ACComplice: Location Inference using Accelerometers on Smartphones

Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, Joy Zhang
{junhan, eowusu, lenguyen, perrig, sky}@cmu.edu
Carnegie Mellon University

Abstract—**The security and privacy risks posed by smartphone sensors such as microphones and cameras have been well documented. However, the importance of accelerometers have been largely ignored. We show that accelerometer readings can be used to infer the trajectory and starting point of an individual who is driving. This raises concerns for two main reasons. First, unauthorized access to an individual's location is a serious invasion of privacy and security. Second, current smartphone operating systems allow any application to observe accelerometer readings without requiring special privileges. We demonstrate that accelerometers can be used to locate a device owner to within a 200 meter radius of the true location. Our results are comparable to the typical accuracy for handheld global positioning systems.**

I. Introduction

Location privacy has been a hot topic in recent news after it was reported that Apple, Google, and Microsoft collect records of the location of customers using their mobile operating systems [12]. In some cases, consumers are seeking compensation in civil suits against the companies [8]. Xu and Teo find that, in general, mobile phone users express lower levels of concern about privacy if they control access to their personal information. Additionally, users expect their smartphones to provide such a level of control [20].

There are situations in which people may want to broadcast their location. In fact, many social networking applications incorporate location-sharing services, such as geo-tagging photos and status updates, or checking in to a location with friends. However, in these instances, users can control when their location is shared and with whom. Furthermore, users express a need for an even richer set of location-privacy settings than those offered by current location-sharing applications [2]. User concerns over location-privacy are warranted. Websites like "Please Rob Me" underscore the potential dangers of exposing one's location to malicious parties [5]. The study presented here demonstrates a clear violation of user control over sensitive private information.

Accelerometers are a particularly interesting case because of their pervasiveness in a large assortment of personal electronic devices including tablet PCs, MP3 players, and handheld gaming devices. This array of devices provides a large network for spyware to exploit.

Furthermore, by correlating the accelerometer readings between multiple phones it is possible for an adversary to determine whether the phones are in close proximity. Because phones undergoing similar motions can be identified by their accelerations, events such as earthquakes or even everyday activities like public transportation (e.g., bus, train, subway) produce identifiable motion signatures that can be correlated with other users. As a consequence, if one person grants GPS access, or exposes their cellular or Wi-Fi base station, then they essentially expose the location of all nearby phones, assuming the adversary has access to these devices.

a) Contributions: Our key insight is that accelerometers enable the identification of one's location despite a highly noisy trajectory output. This is because the idiosyncrasies of roadways create globally unique constraints. Dead reckoning can be used to track a user's location long after location services have been disabled [6]. But as we show, the accelerometer can be used to infer a location with no initial location information. This is a very powerful side-channel that can be exploited even if location-based services on the device are disabled.

b) Threat Model: We assume that the adversary can execute applications on the mobile device, without any special privileges except the capability to send information over the network. The application will use some legitimate reason to obtain access to network communication. This is easily accomplished by mimicking a popular application that many users download; e.g., a video game. In the case of a game, network access would be needed to upload high scores or to download advertisements. We assume that the OS is not compromised, so that the malicious application simply executes as a standard application. The application can communicate with an external server to leak acceleration information. Based on the leaked information, the adversary can extract a mobile user's trajectory from the compromised device via data analysis.

Our goal is to determine the location of an individual driving in a vehicle based solely on motion sensor measurements. The general approach that we take is to first derive an approximate motion trajectory given acceleration measurements–which we discuss in §II. We then correlate that trajectory with map

This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273, from the Army Research Office, and by support from NSF under TRUST STC CCF-0424422, IGERT DGE-0903659, and CNS-1050224, and by a Google research award. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, Google, NSF or the U.S. Government or any of its agencies.

978-1-4673-0298-2/12/$31.00 © 2012 IEEE

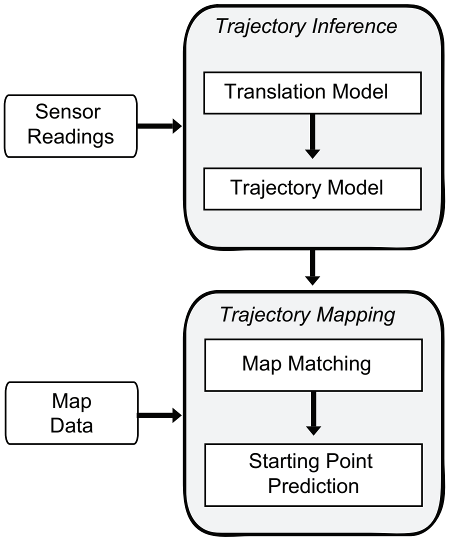


Fig. 1. System architecture of *ACComplice*

information to infer the location–which we discuss in §III. Figure 1 depicts the overall system architecture. In §IV, we present our experimental results. We provide further discussions in §V, a discussion of related works in §VI, and conclude in §VII.

II. Background: Trajectory Inference

The section provides a background description of the trajectory inference model. First, we discuss a statistical method for translating sensor samples to displacements. Next, we present a statistical model for inferring trajectories.

A. Displacement Estimation

We show that a sequence of motion sensor readings can be used to generate an estimation of the motion trajectory of the handset. We use a probabilistic dead reckoning method called Probabilistic Inertial Navigation (ProbIN). In our prior work, we developed ProbIN to estimate a user's indoor location using inertial and magnetic sensors when GPS signals are not available [14], [15].

Classical dead reckoning has inherent limitations when estimating the motion trajectory from motion sensor readings. The main problem is *drifting error*. Drifting occurs because the position of the vehicle at time t depends on the position at $t-1$, and the displacement estimated at time interval $[t-1, t]$. Drifting error aggregates over time, pushing the estimated trajectory further away from the truth. Even after applying noise smoothing methods on the sensor readings, such as the Kalman Filter [17], the estimated trajectory may be significantly different than the ground truth as reported in many previous works [4], [18], [14].

The ProbIN approach is different from the standard dead reckoning approach in two ways:

1) Instead of using the actual values of the sensor measurements for displacement estimation, we treat the measurements only as "observations" of the underlying motion. By using the ProbIN statistical model, these observations are directly *mapped* to the displacement of the vehicle (Figure 2).
2) The standard dead reckoning approaches calculate the displacement deterministically from the sensor readings. ProbIN frames displacement estimation as a probabilistic process, where a sensor reading can correspond to many displacements with different probabilities. ProbIN uses a Bayesian framework to combine the translation model and the trajectory model in order to search for the optimal trajectory.

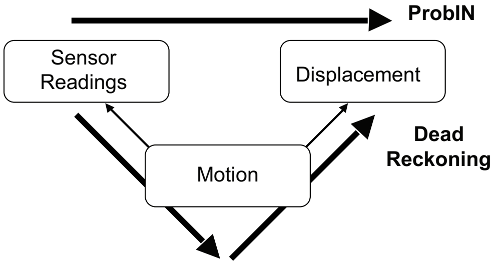


Fig. 2. The standard dead reckoning approach uses a physics-based model to translate the measured values of the underlying motion to displacements. ProbIN uses a statistical model to directly map the sensor readings to displacements.

B. Quantizing Sensor Readings

To simplify the model and to make the computation efficient, we first convert the continuous space to the discrete space by quantizing the sensor readings and the displacement space.

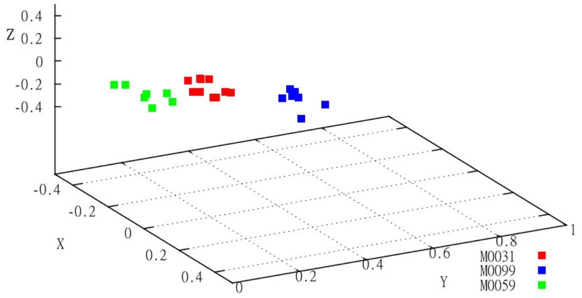


Fig. 3. An example of clustering sensor readings into motion labels. The acceleration vector measurements (in units of g = 9.81 m/s^2) are grouped into clusters $M0031$, $M0099$ and $M0059$ using k-means clustering.

$$[0.08, 0.40, 0.08][0.09, 0.49, 0.06][0.04, 0.16, 0.08]\cdots$$

$$\Downarrow$$

$$M0099, M0099, M0031, \cdots$$

Fig. 4. An example of representing a sequence of sensor readings (in units of g = 9.81 m/s^2) with a sequence of motion labels.

We cluster acceleration vectors in the training data using the standard k-means clustering algorithm which result in k clusters (as shown in Figure 3). During the training and testing phase, an acceleration vector is first labeled by its nearest cluster. This motion label is later used for representing the sensor reading (as shown in Figure 4).

This quantization step results in loss of precision of the input data as we reduce the acceleration readings from a continuous high dimension space to a one-dimensional discrete space. The reduced feature space has relatively small vocabulary size (e.g., k=200, as used in one of our experiments). Empirical results show, however, that such quantization does not affect the end-to-end system performance. Similarly, we quantize all possible displacements seen in the training data, for the sampling time interval, Δt. In the subsequent discussion, we refer to the quantized acceleration values and the displacement values as motion labels and displacement labels, respectively.

C. ProbIN Model

ProbIN builds a statistical model to map the sensor readings directly to the displacement. From the training data, ProbIN learns the probability of translating a displacement label D to its corresponding motion label M with probability $P(M|D)$. Again, there are alternative Ms to translate to one D because of the inherent noise in sensor data. The conditional translation probabilities for a particular value of $D = d$ have to satisfy:

$$\sum_{M} P(M|D=d) = 1 \tag{1}$$

In addition to the translation model, ProbIN also models the trajectory pattern. The underlying assumption is that there are regularities in a vehicle's trajectory and we can calculate the probability of a trajectory given the trained trajectory model. Intuitively, it is more likely to have a trajectory where the car drives forward and then makes a left turn than a trajectory with a sequence of alternating turns (e.g., left, right, left, then right). From the collected training data, we can build a trajectory model and estimate the likelihood of a trajectory $\tilde{D}$ which is a sequence of displacement labels D.

Combined, the goal of ProbIN is to find an optimal displacements vector $\tilde{D}^*$ such that:

$$\tilde{D}^* = \underset{\tilde{D}}{\operatorname{argmax}} P(\tilde{M}|\tilde{D})P(\tilde{D}) \tag{2}$$

The translation model is used to estimate $P(\tilde{M}|\tilde{D})$. Assuming that the translating conditional probability is independent and identically distributed (i.i.d.), we can break down the conditional probability of the sequence $P(\tilde{M} = M_1, M_2, \ldots, M_T|\tilde{D} = D_1, D_2, \ldots, D_T)$ to:

$$P(\tilde{M}|\tilde{D}) = \prod_{t=1}^{T} P(M_t|D_t). \tag{3}$$

By assuming that the trajectories satisfies the Markov assumption that the displacement at time t only depends on $n-1$ previous displacement segments in the past, we can break down the trajectory probabilities $P(\tilde{D})$ down to:

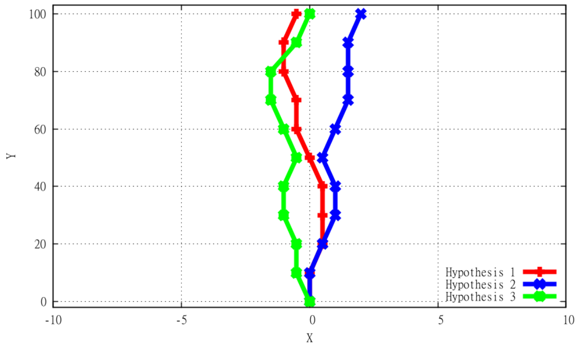


Fig. 5. An example of trajectory hypotheses with the highest probability generated by the decoder (in meters)

$$P(D = D_1, \tilde{D}_2, \ldots, D_T) = \prod_{t=1}^{T} P(D_t|D_{t-n+1}, \ldots, D_{t-1}). \tag{4}$$

D. Training ProbIN

Both statistical translation and trajectory models need to be trained in order to estimate $P(M|D)$ and $P(D_t|D_{t-n+1}, \ldots, D_{t-1})$. If the training data is labeled (i.e., we know the corresponding displacement for each sensor reading), it is straightforward to calculate $P(M|D)$ using the Maximum Likelihood Estimation (MLE) and $P(D_t|D_{t-n+1}, \ldots, D_{t-1})$ using MLE with proper smoothing mechanism such as the modified Kneser-Ney method [3]. Labeled data requires recording the displacement corresponding to each sensor sample. In our case samples occur every 0.02 to 0.1 seconds depending on the sampling frequency, rendering this approach is impractical. To solve this problem, ProbIN uses an approximated EM algorithm to train both the translation and trajectory models.The main idea of this solution is to treat the displacement at each time as “hidden” information. Through the iterative EM process, we approximate the unknown displacement and update the model accordingly.

The training data contains multiple trips of a car driving normally with the smartphone constantly collecting motion sensor data. For each trip, we record the actual starting and ending positions through the GPS receiver. The training process goes through the following steps:

1) Initialize the model by assuming the sensor readings are correct and apply a physics-based approach to estimate D for each M.
2) E-step: based on the current translation model and the trajectory model, estimate the trajectory of each trip with the constraint that the starting and ending positions of the estimated trajectory match the true starting/ending positions.
3) M-step: from estimated trajectories for all trips, update the translation model and the trajectory model. Go to E-step. Repeat till model converges.

E. Decoding Trajectory

Once the statistical models are trained, we can apply the model on the input sensor reading sequence and decode it with an optimal trajectory (as shown in Figure 5).

In the probabilistic navigation framework, each motion label can be translated into multiple displacement labels with different probabilities. For an input sensor reading sequence of T labels, if the average number of displacement labels for a motion label is m then the total number of possible trajectories is m^T. The goal of the decoder is to search through all possible trajectories and find the one with the highest probability $P(\tilde{D}|\tilde{M})$.

It is infeasible to calculate $P(\tilde{D}|\tilde{M})$ for each of the m^T trajectories and select the best one as the output. Instead, we use a stack decoder to approximate the optimal trajectory and keep the computing time complexity linear to T. After applying the translation model on the input motion label sequence and building a lattice, the stack decoder scans the lattice from left $t = 1$ to the end of the sequence $t = T$. At each time t, the decoder extends the partial hypotheses from $t-1$ with all possible displacements translated from M_t and calculate corresponding translation and trajectory probabilities for the augmented partial hypotheses. As the n-gram trajectory model can only differentiate trajectory sequences that have different $n-1$ ending displacement labels, the stack decoder only needs to keep the best partial hypothesis for each trajectory model endings. Also, we prune out partial hypotheses at each time t if their probabilities are much worse than the best alternative at this stage.

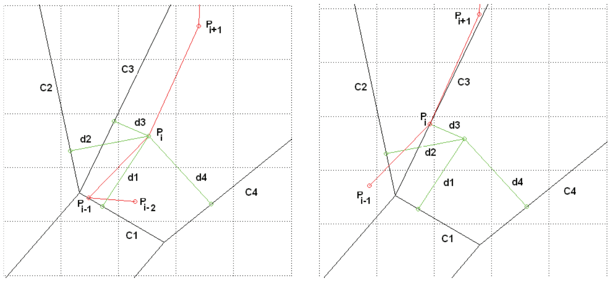
III. Trajectory Mapping

We now present the process for associating the motion trajectories obtained from ProbIN to the most likely location on a map. We thus demonstrate that an adversary can deduce the trajectory and starting point from acceleration information.

The mapping of a motion trajectory to a path on a map is the first challenge we need to address. This problem is complicated by the noise in the motion trajectory. In §III-A we describe our approach to map matching. The map matching algorithm outputs a score of how closely a motion trajectory matches a given path on the map.

Using the similarity score as our guide we search the map for the best fit. We follow a brute-force approach for finding the starting position by computing a similarity for all starting locations. Our predicted map location will be the one containing a driving route most similar to our trajectory.

While this approach is computationally expensive, we make use of several constraints that reduce the number of candidate paths. We make use of the fact that trajectories can be uniquely identified. For example, the inherent idiosyncrasies of roads and the fact that every turn adds an additional constraint reduces the number of potential trajectory candidates. There are more optimizations that can be incorporated in a sophisticated attack. We discuss areas for optimization in §V.



(a) (b)

Fig. 6. An example step of the map matching algorithm. (a) and (b) are the before and after steps as P_i is mapped to a candidate road segment.

A. Map Matching Algorithm

Our map matching algorithm maps motion trajectory points, $MT=[P_1, P_2,..., P_n]$, to corresponding points on a road network. This matching is accomplished by mapping each motion trajectory point to the best corresponding candidate segment. Our algorithm follows Greenfield's approach for map matching [7], by utilizing similarity measures to assign scores of candidate road segments. We use distance and angle as similarity measures to map motion trajectory onto road segments.

Consider the problem of matching point P_i to a road segment as illustrated by Figure 6. Assuming that P_{i-1} is mapped to a point on road segment C_1, potential road segment candidates for P_i are compared. The candidate edges are defined as a road segment onto which the previous trajectory point is mapped, and its adjacent segments. In this case, in addition to C_1, segments C_2, C_3, and C_4 are all candidate segments. In order to map P_i to a candidate road segment, we use two similarity measures S_d and S_α. S_d is calculated using the orthogonal projection, d, of P_i to a candidate road segment. P_i has four projections to its possible candidate segments, d_1, d_2, d_3, and d_4 corresponding to each of its candidate segments. Specifically S_d can be computed with the following equation:

$$S_d = a * d^{-1} \qquad (5)$$

where a is a weighting constant empirically determined to be 100. S_α is calculated using the angle α between a motion trajectory segment and a candidate segment by computing the following:

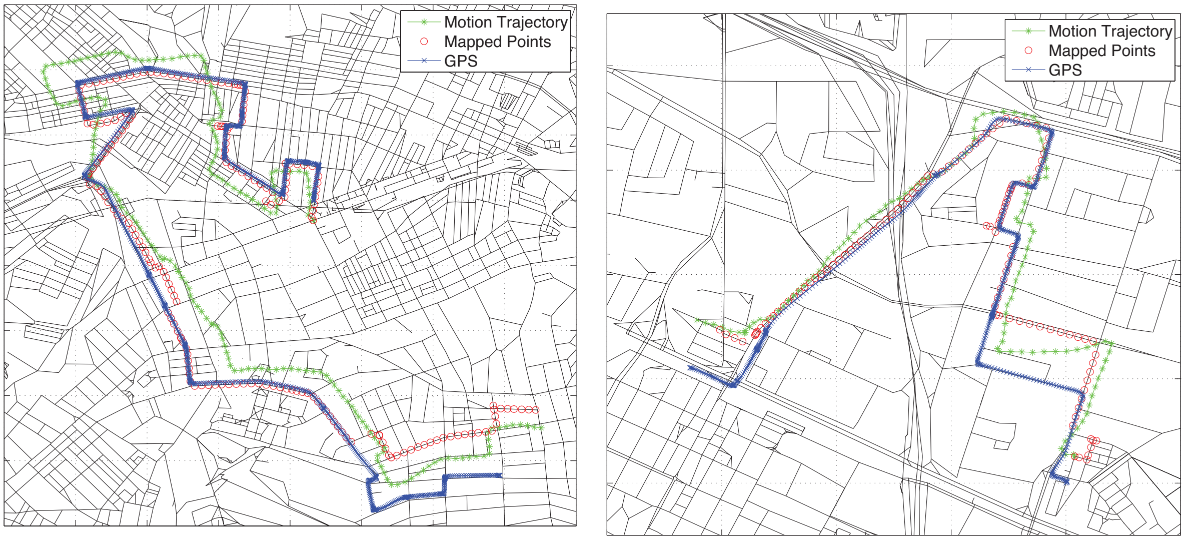
$$S_\alpha = C * cos(\alpha)^N \qquad (6)$$

where C and N are the weighting and rate constants, respectively. The rate constant N is used to vary the rate of decrease of S_α with respect to α. The constants are determined empirically: $C = 25$ and $N = 4$. The final score is computed by the following:

$$S_{Total} = S_d + S_\alpha \qquad (7)$$

where a greater S_{Total} denotes a better score for the corresponding candidate.

Based on this scoring mechanism, P_i is mapped to a point on C_3. The remaining points of the motion trajectory are realigned



(a) Pittsburgh, PA

(b) Mountain View, CA

Fig. 7. Verification of map matching algorithm with the known starting point. (a) and (b) show Experiment 2a (Pittsburgh, PA) and Experiment 2b (Mountain View, CA), respectively. The green (star) curve indicates the motion trajectory obtained from ProbIN. The red (circle) curve indicates the mapped points. The blue (x-mark) curve indicates the ground truth (i.e., actual route traveled) obtained from GPS data.

to the mapped point on C_3, which is depicted in Figure 6(b). This realignment mitigates the effect of the noise inherent in the motion trajectory.

To verify the results, we perform the aforementioned map matching algorithm on a map with a known starting point. We pre-processed motion paths to remove highly transient noise in the motion trajectory, which smooths the data while preserving key features of the trajectory. Figure 7 illustrates the map matching process of a motion trajectory.

Figures 7(a) and (b) show matching procedures for Experiment 2a and Experiment 2b, respectively. The green (star) curve shows the motion trajectory obtained from ProbIN. The red (circle) curve illustrates the corresponding mapped points on a map. The blue (x-mark) curve shows the ground truth using GPS data to validate our algorithm.

B. Starting Point Prediction

This section explains an algorithm that can be used to predict a starting point of a trajectory by applying the map matching algorithm to all points on a given map. First the trajectory is aligned to every point on the map (each node represents a possible starting point). In order to determine the most likely starting point, we compare each mapped route using a trajectory difference metric (explained below).

We take the first point of the trajectory, P_0, and plot all candidate starting pionts. Let M_i be a point on a road network that has been matched to P_i, a point on the motion trajectory. We define $Dist(M_i,P_i)$ as the distance between M_i and P_i. Summing up $Dist(M_i,P_i)$ from 1 to s, where s is the number of mapped points, we obtain the difference score DS, given by the following equation:

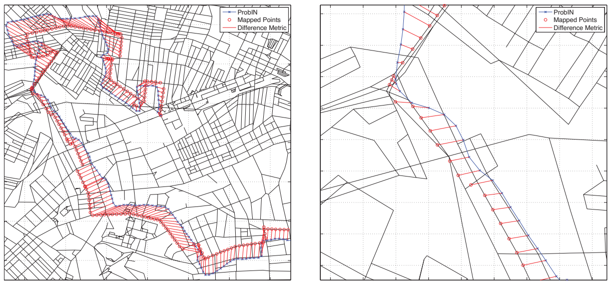
$$DS = \sum_{i=1}^{s} Dist(M_i,P_i) \tag{8}$$

Figure 8 shows $Dist(M_i,P_i)$ for all corresponding points between the motion trajectory and the mapped points. The difference score DS in Figure 8(a) is the sum of lengths of $Dist(M_i,P_i)$ for all i. Once DS is computed for each starting point on the map, the starting points are ranked in ascending order because a small DS is a good indicator that the actual and predicted paths are similar.

In the case of erroneous starting points, the map matching algorithm can encounter a dead end or may not be able to match any road on the map to the motion trajectory, in which case the map matching algorithm simply maps several motion trajectory points to a single map location. Many starting points will not result in any valid candidate paths, especially for longer paths – we detect these cases and reject those paths.

We then take the top ten ranked points sorted in ascending order of DS to locate the actual starting point. Among the highly ranked points, the points are likely to form a few clusters of nodes that have closely matched routes with the aligned motion trajectory. If only a few clusters on a map are found, it is an indication with strong certainty that the actual starting point is in one of the clusters.

However, the motion trajectory is a noisy estimate of an actual traveled route. Because the map matching algorithm is topologically sensitive, some trajectories may not be accurately scaled. In order to account for this fact, we stretch the motion trajectory and repeat the prediction algorithm. With varied trajectory scales, the probability of finding a valid match between motion trajectory and the actual traveled route greatly increases. The following section provides evaluation data.



(a) (b)

Fig. 8. Difference Metric using $Dist(M_i, P_i)$. (a) shows the entire motion trajectory and the corresponding mapped points on the map. (b) shows a close-up of (a)

IV. Evaluation

This section discusses the experiments used to evaluate the methodologies mentioned in §II and §III. First, we show how ProbIN is achieved from data sample collected while driving. Second, we demonstrate map matching and starting point identification with collected data from different driving routes.

A. Experiment 1: ProbIN

1) Configuration: In our research, we focus on the mobile smartphones in common use. Each device is usually equipped with a 3-axis accelerometer and magnetometer providing information about the acceleration in the device's coordinate system and the azimuth angle identifying the direction showing to the north.

Our approach is evaluated on the iPhone 4, which is additionally equipped with a 3-axis gyroscope. By utilizing the rotation rate that is provided by the gyroscope a more accurate orientation can be calculated. However, since the most smartphones do not have a gyroscope, in our experiments we will derive the vehicle's position based solely on the accelerometer and magnetometer reading. We use the magnetometer to improve the accuracy of the trajectory inference phase. However, we are able to obtain a trajectory with only accelerometer measurements.

The orientation pitch and roll angle can be calculated from the accelerometer reading when the phone is not moving. We assume that at the beginning of each drive the phone is lying still in the car and that the car is standing still on a level street. Thus, we will be able to estimate the rough initial orientation of the phone. During the drive, the acceleration values are not a reliable source for recalculating the pitch and roll angle. Even if the car would be temporarily standing (e.g., at a stop sign) the vibration caused by the running car engine adds additional noise to the accelerometer reading. Therefore, the initial pitch and roll angles will be used for transforming the acceleration into the world's coordinate system during the entire drive.

In the training phase, the raw sensor reading is transformed into the acceleration in world's coordinate system, which is then double integrated in order to calculate the displacement. By summing up the displacements the trajectory of the car can be estimated. This trajectory is used for bootstrapping the ProbIN statistical model.

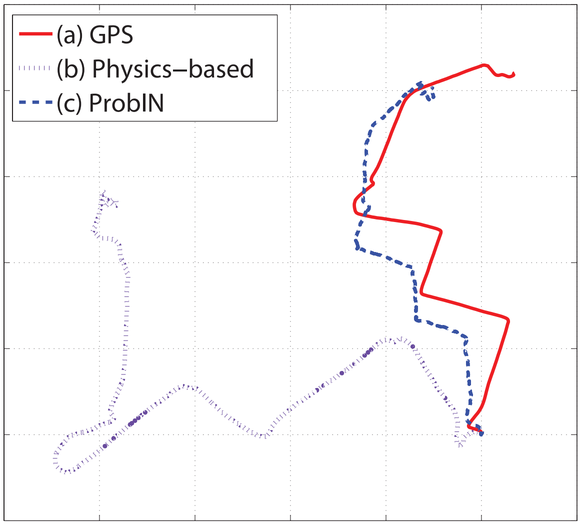


Fig. 9. A comparison of trajectories produced by GPS (a), a physics-based positioning technique (b), and the ProbIN statistical model (c)

In the evaluation phase, the statistical model is applied to translate the raw sensor readings into the displacements, which form the vehicle's trajectory. By applying a map-matching technique introduced in §III the estimated trajectory can be overlayed on the map in order to identify and correct different turns.

We demonstrate ProbIN based on a data sample collected during 22 kilometers of driving. We used the first 18 kilometers of the collected data for training the ProbIN model. Then we used the remaining 4 kilometers for evaluation.

In our approach the estimation provided by the physics model is used for bootstrapping the ProbIN model. However, the trajectory of the physics-based estimation contains an error, which grows with time and traveled distance. Thus, the trajectory of 18 kilometers estimated by the physics model would be inaccurate and inappropriate for bootstrapping the ProbIN model. In our approach we split the long data sequence of 18 kilometers into short sequences based on GPS data. The idea is to identify the locations where the car stopped temporarily (e.g., at a stop sign, at a traffic light, in a traffic jam, etc.). These locations indicate the end of one short sequence and the start of the next short sequence. Thus, instead of using the 18 kilometer long trajectory estimation we utilize the physics based estimation of shorter trajectories, which are more accurate. The short data sequences are then used for creating the ProbIN model, which is then used for estimating the 4 kilometer long test trajectory.

Figure 9 (a) depicts the trajectory of the vehicle based on the collected GPS data in the testing phase. This trajectory is considered to be the true trajectory. Figure 9 (b) is the result achieved by applying the state-of-the-art physics-based positioning technique. As depicted, the estimated trajectory differs significantly from GPS-based trajectory in shape, direction and size.

Figure 9 (c) shows the trajectory estimated based on application of the ProbIN statistical model. The shape of this trajectory is very similar to the shape of the reference trajectory. Based on the shape, different turns can be identified. Additionally, the similarity in the direction and the size of both trajectories allows the application of the map-matching. Then the trajectory can be "snapped" to the road of a map in order to identify the vehicle's current position in the world's coordinate system.

B. Experiment 2: Map Matching

To evaluate the algorithm, we conducted two experiments in different cities and states. We conducted Experiment 2a in Pittsburgh, PA; with a driving distance of ≈ 9.7 km for 15 minutes. Experiment 2a utilized a map of Pittsburgh with an area of size 11 km x 10 km, with a total of 10108 number of points. We conducted Experiment 2b in Mountain View, CA; with a driving distance of ≈ 4.4 km for 13 minutes. Experiment 2b accessed a map covering parts of Mountain View, Sunnyvale, and Santa Clara, with an area of size 12 km x 10 km, with a total of 10096 number of points. This is shown in Table I.

	Experiment 2a	Experiment 2b
Geographical Location	Pittsburgh, PA	Mountain View, CA
Driving Length	≈ 9.7 km	≈ 4.4 km
Driving Time	≈ 15 minutes	≈ 13 minutes
Map Size	11 km x 10 km	12 km x 10 km

TABLE I
DESCRIPTION OF EXPERIMENTS FOR MAP MATCHING.

As mentioned from §III-B, probability of finding a valid match increases by scaling the motion trajectory because of its inherent noise. For both experiments, we repeated the prediction algorithm by varying the lengths of the motion trajectory with stretch factors of 1.0, 1.25, and 1.50 fold. The prediction algorithm formed two starting point clusters for Experiment 2a highlighted in Figure 11(a). We define a cluster to be any five or more predicted points located within 200 meter radius. As shown in the figure, the actual starting point is within one of the two clusters. Similarly, Figure 11(b) for Experiment 2b also illustrates two clusters indicating the probable starting point. The two experiments clearly show that the adversary can accurately pin-point the starting point as well as the traveled route with high probability. A limitation of our implementation is that we often identify more than one possible starting point clusters. However, reducing the search space to a few locations is significant. Moreover, other information can be used to eliminate some of the clusters.

An adversary can deduce a starting point with higher probability for longer trajectories because as the length of trajectory increases, it creates globally unique constraints. We demonstrate this effect empirically in Figure 10. As lengths of the two experiments were varied from 1 km to 9.7 km and 4.4 km for Experiments 2a and 2b, respectively, we find that the accuracy of our starting point prediction increases.

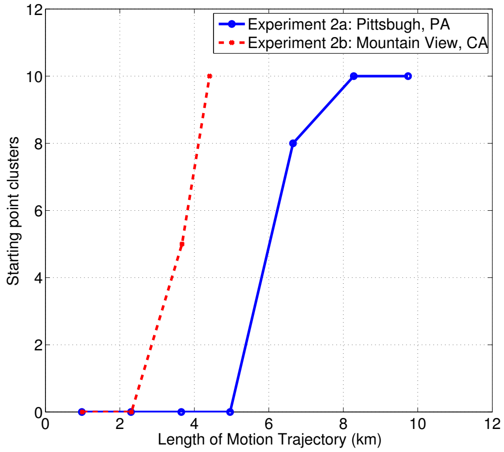


Fig. 10. This figure depicts the crowding of predicted starting point locations, within a 200 meter radius from the true starting point, as the trajectory length increases.

V. Discussion

We now address several topics not yet explained in detail. First, we discuss why an application vetting process may not prevent this sort of attack. Second, we discuss methods for reducing the search space during the map matching phase. Third, we discuss how reducing the sampling frequency can limit the effectiveness of the attack. Fourth, we explore alternative methods for location inference on smartphones.

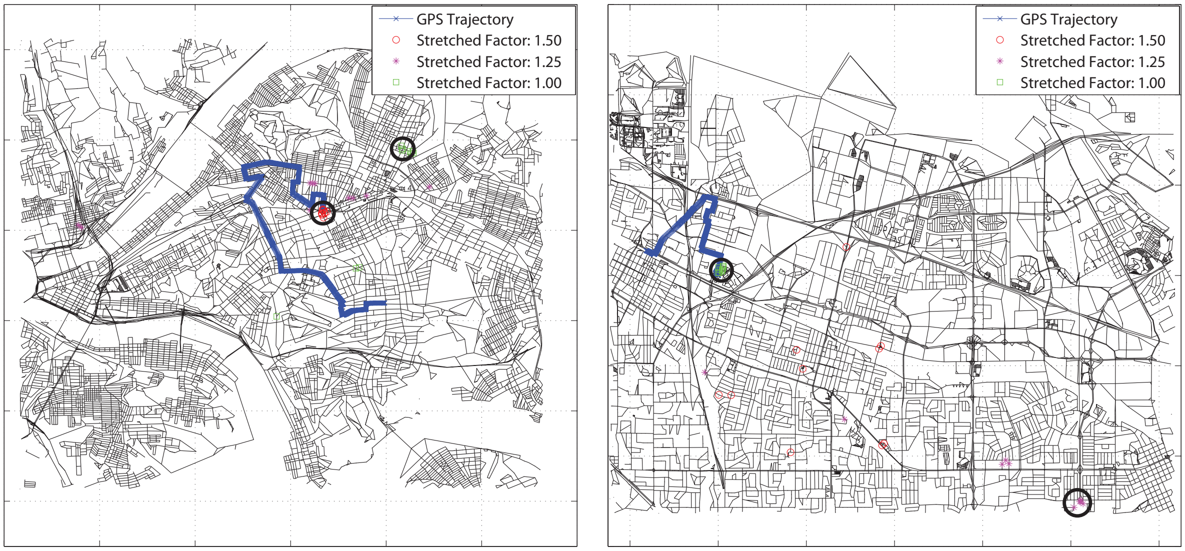
A. Application Vetting

Apple's vetting process may seem to help in detecting applications that attempt to misuse accelerometer readings to infer location. In an effort to identify low-quality and potentially malicious applications, Apple implements a vetting process in which moderators must approve of any application that is developed for the iPhone before it gets distributed to the market. The application vetting cycle is best illustrated by the following process: (1) developers submit an application to Apple, (2) Apple decides to approve or disapprove of the application, (3) the application is distributed to the market via the App Store or iTunes. At first glance it would seem that this vetting process provides for greater security against malicious use of sensory data as compared to models that rely on end users to report suspicious software. However, applications can easily use accelerometer data to violate user privacy even on the iPhone:

- Accelerometer information is currently viewed as harmless; thus an application that uses accelerometer data maliciously is likely to be approved.
- Since it is challenging to detect applications that misuse accelerometer information, application vetting by the marketplace is unlikely to help against this threat.

B. Map Search Tractability

§II showed how one can deduce the trajectory and the starting location from acceleration data given a map of a city. Initially,



(a) Pittsburgh, PA

(b) Mountain View, CA

Fig. 11. This figure shows the top ten predictions made by the map matching algorithm for each stretch factor. Notice that many of the predictions gather in clusters. This is because a trajectory with a slightly different starting point can still trace a similar route as the ground truth; leading to a low *DS*. Experiments 2a and 2b are shown on the left and right, respectively.

the search space may seem intractable to search. This would certainly be an issue if the attacker needed to search the entire map of the United States, let alone the entire world. However in many cases the attacker has access to other sources of information, including publicly available data, that can help reduce the search space from a global scale down to a more tractable area such as a region or a city.

If the attacker knows the name or e-mail address, they can easily reduce the search space using online search (e.g., social networks, city and state public records, etc.). In cases where the victim's phone is accessing a network through Wi-Fi, the attacker can make use of the IP address for geo-ip location lookup.

Our work presents a proof-of-concept of these attacks, with the intent to raise awareness about location privacy leakage associated with accelerometer data. The results presented here use displacements exclusively for starting point prediction. We foresee the possibility of stronger attacks in real world examples if attackers incorporate additional contextual information. For instance, including additional sensors, such as an ambient light sensor and microphone, can further assist in determining where the car is traveling (e.g., through tunnels, or near railroads).

Moreover, different roads have different speed limits, traffic patterns, road surfaces, road angles (both in the direction of road as well as perpendicular), and road features (e.g., bridge transitions, speed bumps, potholes, uneven surface, etc.). Temporal information such as traffic light timings and road congestion information could be used in addition for disambiguation, given the public availability of road congestion data.

C. Sensor Sampling Frequency

The effectiveness of the attack depends on the frequency of the accelerometer readings. Consequently, if the sampling frequency of the sensor is reduced, there is less information to produce an accurate trajectory. The experiments shown here are based on a sampling frequency of approximately 30 Hz. However, the attack is still possible for longer trajectories due to global constraints that accumulate.

D. Alternative Inference Methods

Similar types of attacks are possible with the use of cellular and Wi-Fi signals. While this is true, we claim that our work highlights a more significant attack risk for two reasons. First, inferring a trajectory with accelerometer data is more accurate than using cellular and Wi-Fi traces. As shown from our experiments, our trajectories and matched geo-locations indicate a high degree of accuracy. Second, the method presented in this paper is possible on the large category of devices without cellular or Wi-Fi radio. These devices include PDAs, tablet PCs, digital cameras, MP3 players, and handheld gaming devices. Because these devices are widely used, this may be a prevalent attack. Victims carrying such devices in their bags and/or pockets may be unwittingly sharing their location with malicious software.

VI. Related Work

Several prior works show that accelerometer data collected from wearable sensors can be used for activity recognition [1], [10], [11], [16]. Bao et al. utilized multiple biaxial accelerometers situated on different areas of the body to classify everyday activities with subject-independent training data with

84% accuracy [1]. Ravi et al. found that using a single triaxial accelerometer worn near the pelvic region is sufficient to classify a wide variety of tasks with a high degree of fidelity, but certain tasks that involved only arm movement were comparatively more difficult to identify [16]. Maurer et al., similarly, used a single triaxial accelerometer to identify user activities with great success [11]. In their study, however, the accelerometer was combined with other sensors such as ambient light and sound sensors to classify a wider gamut of applications. Additionally, the accelerometer was placed on the wrist instead of the pelvis. Liu et al. have created the uWave software to use triaxial accelerometer data on mobile phones for highly accurate gesture recognition [10].

There have also been studies on using acceleration data as one part of a set of data to detect user locations [19], [13]. Lee and Mase propose a dead reckoning, or incremental, approach to using human motor data to predict user motion trajectories given a starting point [19]. However, their device utilized a variety of motor sensors worn by human users of which the accelerometer was only one component. Additionally, they tackle the problem of determining location detection in indoor environments via detecting how human users walked. More closely related to our work, Mohan et al. developed Nericell, a software package that piggybacks on mobile smartphones and uses the accelerometer as one of a large suite of smartphone sensors to detect traffic conditions and road deformities such as pot holes [13]. This study also tackles the problem of virtually reorienting the received acceleration data, which can be of arbitrary orientation.

VII. Conclusion

As we demonstrate in this paper, accelerometer readings are highly sensitive. Our results indicate that accelerometers can be used to locate the device owner even if all localization mechanisms on the device are disabled. We illustrate this through a series of experiments conducted in Pittsburgh, Pennsylvania and Mountain View, California. Our proof-of-concept implementation infers a smartphone's location to within a 200 meter radius of the true location.

We hope that this work will encourage future versions of mobile platforms to restrict access to accelerometer information as strictly as microphone and camera sensors.

References

[1] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. pages 1–17. Springer, 2004.

[2] Michael Benisch, Patrick Kelley, Norman Sadeh, and Lorrie Cranor. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing*, pages 1–16. 10.1007/s00779-010-0346-0.

[3] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

[4] Carl Fischer, Kavitha Muthukrishnan, Mike Hazas, and Hans Gellersen. Ultrasound-aided pedestrian dead reckoning for indoor navigation. *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments - MELT '08*, page 31, 2008.

[5] Dan Fletcher. Please rob me: The risks of online oversharing, February 2010.

[6] Clment Fouque, Philippe Bonnifait, and David Btaille. Enhancement of global vehicle localization using navigable road maps and dead-reckoning. 2010.

[7] Joshua Greenfield. Matching GPS observations to locations on a digital map. *Proc. 81st Annual Meeting of the Transportation Research Board*, 2002.

[8] Kashmir Hill. Koreans seek 25 million from apple over iphone location-tracking. August 2011.

[9] J Lester, B Hannaford, and Borriello Gaetano. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In *Proceedings of Pervasive*, 2004.

[10] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657 – 675, 2009.

[11] Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel P. Siewiorek. ewatch: A wearable sensor and notification platform. In *IEEE Workshop on Wearable and Implantable Body Sensor Networks*, 2006.

[12] Declan McCullagh. Microsoft collects locations of windows phone users, April 2011.

[13] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings ACM conference on Embedded network sensor systems*, SenSys, pages 323–336, 2008.

[14] L. T. Nguyen and Y. Zhang. Probabilistic infrastructureless positioning in the pocket. In *Proceedings of The International Conference on Mobile Computing, Applications, and Services (MobiCASE)*, October 2011.

[15] L. T. Nguyen, Y. Zhang, and M. Griss. Probin: Probabilistic inertial navigation. In *The third Mobile Entity Localization and Tracking (MELT) Workshop held at the 7th IEEE International Conference on Mobile Adhoc and Sensor Networks*, November 2010.

[16] Nishkam Ravi, Nikhil D, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pages 1541–1546, 2005.

[17] Greg Welch and Gary Bishop. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel*, pages 1–16, 1995.

[18] Widyawan, Martin Klepal, and Stephane Beauregard. A backtracking particle filter for fusing building plans with PDR displacement estimates. pages 207–212, March 2008.

[19] Seon woo Lee and Kenji Mase. Activity and location recognition using wearable sensors. *IEEE Pervasive Computing*, 1:24–32, 2002.

[20] Heng Xu and Hock hai Teo. Alleviating consumers' privacy concerns in location-based services: A psychological control perspective. In *International Conference on Information Systems*, pages 793–806, 2004.