## Title

From Pressure to Path: Barometer-based Vehicle Tracking.



## Authors

Ho, Bo-Jhang
Martin, Paul
Swaminathan, Prashanth
et al.

## Publication Date

2015-11-01

## DOI

10.1145/2821650.2821665

Peer reviewed

Published in final edited form as:

*BuildSys15 (2015)*. 2015 November ; 2015: 65–74. doi:10.1145/2821650.2821665.

# From Pressure to Path: Barometer-based Vehicle Tracking

**Bo-Jhang Ho**[†], **Paul Martin**[†], **Prashanth Swaminathan**, and **Mani Srivastava**

University of California, Los Angeles, Los Angeles, California

## Abstract

Pervasive mobile devices have enabled countless context-and location-based applications that facilitate navigation, life-logging, and more. As we build the next generation of *smart* cities, it is important to leverage the rich sensing modalities that these numerous devices have to offer. This work demonstrates how mobile devices can be used to accurately track driving patterns based solely on pressure data collected from the device's barometer. Specifically, by correlating pressure time-series data against topographic elevation data and road maps for a given region, a centralized computer can estimate the likely paths through which individual users have driven, providing an exceptionally low-power method for measuring driving patterns of a given individual or for analyzing group behavior across multiple users. This work also brings to bear a more nefarious side effect of pressure-based path estimation: a mobile application can, without consent and without notifying the user, use pressure data to accurately detect an individual's driving behavior, compromising both user privacy and security. We further analyze the ability to predict driving trajectories in terms of the variance in barometer pressure and geographical elevation, demonstrating cases in which more than 80% of paths can be accurately predicted.

### Categories and Subject Descriptors

G.3 [Probability and Statistics]: Time series analysis; I.5.4 [Pattern Recognition]: Applications—*Signal processing*; K.6.5 [Security and Protection]: Invasive Software

### General Terms

Algorithms; Experimentation; Security

## 1. INTRODUCTION

The increasing ubiquity of mobile computing devices has been accompanied by new sensing modalities and data fusion methods to sense or infer a wide array of physical phenomena and stimuli. The result of this is a distributed mobile sensor network whose sensors can yield information about users' behaviors and surrounding environments. Inferences made from



[†]Equal contribution authors

these mobile sensors often provide valuable services such as accelerometer-based motion tracking, camera-based heart rate monitors, and life-logging or quantified-self services. These services have also proven to be integral components for environmental monitoring [5], traffic analyses [15, 17], event discovery [19], and population-level analytics in general in what are increasingly referred to as *smart* cities.

Recent mobile devices have introduced yet another sensing modality in the form of barometric pressure sensors. These sensors can already be seen on mobile devices such as the Apple iPhone 6, Google Nexus 5, and Nexus 6. Barometric pressure sensors introduce a level of geographical dependency unseen in previous sensors, including magnetometers. While a magnetic compass might behave differently in different geographical locations, a barometric sensor is *designed* to differentiate between different pressures and thus, to a large degree, different elevations. In this paper, we show that the high correlation between pressure sensors and elevation allow for accurate tracking of driving patterns based on pressure from mobile devices collected by a centralized computer. Tracking vehicles in this manner provides a low power method for analyzing a user's driving behavior over long periods of time, while the power consumption associated with GPS, cellular, or WiFi positioning methods may prove prohibitively high.

In addition, both Apple's iOS and Google's Android OS treat the barometer as a non-private sensor such as an accelerometer or gyroscope. In other words, an application that wishes to read a mobile device's barometer can do so without alerting the device user. We demonstrate how this *public* data access model, while easing data collection for city-level traffic analytics, can compromise a user's privacy and security, allowing malicious third parties to estimate a user's location over time, undetected by the user. There have been a number of similar efforts that have demonstrated such nefarious inferences made from mobile sensory data. Several notable works in this vein include accelerometer touch-type keystroke identification [14, 20] and gyrophone-based microphones [13] for identifying spoken digits, such as credit card or social security numbers. These privacy and security threats are only made worse by the increasing tendency towards wearable and pervasive sensing [22]. In the face of these unceasing efforts to record and analyze personal user data, mobile computing and users thereof can no longer remain agnostic to the security ramifications of this data deluge.

In this work, mobile devices are treated as distributed sensors collecting pressure data and storing it locally. These sensors then opportunistically transmit their pressure data to a remote server whenever power and connectivity (e.g. WiFi) permit. This pressure data is then analyzed by a more capable and less power-constrained server to reveal the traffic patterns of each user. To do this, a user's coarse-grained location estimate is obtained by associating the device's IP Address with an ISP's geolocation. From there, the user's time-series pressure data is compared against a database of possible road segments and their corresponding elevation signatures. Under certain conditions regarding the uniqueness of the observed pressure data with respect to the elevation of the underlying map, this allows us to obtain a series of 'ranked' paths, ordered by descending likelihood. When a high confidence path can be obtained, the estimated path can be treated as an approximation of the user's

driving route. This provides mobile devices with a low power path logging utility for scenarios where power consumption is an important consideration.

Though many factors affect the pressure reported by barometer sensors (not the least of which are weather, air movement, and sensor drift), mobile barometric sensors can still be correlated with elevation *changes* with surprising precision. Using a simple linear model (the details of which are discussed in Section 3), height can be predicted to within an error of several meters. An example of this correlation is shown for 30 minutes of driving data in Figure 1, where the error rarely exceeds ±2 m. This correlation, however, is made difficult due to several important factors: first, the conversion from pressure to elevation is time-varying and unknown *a priori*. Second, the user's vehicle is traveling at an unknown speed, essentially 'sampling' elevation points at variable and unknown rates. This makes it difficult to directly correlate pressure data to the elevation of a given road segment. Third and most importantly, the search space of possible paths against which to compare the user's collected pressure data is immense, even given coarse grained location estimates such as those obtained from IP Address geolocation.

### 1.1 Contributions

In order to elucidate the degree to which pressure can be used to determine a user's driving path in the face of the difficulties mentioned above, we make the following contributions:

- We evaluate the accuracy with which pressure data can be used to predict the correct path from a *fixed* database of candidate paths using dynamic time warping (DTW). We also describe an algorithm for pressure-based path prediction using dynamic programming and DTW to find a jointly minimal cost path through an *arbitrary* graph of road segments.
- We evaluate the performance of our path estimation algorithms over a number of real test cases totaling 150 km and 4.6 hours of driving data.
- By modeling errors in barometer sensors and elevation estimates, we simulate path estimation results for random driving paths selected across a large number of cities with varying geographical landscapes.
- We evaluate the accuracy of our path prediction algorithms in terms of the distinctness of the pressure data with respect to the surrounding landscape, offering insights into the conditions under which driving paths can be accurately predicted.

## 2. RELATED WORK

Related work in this area can roughly be divided into two categories: low power GPS, cellular, & WiFi positioning methods, and location- or transportation-specific inference mechanisms using low power sensor data.

### Low Power GPS, Cell, & WiFi Positioning

Considerable work has gone into decreasing the power consumption of global (GPS) and regional (Cell & WiFi) positioning schemes. This includes the numerous assisted GPS (A-

GPS) techniques [6] and cloud-offloaded (CO) post-processing of raw GPS [12]. These technologies range from power consumption in the hundreds of milliwats (traditional GPS) to milliwatts (A-GPS) and even hundreds of microwatts (CO-GPS), while the latter requires highly customized GPS receiver hardware. By comparison, barometer power consumption is typically less than 10 $\mu$W [1], requiring no specialized hardware and providing a very practical tradeoff between traditional high power, high accuracy positioning techniques and ultra-low power mobile path *estimation* and analytics, where highly robust positioning may not be a hard requirement. As a comparison, a GPS module like the GlobalSat EM-506 would consume 170 mW even during a *hot start* measurement (acquisition recently acquired) and requiring up to 1 second to achieve a new fix [2]. Duty cycling such a GPS receiver to achieve 10 $\mu$W average power consumption would allow for just 1 reading every 4 hours.

### Inferring Location and Transportation

In the realm of location discovery techniques, recent work has demonstrated trajectory identification through inertial navigation (dead-reckoning) for both pedestrians [21] and vehicles [9]. Han *et al.* further demonstrate in [10] that accelerometer time-series data can be used to constrain a user's driving path to a subset of possible candidate paths within a given map. Hemminki *et al.* [11] demonstrate methods for inferring transportation modes using mobile accelerometer data, and Sankaran *et al.* demonstrates methods for inferring transportation modes using barometer data [23]. Finally, Zhou *et al.* demonstrate in [25] that app usage statistics, network address-resolution, and speaking detection can be used to infer user identity, coarse geo-location, and even whether or not a person has a certain disease. The methods presented in this paper demonstrate how pressure data collected from mobile barometers can be used to predict driving paths. Unlike methods like those presented in [9] and [10], the proposed methods allow for accurate *absolute* path predictions, benefiting from the high correlation between barometer and elevation, as detailed in [16].

## 3. ESTIMATING ELEVATION

Though barometric sensors are strong indicators of geographic elevation, they are sensitive to a host of other pressure changes as well, making the conversion from pressure to elevation non-trivial. As elevation changes, changes in air density due to Earth's gravitational pull and many other factors cause a pressure gradient dictated by the *barometric formula*. Ignoring the effects of temperature change as a function of altitude, this formula is given in [4] as

$$P = P_0 \cdot \exp\left[\frac{-gM(h-h_0)}{RT_0}\right] \quad (1)$$

where $P$ is the pressure in hecto-pascals (hPa) at height $h$ meters above reference level $h_0$, $g$ is the gravitational acceleration constant, $M$ is the molar mass of Earth's air, and $R$ is the universal gas constant for air. $P_0$ is the pressure measured at the reference height $h_0$ with temperature $T_0$, all of which can be measured beforehand. From (1), we can derive the equation $(h-h_0) = \frac{\log(P_0)RT_0}{gM} - \frac{RT_0}{gM}\log(P)$. for relatively small changes in elevation

(hundreds of meters), this can be approximated by a simple linear function in $P$. Because the reference pressure $P_0$ is in general a function of time, height $h$ can be more generally approximated as

$$h(t) \approx \alpha(t) - \beta P \quad (2)$$

for scalar $\beta$ and time-varying offset $\alpha(t)$. Over short periods of time (less than 1 hour), this prediction can be quite accurate. In fact, though the Bosch BMP280 pressure sensor used in both the iPhone 6 and Nexus 5 specifies a vertical pressure resolution of about 1 m [1], we have experimentally validated relative pressure sensitivity closer to 10–20 cm. Despite this strong correlation, determining the parameters $\alpha(t)$ and $\beta$ can be challenging, especially given the time-varying aspect of the offset $\alpha(t)$ due to, among other things, weather.

### 3.1 Elevation Model Estimation

The model parameters $\alpha(t)$ and $\beta$ in (2) dictate the accuracy of *absolute* elevation prediction. Thankfully, the scaling term $\beta$ can be considered constant over very large ranges in elevation, due to the relative flatness of earth's surface with respect to its diameter. The offset $\alpha(t)$, on the other hand, varies wildly with time and coarse location. This can be seen in Figure 2(a), where pressure collected at a static location over a 70-hour period exhibits pressure changes nearing 7 hPa or nearly 60 m estimation error. Muralidharan *et al.* describe this problem in detail in [16]. Note, however, if we look at the pressure change over 1 hour periods, the change rarely exceeds ±1 hPa (or roughly 8.3 m). Furthermore, a survey of hourly pressure data from 2,309 cities in the U.S. provided by the National Oceanic and Atmospheric Administration (NOAA) Climatic Data Center [3] shows that, the pressure change in 1-hour periods is below 1 hPa over 99% of the time. Additionally, analyzing these changes in the frequency domain indicates that the vast majority of pressure changes happen at the scale of 1 or more days, rather than hourly. This is shown in Figure 2(b). Given the slow dynamics of weather, pressure data provided by weather stations can be used to calibrate the offset term $\alpha(t)$ to within 1 hPa error—the typical resolution of pressure reported by weather stations. If such a station does not exist in close proximity to a user's coarse location, in some scenarios *relative* elevation can still be used to estimate a user's driving path with some reduction in estimation accuracy. This is discussed in more detail in Section 4.

### 3.2 Pressure Events & Noise Sources

In addition to model dynamics caused by weather, mobile barometers experience 'noise' from a number of events causing changes in air flow. This includes opening and closing doors and windows as well as changing air conditioning. Examples of this are shown in Figure 2(c). The largest magnitude pressure change is 0.5 hPa (or 4.2 m error) when air conditioning is turned completely on or off. Our path detection algorithm must be resilient to these slight perturbations.

Finally, barometer sensors themselves are not perfect and typically exhibit (small) drift over time. This error is in general non-gaussian, exhibiting temporary drifts from the true

pressure while periodically returning to accurate estimates. We propose modeling this error using an Ohnstein-Uhlenbeck diffusion process, which is similar to a low-pass-filtered white noise process [7]. This error model will be used to generate realistic barometric pressure traces for simulations, as outlined in Section 5.2.

## 4. SYSTEM OVERVIEW

Our path prediction system is composed of two main components: (1) a low power mobile app that continually monitors barometer data, periodically sending the data back to the second component: (2) a centralized analytics server that maintains road maps and elevation data and uses the collected sensory data together with the map and elevation database to estimate likely paths. This is shown in more detail in Figure 3.

Each server for a given city contains the corresponding road maps and elevation data from publicly available online databases. Specifically, the server downloads and manipulates data from (i) Open Street Map (OSM) [18], providing road topologies including segments and intersections, and (ii) the Google Elevation API [8], which provides a database from which to query the elevation of individual latitude and longitude points. Upon receiving the barometer data from a mobile device, the server's elevation conversion module converts the pressure readings to an estimate of absolute elevation values. If the sensor data is collected in a region close to a weather station, the pressure from that station can be used to calculate the offset $\alpha(t)$ in (2) and *absolute* pressure can be obtained. Otherwise, the system reduces to comparing *relative* elevation changes instead. Finally, we perform a number of pattern recognition routines including dynamic time warping (DTW) to perform path matching. These estimation routines are the main contribution of this work and the subject of the following sections.

### 4.1 Elevation Map Generation

In order to estimate the path along which a user has driven, the server must first generate a database of possible road segments and their corresponding elevations. To do this, we combine the OSM road topologies with Google's publicly available Elevation API. Road maps are downloaded from OSM in an XML format composed of 64-bit unique *node* identifiers and their corresponding latitude and longitude values. These 'nodes' are connected by elements termed *ways*—ordered lists of connected road nodes. In general, either endpoint of a *way* aligns with road intersections. Ways composed of more than 2 nodes are often used to better represent road curvatures between intersections. From the OSM road nodes and ways, we construct a graph $G = (V, E)$ where $V$ is the set of road nodes such as intersections and dead ends on the map, and each edge in $E$ represents a road segment. The elevation of each segment is then queried every 10 meters, creating new *internal* nodes belonging to set $N$ and extending the original graph to $G' = (V, E, N)$. All nodes $n \in N$ are augmented with latitude, longitude, and elevation attributes.

### 4.2 Dynamic Time Warping

Because the user is traveling at an unknown and variable speed, the corresponding pressure data behaves as a sampled version of the true elevation with variable sampling rate. In other

words, the collected pressure may be of a short duration, long duration, or it may contain pauses (when the vehicle is not in motion) or increased speeds. To compute the similarity of two signals which could potentially be scaled by time, we use dynamic time warping (DTW) algorithms like those developed for speech recognition (where the same word may be spoken with variable durations) [24]. DTW is a time-series alignment algorithm in which two signals are compared against each other by means of a cost matrix. If the two series are denoted by column vectors $\mathbf{s}^p = \{s_i^p\} \in \mathbb{R}^{N_p}$, representing the elevation corresponding to a candidate **p**ath in $G'$, and $\mathbf{s}^b = \left\{s_j^b\right\} \in \mathbb{R}^{N_b}$, corresponding to the **b**arometer-based elevation estimate, the cost matrix $C$ contains $N_p \times N_b$ elements where element $c_{i,j} = f(s_i^p, s_j^b)$. The function $f(\cdot)$ serves as a distance function to represent the difference between the two signals at given indices and is typically defined by an $\ell_2$-norm. The goal of DTW is to find the minimum-cost path through cost matrix $C$ starting at $c_{0,0}$ and ending at $c_{N_p,N_b}$ The details of the DTW algorithm are given in Algorithm 1. For each element $d_{i,j}$ in a DP matrix $D$, we store the minimum cost of all possible paths from $c_{0,0}$ to $c_{i,j}$. Each element $\psi_{i,j}$ in a traceback matrix $\Psi$ records the last transition which leads to the minimum cost of $d_{i,j}$. Thus, the final similarity between $\mathbf{s}^p$ and $\mathbf{s}^b$ is embedded in $d_{N_p,N_b}$. if $\mathbf{s}_p$ and $\mathbf{s}_b$ are similar, their corresponding $DTW$ score will be low, and if they are dissimilar their cost will be high, regardless of variable lengths or sampling rates. An example of the DTW procedure is illustrated in Figure 4(a) for example map path-and barometer-based elevation data collected from Los Angeles, CA, with the corresponding DP matrix $D$ in Figure 4(b). Following Algorithm 1, the complexity of the DTW algorithm is $\mathcal{O}(N_p \cdot N_b)$.

DTW can correctly identify paths using barometer measurements provided that the errors in the barometer sensor and Model (2) are sufficiently smaller than the variance in the traversed path elevation—i.e. when the path elevation is sufficiently distinct in the presence of noise. The result of an initial experiment performed over 29 road segments using DTW to compare measured pressure to “candidate” path elevations is shown in Figure 5. Here the true path is correctly identified by the minimum DTW score for all test cases, and the runner-up paths have anywhere from 10× to 10000× higher cost.

### 4.3 Candidate Path Generation

DTW provides the means for comparing measured pressure data against candidate path elevations: the server must search for a path $\hat{p}$ through $G'$ such that the elevation along $\hat{p}$, denoted by $\mathbf{s}^{\hat{p}}$, and and the elevation converted from barometer data, denoted by $\mathbf{s}^b$, are similar, i.e.,

$$\hat{p} = \underset{p}{\operatorname{argmin}} DTW(\mathbf{s}^p, \mathbf{s}^b)$$

Unfortunately, the search space of all candidate paths can be quite large. In fact, if we allow for path loops, the search space can be infinite—i.e., there are infinite combinations of paths in $G'$. Because of this, the server must perform DTW while traversing $G'$ in an efficient manner. In the following sections we present two potential methods for doing so: (i) a naive

breadth-first graph traversal method with pruning heuristics, and (ii) a joint optimization approach using dynamic programming.

**Algorithm 1**

Dynamic Time Warping via Dynamic Programming with traceback.

**Data**: Signals $\mathbf{s}^p \in \mathbb{R}^{N_p}$, $\mathbf{s}^b \in \mathbb{R}^{N_b}$
**Result**: similarity score $R_{score}$ and traceback vectors $\mathbf{v}^p \in \mathbb{Z}_+^{N_p}$ and $\mathbf{v}^b \in \mathbb{Z}_+^{N_b}$
Cost matrix: $C = \{c_{i,j}\} = |s_i^p - s_j^b|^2$;
DP matrix: $D = \{d_{i,j}\} = 0$;
$d_{0,j} \leftarrow \infty, \forall j = 1 \dots N_b$;
$d_{i,0} \leftarrow \infty, \forall i = 1 \dots N_p$;
Traceback matrix: $\Psi = \{\psi_{i,j}\} \leftarrow \phi$;
**for** $r$ *in* $1 \dots N_p$ **do**
  **for** $c$ *in* $1 \dots N_b$ **do**
    $d_{r,c} \leftarrow \min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c}) + c_{r,c}$;
    $\psi_{r,c} \leftarrow \arg\min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c})$;
  **end**
**end**
$R_{score} \leftarrow d_{N_p, N_b}$;
traceback: $r \leftarrow N_p$, $c \leftarrow N_b$;
**while** $r > 1$ *or* $c > 1$ **do**
  $v_r^p \leftarrow c$; $v_c^b \leftarrow r$;
  $(\Delta r, \Delta c) \leftarrow \psi_{r,c}$;
  $r \leftarrow r - \Delta r$; $c \leftarrow c - \Delta c$;
**end**

**4.3.1 Greedy Path Finding—**In order to determine which of all possible candidate paths would most likely generate an observed series of pressure data, we can use an agent-based approach in which each agent traverses $G'$ beginning at one specific node in $V$. Because $G'$ is not necessarily (and in general is not) free of cycles, a breadth-first traversal will quickly escalate into an exponential problem. To combat this, each agent ensures that no path it explores creates a loop of length less than a threshold $\Gamma_{loop}$. If $\Gamma_{loop}$ is large enough, this allows for reasonable driving trajectories while greatly limiting the search space.

At each iteration, agents perform an exploration phase and a pruning phase. Pruning occurs in three stages: (1) after all agents have finished exploring new nodes, the solver calculates all candidate path scores using a path-length-normalized version of Algorithm 1. These scores are sorted and a threshold $T_{score}$ is computed so that (2) any path whose DTW score exceeds $T_{score}$ is pruned, and finally (3) all agents are instructed to prune their worst paths until they contain no more than $\Gamma_{maxpath}$ paths. This allows for diversity in possible path starting locations and reduces the complexity of the search algorithm to polynomial time. The solver terminates if the minimum cost of all candidate paths does not change by more

than $\varepsilon$ (1%) across a recent period of $W$ (10) iterations. Upon terminating, the greedy solver returns the top ranked (lowest cost) paths and their corresponding scores. A series of snapshots from the operation of a single agent in the greedy solver is shown in Figure 6. Here we have set $\Gamma_{maxpath}$ to 4, so that at each iteration only 4 paths are being considered (labeled by green squares). For each iteration, the minimum cost path is displayed by a string of magenta triangles, while other candidate paths are displayed with a dashed blue line. At each iteration, there are at most $|V| \cdot \Gamma maxpath \cdot d_{out}^{max}$ paths, where $d_{out}^{max}$ is the maximum out-degree of any node in $V$. As a result, the time complexity can be bounded by $\mathcal{O}(|V|) \cdot \mathcal{O}(DTW) = \mathcal{O}(|V| \cdot N_p \cdot N_b)$. In practice, we truncate the map elevation segment to at most length $N_b$, giving a final complexity of $\mathcal{O}\left(|V| \cdot {N_b}^2\right)$. These calculations are performed over a number of iterations, but by setting a maximum number of iterations (30 in our experiments), the computational complexity remains unchanged.

**4.3.2 DP-based Path Finding—**The greedy search algorithm explained in the previous section is intuitive and can search efficiently over large maps. However, the pruning heuristics and search space reduction can easily lead to local minima and non-optimal path prediction. Additionally, the greedy search heuristics do not consider path timing information, reducing estimation accuracy once more. To overcome these drawbacks, we instead consider an approach inspired by Dijkstra's shortest path search algorithm, whose underlying algorithm is again solved via dynamic programming. We call this the *DP-based* path finding algorithm.

Intuitively, we would like to use a shortest path algorithm such as Dijkstra's, where the cost of each edge is calculated by DTW. However, there are two obstacles in doing this: first, the cost of each edge cannot be determined statically, as the cost of traversing a road segment depends on what segments were crossed previously. For example, road segment $A$ might be a poor match for the *start* of our collected pressure data, but it may be a very close match if the user passed through segment $B$ first on their way to $A$. Second, we must have some notion of time—e.g., for each node $v_i \in V$, what is the minimum cost of visiting that node given arrival time $t_i$? This adds an additional dimension to our dynamic cost shortest path algorithm.

To provide a notion of time, we consider a path to be not just a series of locations but also a series of corresponding timestamps. Thus, we redefine path $p$ to encompass a series of *states* $[q_0, q_1, \ldots, q_k]$ where each state $q_i$ is a 2-tuple $(v_i, t_i)$ indicating that the user has reached intersection $v_i$ at time $t_i$. Rather than define cost in terms of the DTW score between two entire paths, we can now define the marginal cost in transitioning from state $q_i$ to $q_j$. Specifically, if we have already discovered a portion of the true path, say $\mathbf{q}' = [(v_0, t_0), (v_1, t_1), \ldots, (v_i, t_i)]$ whose matching cost thus far is $\delta_{q_i}$, then the cost of transitioning to $q_j = (v_j, t_j)$ is defined as $cost(q_i, q_j)$, so that $\delta_{q_j} = \delta_{q_i} + cost(q_i, q_j)$.

Thus, for each state $q = (v, t)$, the optimal $\delta_q$ is defined as a partial path ending at vertex $v$ at time $t$ and can be determined by the following recursive function:

$$\delta_q = \min_{t_- < t, \langle v_-, v \rangle \in E} (\delta_{q_-} + DTW(\mathbf{s}^p, \mathbf{s}^b)) \quad (3)$$

$$\delta_q = (v, 0) = 0, \forall v \in V$$

where $t_-$ and $v_-$ specify the time and node corresponding to the previous state $q_-$, $\mathbf{s}^p = Elev(\langle v_-, v \rangle)$ is the elevation along the edge $\langle v_-, v \rangle$, and $\mathbf{s}^b = [s^b_{t_-}, \ldots, s^b_t]$ is the barometer-based elevation estimate from time $t_-$ to $t$. This recursive definition successfully reduces the exponential number of candidate paths to a polynomial time search algorithm: the complexity is decided by (1) the table size of $\delta$, (2) the number of state transitions, and (3) the complexity of DTW, leading to the final complexity of $\mathcal{O}(|N|^2 \cdot {N_b}^3)$. This does not scale to large maps as well as the greedy approach discussed in Section 4.3.1, but the jointly minimal solution provided by dynamic programming provides a drastically increased accuracy in path prediction.

**4.3.3 Improving DP-based Search Complexity—**The DP-based path finding algorithm suffers from high complexity due to many redundant calculations, both across time and location (i.e., vertex). For example, state $\delta_{q=(v,t)}$ is updated by any prior state $q_-$ with edge $\langle v_-, v \rangle \in E$. As described in (3), DTW is performed for each possible transition to $q$. It is possible to amortize this DTW cost by flattening out this recursive relation.

The root cause of this redundancy is that we treat each node $v \in V$ as a 'checkpoint' representing a temporary path. Traveling from $v_a$ to $v_b$ requires enumerating possible arrival times $t_b$. If, however, we consider *intermediate* nodes $n \in N$ and treat adjacent nodes as 'micro' edges, the cost of computing each edge cost by DTW is reduced since each edge length is always 1. Furthermore, we can assume that it takes at least one time unit to travel to any adjacent node if the barometer sampling rate is sufficiently low. This removes our dependency on time, and the number of possible state transitions reduces to

$$\delta_q = \delta_{q_-} + \min_{\langle n, n_- \rangle} \left| s^p_n - s^b_t \right|^2 \quad (4)$$

which is bounded by the constant $\mathcal{O}(d^{max}_{out})$, yielding a final complexity of $\mathcal{O}(|N| \cdot N_b)$. The runtime of the DP-based prediction algorithm written in MATLAB and running on an Intel i7 laptop takes roughly 7–15 minutes for a 92 km$^2$ map running on a single thread, while the greedy algorithm typically completes in 5–10 minutes.

**4.3.4 Additional Pruning Metrics—**In addition to pressure data, a number of other inertial sensors can be used to further improve path prediction accuracy and prune improbable paths to increase runtime efficiency. Most notably, mobile accelerometers, gyroscopes, and magnetomers can be combined to give accurate turn estimation as described

in [10]. These additional metrics can be easily integrated into both the greedy and DP-based path discovery algorithms at the cost of increased power consumption on the mobile devices.

**4.3.5 Elevation estimation robustness—**As described in Section 3.1, it is not always possible to achieve a high accuracy *absolute* elevation estimate. Our path search routines remain robust to errors in elevation estimation in two ways: first, the path search algorithm can be operated in *relative* elevation mode, in which only relative pressure changes are considered. Additionally, the DP-based search routine can be instructed to search over a range of possible elevation offset values, $\alpha(t)$. In doing so, the minimum score of all paths from all offset values is reported. This inevitably reduces the accuracy of the prediction algorithms, but it allows for some error margin in Eq. (2).

## 5. EVALUATION

In order to evaluate the performance of our path prediction algorithms, we collected real driving data and per formed extensive simulations over a wide range of geographical landscapes.

### 5.1 Tests on Real Driving Data

We collected real driving data across 150 km, totaling 4.6 hours of driving time and covering a range of different map topologies. Data was collected using a Nexus 5 smartphone with barometer pressure data sampled at 30 Hz and GPS sampled at 1 Hz for ground truth analysis.

The results of the two path prediction algorithms over all driving data are shown in Figure 7. For each, we plot the CDF of prediction root-mean-square errors (RMSE) for a number of ranked paths versus the average error induced by a random walk. More specifically, for each point on the predicted path we calculate the squared distance to the corresponding point on the true path, averaging the squared errors and taking the square root to give us our final RMSE value. The result from ‘5 paths’ represents the best result from the 5 lowest cost paths as estimated by the solver, the result from ‘1 path’ represents the single lowest cost path from the solver, and so forth. The random results represent the minimum error from 5 random walks of $G'$. Figure 7(a) shows that the Greedy solver demonstrates a median of around 800 m error, versus the random walk’s median error of 1600 m—only a marginal improvement over a random guess. On the other hand, the median error reduces to less than 60 m by using the dynamic programming algorithm, as shown in Figure 7(b). Additionally, in 80% of the cases the lowest cost path has an average error of just 200 m—roughly the length of a standard city block—and in 90% of cases one of the 5 lowest cost paths is correct to within 200 m.

### 5.2 Simulation

In lieu of collecting driving data across multiple cities for many hours, we conducted a series of simulated test cases. To begin, we downloaded 92 $km^2$ regions of road data and corresponding elevation data for 26 high-population cities in the U.S with, on average, 1046 km of roads per map. For each city, we conducted a series of random walks of variable

lengths and speeds and with barometer noise modeled using an Ohnstein-Uhlenbeck diffusion process as stated in Section 3.2. This random process simulates a signal with periodic deviations from a mean $\mu$ (true pressure). The frequency and magnitude of these deviations are dictated by a volatility constant $\sigma$ and reversion time $\theta$ describing how quickly disturbances return to the mean. For barometric sensors, we determined empirically that values of $\sigma = 0.04$ and $\theta = 150$ accurately simulate the errors observed in our collected data sets. These simulated barometer pressures and map/elevation databases were passed to our estimation algorithms in an identical manner to solving the real driving cases. The results of path estimations over these simulated data are summarized in Figure 8. Here, the greedy solver shows an improvement over the real driving data, but the DP-based algorithm shows a reduced estimation performance. On average over more than 500 simulated test cases, the greedy solver can predict paths to within about 200 m with 50% probability while the DP-based algorithm can predict paths to within 100 m with 50% probability and to within 300 m with around 80% probability.

### 5.3 Analysis of Parameters

The results from real and simulated experiments demonstrated in the previous section indicate that under certain circumstances, driving paths can be quite accurately predicted from pressure alone. In this section, we provide some intuition into factors that affect this prediction accuracy.

**5.3.1 Path Length—**As more barometer data is collected, the probability of distinguishing the correct path from the set of all candidate paths increases. In other words, increased path length typically (though not always) leads to increased path uniqueness. This can be seen in Figure 9 for a city with low elevation variation (a particular 9.5 $km \times 9.5\ km$ block in Chicago) and one with high variation (a 9.5 $km \times 9.5\ km$ block in Seattle). For each iteration, we generate two random paths $p_a$ and $p_b$ with the same length. Path $\tilde{p}_a$ is generated by adding modeled barometer noise over $p_a$. A *confusion error* is defined when DTW fails to distinguish the correct, noisy path $\tilde{p}_a$ from the incorrect path $p_b$. Over multiple iterations of simulation, we observe that an increase in length decreases this confusion error.

**5.3.2 Map Size—**Surprisingly, there does not seem to be a high correlation between map size and path error, as shown in Figure 10. This is most likely due to variations in the underlying map's elevation—if absolute elevation estimates can be accurately made, increasing the map size is unlikely to add potential paths whose starting points are of a similar elevation *and* who exhibit similar relative elevation signatures.

**5.3.3 Geographical Landscape—**The elevation variation of the underlying map also plays a significant role in the ability to accurately predict paths based on pressure. The variations in elevation for the 26 city maps tested in this work are shown sorted in Figure 11. Revisiting Figure 9, we see that high variation cities like Seattle have a much lower % Error than low variation cities like Chicago. This trend was also observed in general over the 26 cities studied in this paper. For example, Figure 12 shows the probability of confusing a given random path with any other path in a particular map. As the elevation variation of the underlying map increases (cross-listing again with Figure 11), there is an increased chance

for path confusion, i.e. an increased probability that any given path may exhibit non-unique elevation signatures.

## 6. DISCUSSION

We have shown through extensive tests in real driving experiments and simulated test-cases that it is often possible to predict a user's driving path with high accuracy from a time-series of barometer data. This prediction, however, is not without its limitations, as discussed below.

### 6.1 Prediction Robustness

As discussed in Section 3, the process of converting pressure to elevation depends largely on determining the pressure offset $a(t)$. When this cannot be determined by nearby weather stations, the accuracy will be greatly decreased. This can be counteracted by including additional sensor data such as turn-detection using accelerometers, gyroscopes, and magnetometers. For example, the greedy path estimation algorithm can operate on relative elevation rather than absolute, obviating the need for $a(t)$ entirely. If in addition to using relative elevation estimates we use information from mobile inertial sensors such as accelerometers and gyroscopes, the solver is still able to predict 50% of paths to within 500 m RMSE. This may be further improved by considering metrics such as driving speed estimation, driving mobility models, high traffic/highly probable routes, etc.

### 6.2 Privacy Implications

This work demonstrates real driving data in which 80% of tested paths can be predicted to within 200 m RMSE using barometer data alone. Additionally, this accuracy considers only single prediction instances—by combining data across multiple days it is likely that commonly traveled routes can be predicted with a much higher accuracy. With increasingly tight integration of social media applications in mobile devices, the potential privacy risks escalate from associating an *anonymous* user with an estimated driving path to associating a *specific*, personally identified user with a given driving path. When user anonymity, location, and behavior are compromised, the potential for breaches in security and privacy are all-the-more impressive. Additionally, mitigating privacy leaks through innocuous sensors like barometers may not be as simple as implementing stricter access controls— balancing usability with utility is a non-trivial task, both technically and philosophically.

### 6.3 Future Work

This work demonstrates methods for accurately predicting driving paths based on barometric pressure data, resulting in a very low power method for large scale traffic analysis in emerging smart cities. The high correlation between pressure and elevation and the inclusion of these sensors on modern mobile devices raises a number of additional research questions. For example, can barometer pressure be leveraged to improve location-services in real-time to aid in spotty GPS coverage or to further reduce power consumption of location services? In addition, can similar methods to those discussed in this work be used to predict pedestrian paths in an unconstrained environment, such as for hikers? Finally, leveraging results describing pressure changes as a function of vertical motion indoors (i.e. elevators,

escalators, and stairs) [16], is it possible to infer which building or subset of buildings a user may be walking through based on unique patterns of floor changes, enhancing path estimation and occupancy detection algorithms? In future work, we plan to explore these questions in an attempt to further evaluate the benefits of a city-wide, distributed network of pressure sensors.

## 7. CONCLUSION

We have demonstrated methods by which barometric pressure data collected on mobile phones can be used to infer driving paths with surprisingly high accuracy. Specifically, we described both a greedy graph traversal approach and a dynamic-programming approach to estimating likely driving paths given pressure-based elevation estimates and a map of potential road segments. These methods leverage results from dynamic time warping literature to calculate a rate-independent similarity score between estimated and candidate path elevation signatures. Pressure data collected over a total of 4.6 hours and 150 km demonstrates that these algorithms can predict upwards of 90% of paths with less than 100 m error. Additionally, we illustrated the accuracy of these prediction methods for more than 500 simulated test cases across 26 cities. The results of these simulations show that across all cities more than 70% of paths can be predicted to within an error of 200 m. We further evaluated the ability to estimate a user's driving path as a function of several variables, including length of barometer data, map size, and the elevation variance of the underlying map.

The results of the methods described in this paper serve to emphasize the importance of distributed networks of smart devices in emerging smart cities as well as the growing problem of personal data privacy, lending credence to research efforts focused on treating data in a privacy-preserving and security-aware manner. Finally, all sensor data and software described in this paper is open source and available at https://github.com/nesl/mercury.

## Acknowledgments

This research is funded in part by the National Science Foundation under awards CNS-1136174 and CNS-1213140 and by the Center for Excellence for Mobile Sensor Data-to-Knowledge under National Institutes of Health grant #1U54-EB020404. The first author also acknowledges support from a Taiwan Technologies Incubation Scholarship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above-mentioned funding agencies.

## References

1. Bmp280 digital pressure sensor. https://ae-bst.resource.bosch.com/media/products/dokumente/bmp280/BST-BMP280-DS001-10.pdf. Accessed: 2015-03-04
2. Globalsat em-506 gps module. http://www.globalsat.com.tw/. Accessed: 2015-09-12
3. National oceanic and atmospheric administration (noaa) national climatic data center. http://www.ncdc.noaa.gov. Accessed: 2015-03-05
4. US Standard Atmopshere. U.S Government Printing Office; Washington, D.C.: 1976.
5. Aram S, Troiano A, Pasero E. Environment sensing using smartphone. Sensors Applications Symposium (SAS), 2012 IEEE. Feb.2012 :1–4.
6. Diggelen, V., Tromp, FS. A-gps: Assisted gps, gnss, and sbas. Boston: Artech House; 2009.

7. Enrico Bibbona PT, Panfilo Gianna. The ornstein-uhlenbeck process as a model of a low pass filtered white noise. Metrologia, Metrologia. 2008; 45
8. Google. Google elevation api. 2015. https://developers.google.com/maps/documentation/elevation/ [Online; accessed 7-March-2015]
9. Guha, S., Plarre, K., Lissner, D., Mitra, S., Krishna, B., Dutta, P., Kumar, S. Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10. New York, NY, USA: ACM; 2010. Autowitness: Locating and tracking stolen property while tolerating gps and radio outages; p. 29-42.
10. Han J, Owusu E, Nguyen L, Perrig A, Zhang J. Accomplice: Location inference using accelerometers on smartphones. Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on. Jan.2012 :1–9.
11. Hemminki, S., Nurmi, P., Tarkoma, S. Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys '13. New York, NY, USA: ACM; 2013. Accelerometer-based transportation mode detection on smartphones; p. 13:1-13:14.
12. Liu, J., Priyantha, B., Hart, T., Ramos, HS., Loureiro, AAF., Wang, Q. Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys '12. New York, NY, USA: ACM; 2012. Energy efficient gps sensing with cloud offloading; p. 85-98.
13. Michalevsky, Y., Boneh, D., Nakibly, G. 23rd USENIX Security Symposium (USENIX Security 14). San Diego, CA: USENIX Association; Aug. 2014 Gyrophone: Recognizing speech from gyroscope signals; p. 1053-1067.
14. Miluzzo, E., Varshavsky, A., Balakrishnan, S., Choudhury, RR. Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12. New York, NY, USA: ACM; 2012. Tapprints: Your finger taps have fingerprints; p. 323-336.
15. Mohan, P., Padmanabhan, VN., Ramjee, R. Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys '08. New York, NY, USA: ACM; 2008. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones; p. 323-336.
16. Muralidharan, K., Khan, AJ., Misra, A., Balan, RK., Agarwal, S. Proceedings of the 15th Workshop on Mobile Computing Systems and Applications, HotMobile '14. New York, NY, USA: ACM; 2014. Barometric phone sensors: More hype than hope!; p. 12:1-12:6.
17. Musa, ABM., Eriksson, J. Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys '12. New York, NY, USA: ACM; 2012. Tracking unmodified smartphones using wi-fi monitors; p. 281-294.
18. OpenStreetMap. Main page — openstreetmap wiki. 2015. http://wiki.openstreetmap.org[Online; accessed 7-March-2015]
19. Ouyang, RW., Srivastava, M., Toniolo, A., Norman, TJ. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14. New York, NY, USA: ACM; 2014. Truth discovery in crowdsourced detection of spatial events; p. 461-470.
20. Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J. Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications, HotMobile '12. New York, NY, USA: ACM; 2012. Accessory: Password inference using accelerometers on smartphones; p. 9:1-9:6.
21. Park, J-g, Patel, A., Curtis, D., Teller, S., Ledlie, J. Online pose classification and walking speed estimation using handheld devices. Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UbiComp '12. 2012:113–122.
22. Raij A, Ghosh A, Kumar S, Srivastava M. Privacy risks emerging from the adoption of innocuous wearable sensors in the mobile environment. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11. 2011:11–20.
23. Sankaran, K., Zhu, M., Guo, XF., Ananda, AL., Chan, MC., Peh, LS. Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, SenSys '14. New York, NY, USA: ACM; 2014. Using mobile phone barometer for low-power transportation context detection; p. 191-205.
24. Vintsyuk T. Speech discrimination by dynamic programming. Cybernetics. 1968; 4(1):52–57.
25. Zhou, X., Demetriou, S., He, D., Naveed, M., Pan, X., Wang, X., Gunter, CA., Nahrstedt, K. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security,

CCS '13. New York, NY, USA: ACM; 2013. Identity, location, disease and more: Inferring your secrets from android public resources; p. 1017-1028.

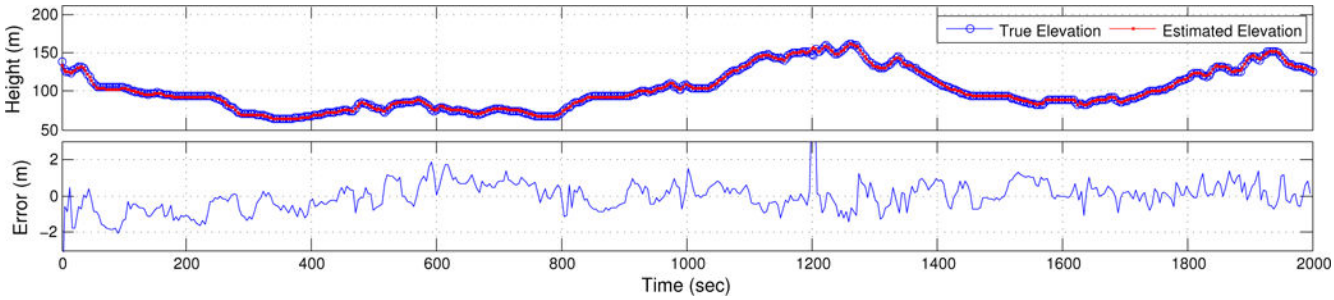
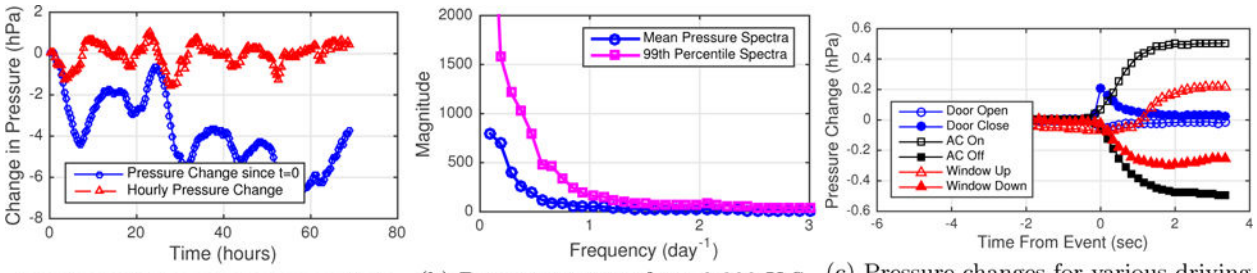



**Figure 1.**
Elevation estimation from pressure with corresponding error, using simple linear model.




(a) Pressure change across multiple days.

(b) Pressure spectra from 2,309 U.S. cities provided by NOAA.

(c) Pressure changes for various driving events.

**Figure 2.**

Noise and variations in barometric pressure measurements.

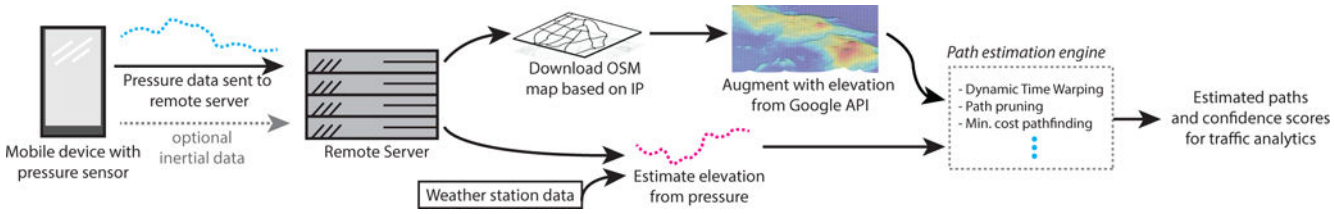
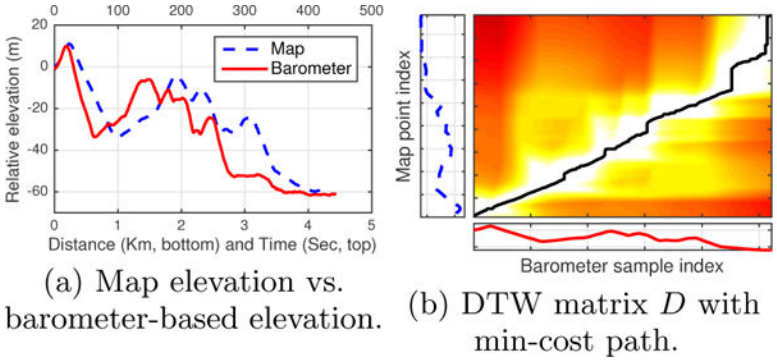



**Figure 3.**
System overview, from pressure data collection to path estimation and confidence score.




(a) Map elevation vs. barometer-based elevation.

(b) DTW matrix $D$ with min-cost path.

**Figure 4.**
An illustration of path elevation matching using dynamic time warping.

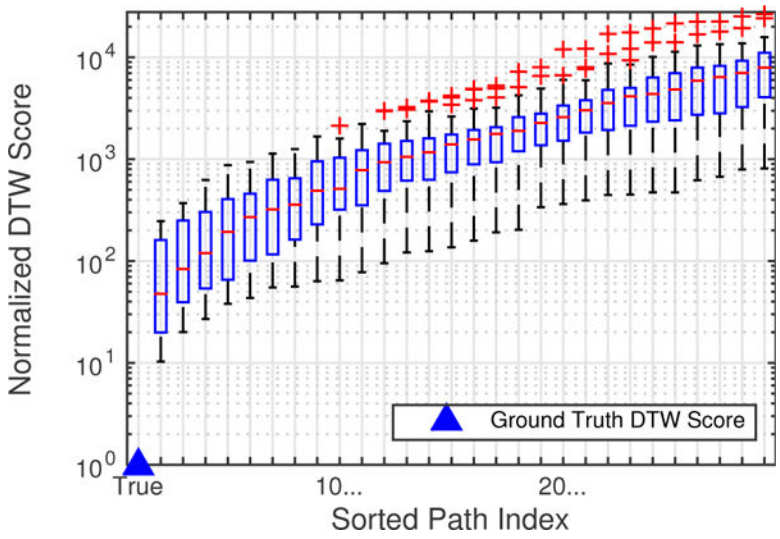
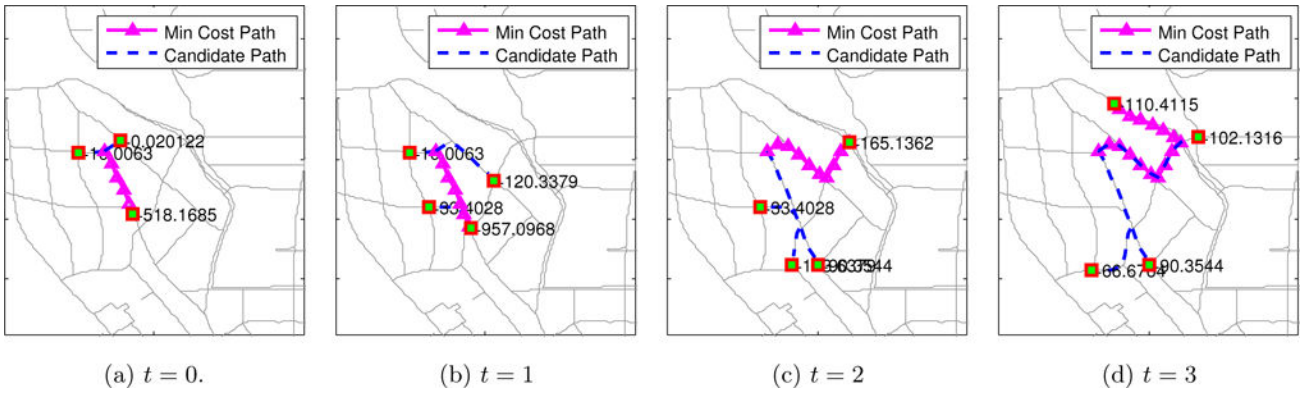



**Figure 5.**

Normalized DTW scores for 29 barometer traces collected while driving. False paths have between 10 and 10000× higher DTW scores.




(a) $t = 0$. (b) $t = 1$ (c) $t = 2$ (d) $t = 3$

**Figure 6.**
Snapshots of an agent from greedy pathfinding, exploring 4 possible paths.

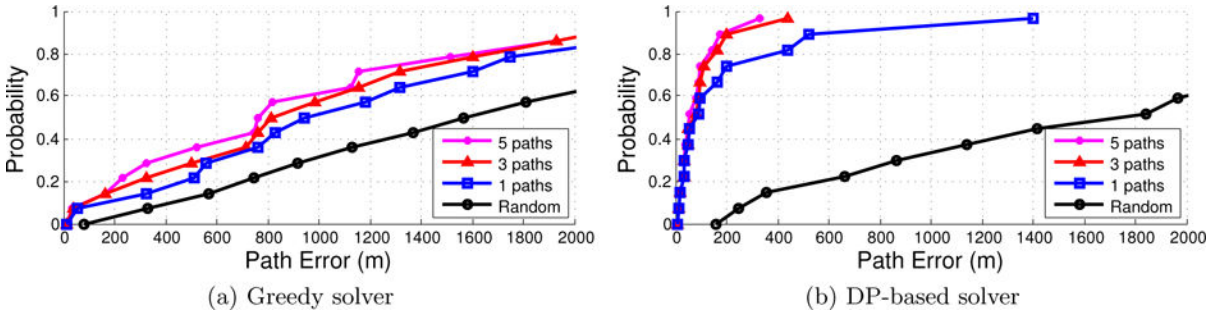



**Figure 7.**
Path prediction errors for real driving data

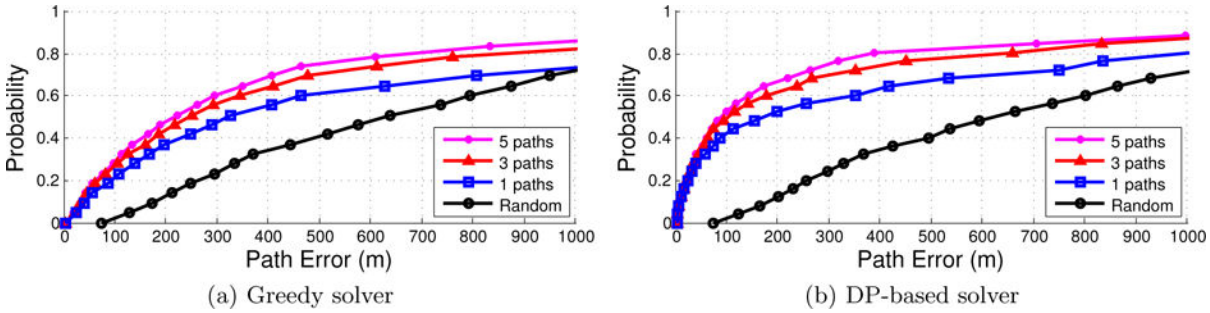



**Figure 8.**
Path prediction errors for simulated driving data

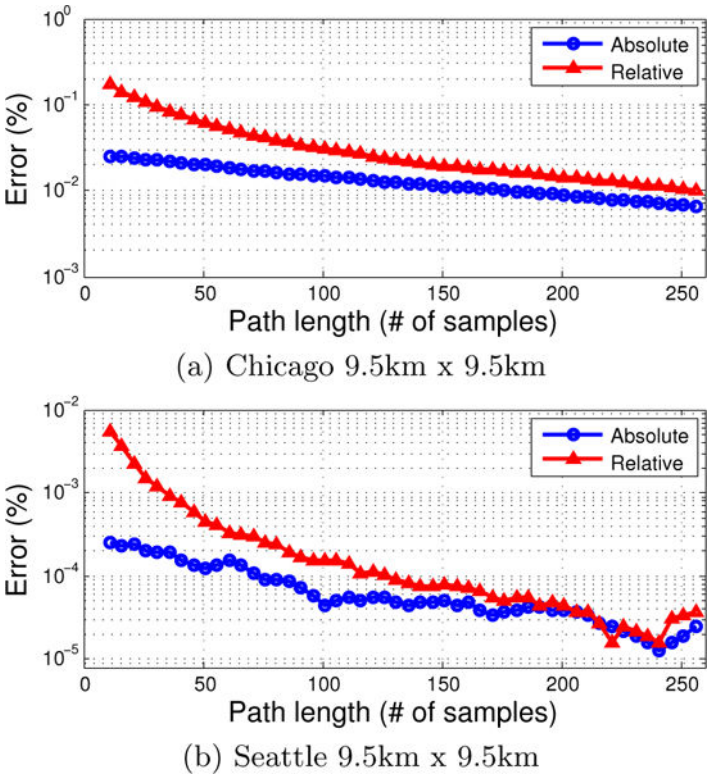



**Figure 9.**
Path length v.s. DTW Confusion Error

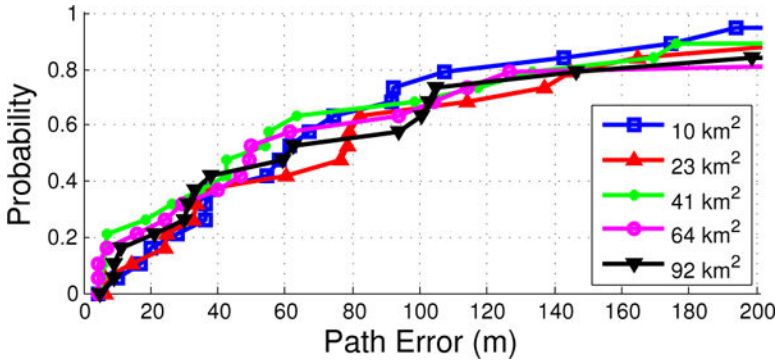



**Figure 10.**
Path prediction errors vs. map size.

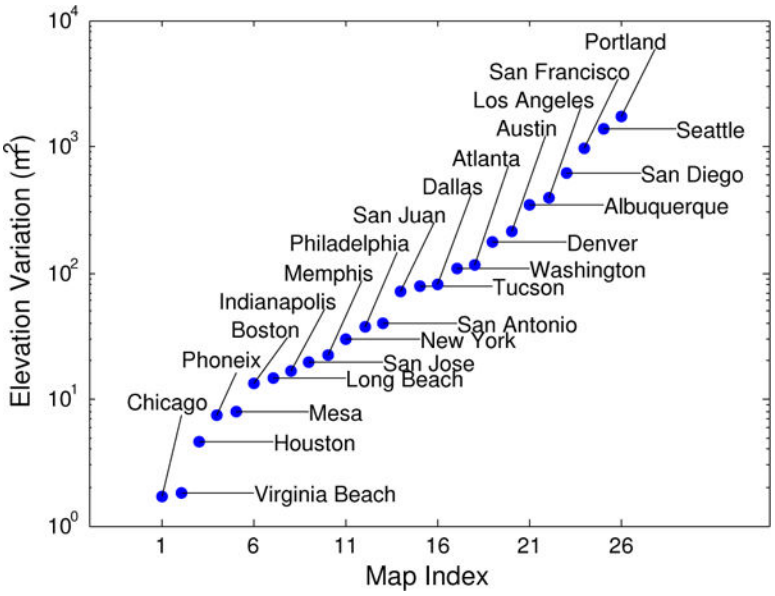



**Figure 11.**
Elevation variations for sampled city maps.

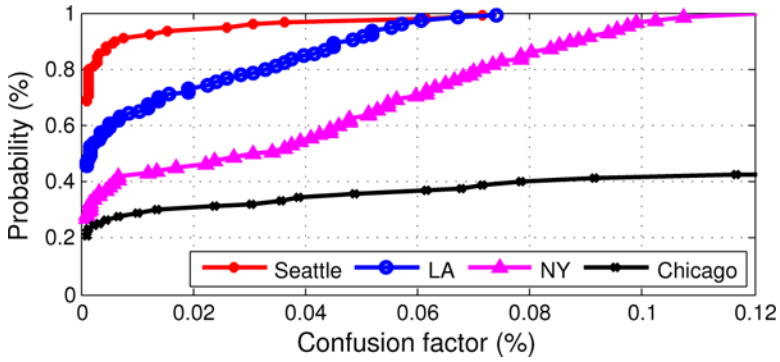



**Figure 12.**
CDF of path confusion factors for cities of varying elevation variation.