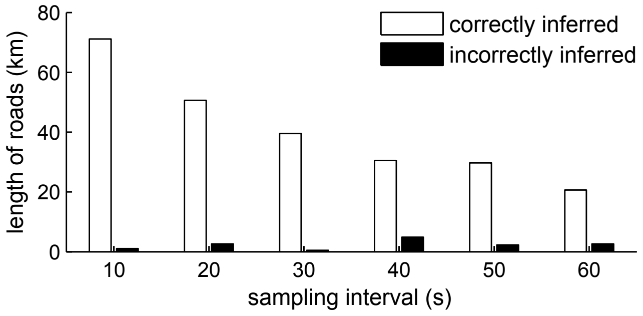
Our evaluation of server map update uses taxi traces collected from 70 taxis in Beijing for 8 days in December 2008, a total of 4,351,977 samples. The dataset was sampled at 10 second intervals, which is relatively frequent and helps in the verification of map matching results. We also sub-



(a) New roads inferred vs. different support thresholds



(b) New roads accumulated by day (support = 3)



(c) Using sub-sampled data (support = 3)

Figure 8: New road inference

sample the data to evaluate the performance of CrowdAtlas under long-sampling-interval data. The standard deviation of the Gaussian GPS noise is within 10 m under normal conditions, but typical outliers exist, especially in urban canyon areas, e.g. Figure 4 (detailed characteristics of the dataset are available in [55]). We use the OSM map of 10/31/2012 to evaluate our map update results, and compare them with an earlier map and the latest map to understand how the manually created OSM map evolves.

After map matching, only 37,355 unmatched samples remain in 4,968 unmatched segments. Thus the CrowdAtlas app could have reduced the amount of data communicated to less than 1%. Figure 8 shows the new road inference results. We verified our generated roads manually using both Bing aerial images and Google maps (which agree with each other in most cases except for two inferred roads, which are consistent with the aerial image but still missing from Google maps). Figure 8a shows that increasing the support threshold can eliminate incorrectly inferred roads, mostly due to noise or insufficient data, leading to errors

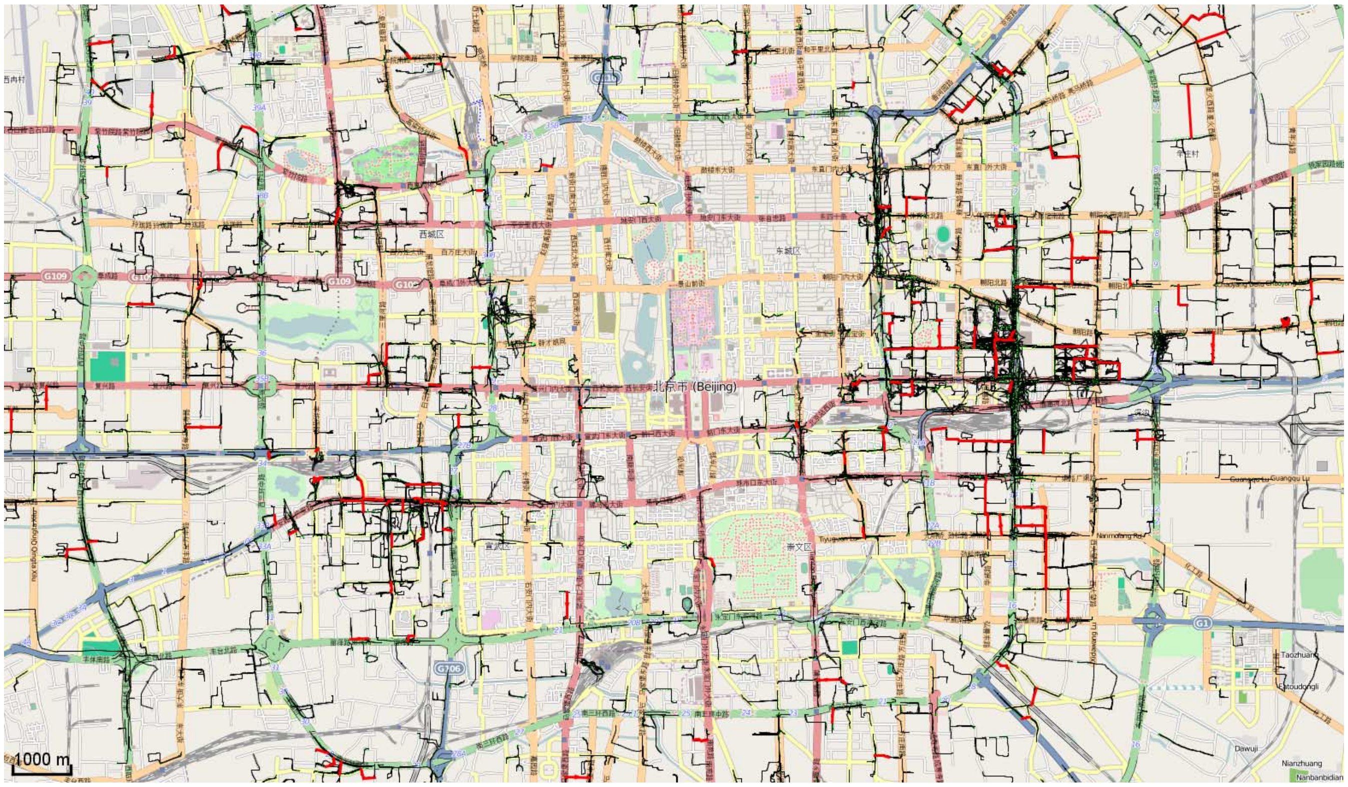


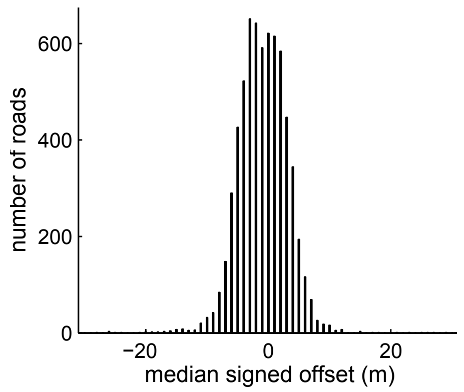
Figure 9: New roads (red) inferred from unmatched trace segments (black) overlaid on the latest OSM base map

in road geometry. At the support threshold of three (or four) traces, there are only six (or three) incorrectly inferred roads, totaling less than 1 km in length in either case; a support threshold of five or more eliminates false positives altogether. Given our limited data volume, we also considered even lower thresholds that would not be viable in production. Using a threshold of two almost doubles the length of roads inferred, albeit with more false roads. Figure 8b shows that new roads increase linearly as time progresses, which indicates that our dataset is not sufficient to complete the whole city map. Finally, Figure 8c shows that longer sampling intervals can reduce the number of roads inferred and increase noise. Up to 30-second intervals CrowdAtlas works reasonably well and the false roads can be eliminated by slightly increasing the support threshold. Beyond 30-seconds, the ratio between correct and incorrect roads significantly decreases, because of the increasing errors in map matching and trajectory-based clustering.

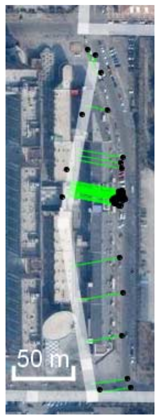
Figure 9 shows all unmatched segments and new roads inferred in central Beijing using all eight days of data, with a support threshold of three. There are a few urban canyon pockets where unmatched traces are quite noisy, but the clustering step finds the right traces to infer new roads. The rest of the unmatched segments, especially further from central Beijing, are indeed from missing roads but lack the support to trigger the fitting process. In summary, the inferred roads showed good accuracy as well as reasonable coverage considering the volume of data, which naturally would grow over time.

Note that the backdrop raster map in Figure 9 was retrieved from OSM in early December, which includes manual edits people had entered online after the date of our map. We compared our inferred roads to these online edits, noting that Beijing is one of the most actively updated areas in OSM. Their map of 12/3/2012 with over a month of additional edits includes 17 of our inferred roads, a total length of 8.5 km—a modest subset of our ∼71 km using a small dataset. However, compared with an earlier map of 5/8/2012, we find several correct roads that have since been removed and not re-entered; each of these was correctly inferred by CrowdAtlas. In general, manual edits are error-prone and may not converge. There are also five inferred roads that existed back in 2008 but are not in the latest map, a total length of 1.8 km. But instead of being false positives, we verified from dated satellite images by Google Earth that most were actually detours used during construction. In addition to new roads, CrowdAtlas detected 11 incorrectly marked one-way roads, a total of 3.2 km. We have submitted 156 inferred roads not on the latest map to OSM (by userID "CrowdAtlas"), a total length of 61 km.

Next we use the matched segments to refine road geometry. Among all 25,555 roads in the map of 10/31/2012, 13,303 have no sample matched, partly because of the limited data volume. There are also many roads for non-motorist traffic only, and roads in rural areas of Beijing not frequently visited by taxis. Finally, some of the roads are constructed after 2008. For the rest of the matched roads, we consider those visited by at least three different taxis, with at least ten samples matched, and at least five samples matched to locations more than 10 m apart from each other. The last requirement ensures that samples are spread out for correct centerline fitting. There are 6,568 roads satisfying



(a) Histogram

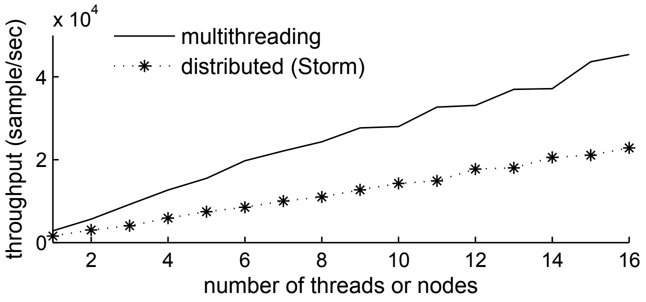


(b) Example

Figure 10: Geometry refinement of existing road

	map matching	new road	refinement
time (s)	1,544	67	2

(a) Baseline computation time using a single thread



(b) Parallel map matching with memory-cached maps

Figure 11: Computation time and scalability

these requirements, and Figure 10a is the histogram of the median signed offset from the centerline. There are only 32 roads with the absolute median larger than 15 m. Our visual inspection suggests that 15 m is a reasonable cutoff threshold for well aligned roads. Within 15 m it is hard to judge which is more accurate: the samples, the vector map, or the aerial image. We manually inspected these 32 roads and found 19 of them are due to missing parallel roads or missing connections between parallel roads. For example, if the express freeway is included in the map without representing the local freeway by separate polylines, samples generated from local freeways are matched to express freeways, since they are typically within our error radius of 50 m for map matching. These samples would exhibit a significant offset. Using CrowdAtlas app for data collection can solve the problem with its dynamic error radius feature, which would extract the trace segment as unmatched rather than matched under good GPS measurements. Another 8 roads are due to insufficient data such that the whole matched sample set is dominated by noise, caused by high-rise buildings or covers, e.g., elevated roads and airport driveways. The remaining five roads are indeed misaligned; three of them are due to earlier construction; one of them has been manually corrected in the latest map of 12/03/2012. The last one is shown on Figure 10b, where the vertical road shown in white is clearly misaligned with both the samples and the aerial image. Without CrowdAtlasit would be very challenging to detect one misaligned road in a city-scale map.

Finally, we examine the computation time and the scalability of CrowdAtlas. Figure 11a shows the baseline computation time of different tasks using a single core on a dual Xeon E5 2650L 1.8G Hz workstation. We used the full Beijing dataset, and the support threshold for new road inference is again three. Map matching is orders of magnitude slower than new road inference and geometry refinement. New road inference takes much longer than refinement because there are lots of missing roads. Inferring each road takes less than half a second on average. Therefore with stable maps, new road inference should be lightweight, and one server ought to be able to handle global updates. Figure 11b shows the scalability of map matching by multithreading, using the same Xeon workstation, or distributed computing, using 16 PCs. In each case, the map is fully cached in memory so there is no communication cost or database overhead with the map server. Not surprisingly, the throughput scales almost linearly since each thread or node operates on the input trace independently. Using the Xeon workstation, the throughput is roughly 2,800 samples per second per thread. At the sampling interval of 10 seconds, this throughput implies the scale of 28k vehicles per core, or 448k vehicles in total for the workstation. Longer sampling intervals may require more processing time for the shortest-path calculation. Nevertheless one workstation is more than sufficient for map matching city- or state-scale data.

7. RELATED WORK

Map matching: Section 3.1 mentioned incremental map matching algorithms using only previous observations [38, 53, 57], and global algorithms using Viterbi dynamic programming [34, 40, 51]. Incremental algorithms perform poorly at narrow Y-splits or close parallel roads, where future observations are often needed to correctly match the current sample to one of the two extremely close roads. There are also map matching algorithms that minimize the geometric measure of *Fréchet distance* between a GPS trace and a path on map [8, 14]. These algorithms are inaccurate at long sampling intervals, when there are too many alternative paths with the same Fréchet distance to the input trace, making it difficult for the algorithm to choose the right one [56].

Map Inference: There are three categories of methods for GPS-based map inference: K-means [6, 23, 45, 58], *Kernel Density Estimation* (KDE) [12, 16, 20, 47, 50], and trace merging or clustering [15, 30, 32, 41]. Most of these algorithms have various unrealistic assumptions of the GPS data, including low noise (e.g., only Gaussian noise with a standard deviation less than 5 m [24, 40]), high sampling frequency (e.g., 1 Hz), and uniform distribution (e.g., passing through each road to be inferred exactly once). Their performance deteriorates rapidly if any of these assumptions does not hold; see the comparative studies for detailed analyses [11, 32]. The only exceptions are the trace clustering algorithm [32], which CrowdAtlas improves upon, and a variant of a KDE-based algorithm [32] and its recent extension [12]. The KDE-based methods estimate road centerlines from GPS samples alone, which is particularly good for sparse and noisy data. Road connectivity and attributes are inferred by map matching GPS traces to the inferred roads. Our clustering-based method is simpler since each cluster contains all the information needed to infer a new road and insert it into the map. Furthermore, our trace-based clustering is more robust against random noise since traces have to tightly overlap to form clusters, while KDE would infer roads from noise samples if they accumulate in one area. Finally, it is straightforward to adapt our clustering algorithm to streams, while the KDE-based methods are designed for offline processing. Our road fitting step, however, could use KDE as an alternative, which may be more robust than principal curve estimation but requires full coverage of samples on the road.

Another category of automated map inference is based on aerial imagery [28, 46]. Roads covered by tree canopy and road connectivity are very challenging for these methods. Currently most contributors of OSM create maps from aerial imagery manually, and missing connections are frequently identified by CrowdAtlas in our experiments. Also, obtaining frequently updated, high-resolution aerial images is very expensive, so updating maps periodically using aerial imagery is not economical. Valuable future work could combine CrowdAtlas with aerial imagery analysis to increase the confidence of map updates.

Relevant Systems: Google, TomTom, and Waze leverage crowdsourcing for map update, where user-submitted changes can be integrated into their map products after manual review [39]. OpenStreetMap is the largest crowdsourcing map community. All of its user-contributed maps so far are manually created using its GUI tools. There are also numerous services, mobile apps, and GPS devices that allow users to store and share their GPS traces, but they do not create navigable maps. There is considerable interest in mining large-scale GPS trace sources for various transportation related tasks, e.g., traffic monitoring [27, 59], incident detection [49], empirical shortest-time route calculation [60], and empirical taxi fare calculation [10]. It is straightforward to incorporate these systems into CrowdAtlas and benefit from its scalable and reliable map matching engine. There are also systems designed to store, compare, and query GPS traces [26]. CrowdAtlas currently does not store traces after map update, but can use these systems to do so, and augment each trace with the map matching result to benefit other applications.

8. CONCLUSION

CrowdAtlas offers a solution to the ongoing problem of digital road map inaccuracy—automation to fix errors, reduce lag times for changed roadways, and keep up with the growing demand for specialized maps. By transmitting high frequency samples to the server only when off the known road network, our mobile app enables new roads to be accurately inferred with a minimum of communication overhead. Because the server only infers roads when several unmatched vehicle traces cluster tightly together, the roads produced have high precision and do not require a specialist to edit and integrate them, yielding a high level of automation.

9. ACKNOWLEDGMENTS

We thank our shepherd Gaetano Borriello and anonymous reviewers for helping us improve the paper. Yang Yue provided us the taxi dataset from Beijing, for which we are grateful. We also acknowledge the support of the HP Labs Innovation Research Program (CW267311), the Shanghai Pu Jiang Talents Program (10PJ1405800), NSFC (61170238, 60903190, and 61027009), the Doctoral Fund of the Ministry of Education of China (20100073120021), and the 863 Program (2009AA012201 and 2011AA010500).

10. REFERENCES

[1] GPS dataset. `grid.sjtu.edu.cn/mapupdate/`.
[2] NAVTEQ. `navteq.com/company_history.htm`.
[3] OsmAnd: OSM Automated Navigation Directions Android app. `www.osmand.net`.
[4] Storm: Distributed and fault-tolerant realtime computation. `storm-project.net/`.
[5] TomTom Map Update Service. `www.tomtom.com/en_gb/maps/map-update-service/`.
[6] 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.
[7] M. Ali, T. Rautman, J. Krumm, and A. Teredesai. ACM SIGSPATIAL Cup 2012. In *GIS*, 2012.
[8] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.
[9] J. Angwin and J. V.-Devries. Apple, Google collect user data. *Wall Street Journal*, April 2011.
[10] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. In *MobiSys*, 2011.
[11] J. Biagioni and J. Eriksson. Inferring road maps from GPS traces: Survey and comparative evaluation. In *Transportation Research Board*, 2012.
[12] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *ACM GIS*, 2012.
[13] 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 *SIGMOD*, 2010.
[14] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, 2005.
[15] L. Cao and J. Krumm. From GPS traces to a routable road map. In *ACM GIS*, 2009.
[16] C. Chen and Y. Cheng. Roads digital map generation with multi-track GPS data. In *International Workshop on Geoscience and Remote Sensing*, 2008.
[17] D. Chen, D. Chen, L. J. Guibas, J. Hershberger, and J. Sun. Road network reconstruction for organizing paths. In *SODA*, 2010.
[18] Y. Chen and J. Krumm. Probabilistic modeling of traffic lanes from GPS traces. In *ACM GIS*, 2010.
[19] T. Cook. A letter from Tim Cook on maps, 2012. `apple.com/letter-from-tim-cook-on-maps`.
[20] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

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

[22] B. Dudley. Google Maps gets bike routes, built in Fremont. *The Seattle Times*, March 2011.

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

[24] J. Eriksson et al. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys*, 2008.

[25] R. Gordon and J. Tucker. Ruling paves way for San Francisco bike lanes. *San Francisco Chronicle*, August 2010.

[26] M. Haridasan, I. Mohomed, D. Terry, C. A. Thekkath, and L. Zhang. Startrack next generation: A scalable infrastructure for track-based applications. In *OSDI*, 2010.

[27] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. B. Work, J. C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *MobiSys*, 2008.

[28] J. Hu, A. Razdan, J. Femiani, M. Cui, and P. Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE T. Geoscience and Remote Sensing*, 45(12-2):4144–4157, 2007.

[29] C. Johnson, C. Shea, and C. Holloway. The role of trust and interaction in GPS related accidents. In *Int'l Conference on Systems Safety*, 2008.

[30] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 89–98, 2012.

[31] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(3):281–297, 2000.

[32] 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*, 2012.

[33] N. Lomas. GPS driving British motorists to distraction. *Bloomberg Businessweek*, July 2008.

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

[35] S. Madden. Keynote: Going big on spatial data. In *ACM GIS*, 2012.

[36] A. C. Madrigal. How Google builds its maps. *The Atlantic*, Sept 2012.

[37] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. Parknet: drive-by sensing of road-side parking statistics. In *MobiSys*, 2010.

[38] O. Mazhelis. Using recursive Bayesian estimation for matching GPS measurements to imperfect road network data. In *IEEE Conference on Intelligent Transportation Systems*, 2010.

[39] B. McClendon (Google VP). Keynote: The path from paper to product: How spatial research reaches users. In *ACM GIS*, 2012.

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

[41] 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 *Advances in Satellite and Space Communications*, 2009.

[42] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive GPS-based positioning for smartphones. In *MobiSys*, 2010.

[43] S. Reddy, M. Mun, J. Burke, D. Estrin, M. H. Hansen, and M. B. Srivastava. Using mobile phones to determine transportation modes. *TOSN*, 6(2), 2010.

[44] S. Rogers, P. Langley, and C. Wilson. Mining gps data to augment road models. In *KDD*, 1999.

[45] S. Schrödl, 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.

[46] Y.-W. Seo, C. Urmson, and D. Wettergreen. Exploiting publicly available cartographic resources for aerial image analysis. In *GIS*, 2012.

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

[48] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[49] A. Skabardonis, T. Chira-Chavala, and D. Rydzewski. The I-880 field experiment: Effectiveness of incident detection using cellular phones. Technical Report UCB-ITS-PRR-98-1, University of California, Berkeley, 1998.

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

[51] A. Thiagarajan et al. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI*, 2011.

[52] T. Vanderbilt. It wasn't me, officer! It was my GPS. *Slate*, June 2010. `www.slate.com`.

[53] N. R. Velaga, N. R. Velaga, M. A. Quddus, and A. L. Bristow. Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683, 2009.

[54] R. Wabash. 9 car accidents caused by Google Maps & GPS. *Ranker*, April 2012. `ranker.com`.

[55] 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.

[56] H. Wei, Y. Wang, G. Forman, Y. Zhu, and H. Guan. Fast Viterbi map matching with tunable weight functions. In *GIS*, 2012. (SIGSPATIAL Cup).

[57] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal

navigation assistants. *Transportation Research Part C*, 8(1–6):91–108, 2000.

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

[59] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *MobiSys*, 2007.

[60] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-Drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Trans. Knowl. Data Eng.*, 25(1):220–232, 2013.

[61] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw GPS data for geographic applications on the web. In *WWW*, 2008.