A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors

Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, Feng Zhao

Microsoft Research Asia
Beijing, China

ABSTRACT

This paper addresses reliable and accurate indoor localization using inertial sensors commonly found on commodity smartphones. We believe indoor positioning is an important primitive that can enable many ubiquitous computing applications. To tackle the challenges of drifting in estimation, sensitivity to phone position, as well as variability in user walking profiles, we have developed algorithms for reliable detection of steps and heading directions, and accurate estimation and personalization of step length. We've built an end-to-end localization system integrating these modules and an indoor floor map, without the need for infrastructure assistance. We demonstrated for the first time a meter-level indoor positioning system that is infrastructure free, phone position independent, user adaptive, and easy to deploy. We have conducted extensive experiments on users with smartphone devices, with over 50 subjects walking over an aggregate distance of over 40 kilometers. Evaluation results showed our system can achieve a mean accuracy of 1.5m for the in-hand case and 2m for the in-pocket case in a 31m$\times$15m testing area.

Author Keywords
Indoor Localization, Inertial Tracking, Pedestrian Model

ACM Classification Keywords
C.2.8 Communication/Networking and Information Technology: Mobile Computing – Support Service

General Terms
Algorithms, Design, Experimentation, Measurement

INTRODUCTION

Indoor navigation is an important enabling technology for applications such as finding a conference room in an office building, safety egress during an emergency or targeted retail advertisement in a shopping mall. While GPS or cell signals are commonly used for navigation in an outdoor environment, robust and accurate indoor positioning remains as an unsolved problem.

Because of the lack of reliable GPS signals inside a building, researchers have explored the use of WiFi beacons [2, 27], magnetometer [6], vision [10], or ultrasound [24], to name a few, for coarse level indoor positioning. But few of them have achieved the needed meter-level accuracy for the above mentioned applications. Besides, WiFi and other infrastructure assisted approaches rely on installation of beacons.

Other approaches rely on inertial sensors (or IMU, short for inertial measurement unit) to track a user by continuously estimating displacement from a known location. The so-called pedestrian navigation system (PNS) [25, 12, 11, 20] is an instance of such a dead reckoning approach, without the need for infrastructure assistance. While most of these existing PNS approaches rely on a dedicated sensor device on the user body for tracking, few has exploited the now widely available smartphone with inertial sensors (e.g., accelerometer, gyro) for indoor positioning, with acceptable accuracy.

In this paper, we developed a practical indoor localization system which relies on smartphone sensors only. To the best of our knowledge, this is the first system that is able to reliably provide the meter-level positioning accuracy for common smartphone users without any infrastructure assistance. Specifically, the contributions of this work are fourfold:

- We designed a reliable algorithm for **step detection**. The algorithm is robust to random bouncing motions, whether the user is walking or not, and to variations in position and orientation of the device on the user body.
- We developed a personalization algorithm that adapts the **stride length estimation** to each user. The algorithm provides an adaptive estimate at every step, and is position and orientation free.
- We designed a **heading inference** algorithm that determines the heading direction of the user at each step.
- We built an **end-to-end system** for indoor positioning using smartphones, and demonstrated that the system can achieve reliable meter-level accuracy.

In the rest of this paper, we first describe the challenges of indoor localization using smartphone inertial sensors, discuss related work, introduce the overall architecture of our indoor positioning system, and then present algorithms for step detection, heading inference, and personalization of step length models. We evaluate the performance of each algorithmic module and the end-to-end system. The paper ends with discussions on the system and future work.

CHALLENGES

In a typical PNS system, the position displacement is determined by aggregating individual steps. Thus, the key is to accurately and reliably detect step boundary and estimate the associated step length, for every step. Doing this on commodity smartphones with noisy inertial sensor readings faces significant challenges.

A typical method to detect step boundaries is to analyze the phone inertial sensor readings and thus determine the user walking motion. There are several problems which make reliable step detection hard. During walking, the position of the phone, such as in a hand or pocket, can affect the sensor readings in different ways, making reliable detection across positions more challenging. The random bouncing of mobile phones, caused by putting phone in a pocket, switching from left to right hand, operating on touch screen, or taking a phone call, can generate false positives in the detection.

Once a step is detected, the system needs to estimate the step length. The step length of a person's walking can vary quite a lot over time, due to speed, terrain, and other environmental constraints. A common approach to estimation is to relate the sensor readings to step length using a model. The step length model may be trained using collected user data offline. But a one-size-fit-all model is going to produce large errors for people with more extreme walking characteristics.

It is well understood that people with different physical profiles such as height, weight, sex, or walking style have different step length. Even for the same person, the step length can vary due to differences in shoes or ground surface. What is desired is the ability to adapt a trained generic model for a specific user over time, as the more personal sensor data is being accumulated.

Another challenge for the system is to infer the heading direction at each step accurately, despite the fact the phone can be at any position on the body. An overall positioning system needs to integrate the step length estimate, together with the heading information, to produce a reliable displacement estimate. Due to the well-known problem in drifting of inertial sensors, one needs additional constraints to keep the error accumulation at bay. One source of constraints is the indoor floor map, which constrains the motion with corridors and walls.

RELATED WORK

Many existing efforts leverage some form of infrastructure assistance for positioning. WiFi-based techniques are commonly used due to the broad availability of WiFi infrastructure. WiFi triangulation makes use of receiving signal strengths from access points (APs) and a propagation model of the signal to compute the receiver location. However, as this requires information about the AP positions, and the propagation model can vary significantly indoors, the accuracy of the WiFi triangulation can be uneven. The EZ algorithm [3] computes both the receiver as well as the AP positions, along with parameters of the propagation model, by solving a set of simultaneous nonlinear equations. But this can suffer from the convergence problem during the nonlinear equation solving as well as the poor signal quality.

WiFi fingerprinting does not require prior knowledge about APs and a propagation model. Instead, it builds a RSSI fingerprint map and looks up the associated location by matching the signature with the fingerprint map [2, 27]. The method can be sensitive to device heterogeneity and requires building the fingerprint map. Other infrastructure-based localization solutions have also been proposed [6, 10, 24], with some limitations due to the limited coverage of infrastructure or heavy deployment and training requirement.

The work on PNS [25, 12, 11, 20], has focused on IMU based solutions. Woodman and Harle [25] proposed a dead reckoning approach, using a foot-mounted IMU device for data collection. Klingbeil and Wark [11] requires the user to carry a sensor board at a fixed orientation. The work of [7] leverages the phone inertial sensors for outdoor localization, and [18] for turn detection indoors.

A pedestrian model, including step detection and a step model, is a key component of a PNS system. Step detection typically relies on peak detection over accelerometer data [14], which is sensitive to noise and other irrelevant motion, producing a high rate of false positives. Other methods rely on detecting features such as zero-cross [13, 5] or flat [17, 26]. Dynamic time warping was used to align similar waves [23] or classify activities [19].

Several step detection solutions require a user to carry a sensing device and attach it to the body such as on the foot, making the detection problem easier. Detection methods based on phone sensors [1, 16, 8] have also been proposed, with limited accuracy in step counting and detection. A number of step models have been proposed based on acceleration [21], walking frequency [5], or a combination of them [9]. Personalization of step model has also been considered, with [4] proposing to automatically learn model parameters using GPS readings outdoors.

OVERVIEW OF THE SYSTEM

The indoor localization system is comprised of several modules, the step detector, step length estimator, heading estimator, particle filter, and a personalization module for adapting a step model to an individual user, as in Figure 1. The system interacts with a user to get the initial location through user input, and provides the current position estimate on an indoor map. In addition, a sensor module provides IMU readings to these estimators continuously.

Distance estimation: In the system, the position update is performed at each step. Every time a new step is reported by the step detector, a new length estimate is generated by the step length estimator based on a step model, triggering the particle filtering. The step model can be either an initial generic model, or a personalized model learned locally from

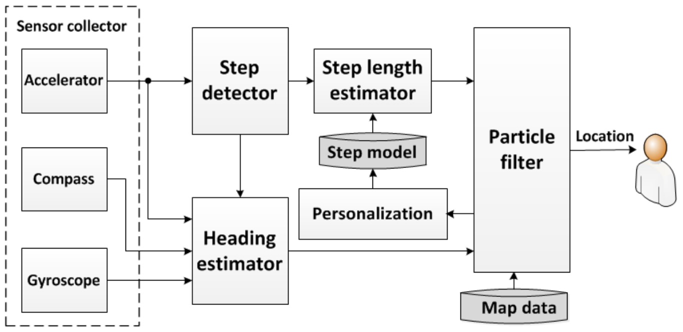


Figure 1. The overall system architecture.

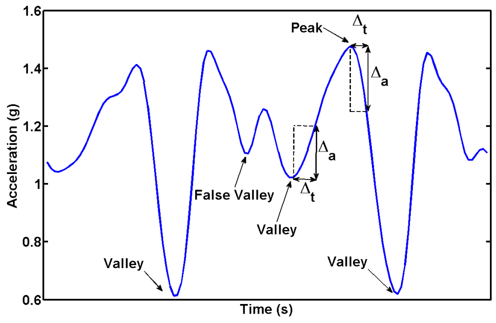


Figure 2. Peak detection on acceleration waveforms.

the latest step data of the user.

User heading inference: The heading estimator fuses data from the compass, gyro, and accelerometer, to ensure high reliability and accuracy of user heading inference at each step, no matter which position the phone is put into. The orientation of the phone itself can be obtained, for example on Windows Phone, via the Motion API [15] or computed from the raw sensor data.

Particle filtering: We use the particle filter algorithm, a non-parametric form of Bayesian estimation, commonly used in computer vision and tracking, to compute the overall position. The algorithm integrates information from the step length estimator and heading estimator, in addition to the constraint from the floor map, to arrive at the posterior distribution of the position. It is customized to support on-the-fly personalization of step model, which differentiates it from previous algorithms.

RELIABLE STEP AND HEADING DETECTION

The step detection and heading inferences are two key modules in the overall location system, as described above. Once the system is initialized, it continuously detects any step of a user with high reliability, and for every step detected, it estimates the heading direction using a heading inference algorithm.

Step Detection

Accelerometer has been commonly used for step detection. Considering the fact a mobile phone can be at any position on the user body, with time-varying orientation change, we use the magnitude of 3-axis accelerometer reading instead of its vertical and horizontal components.

To recover the true periodicity of the signal, the step detection algorithm first uses a low pass FIR digital filter with a certain cut-off frequency to remove the high frequency noise and spikes in raw acceleration magnitudes. The cut-off frequency is currently set to 3Hz based on the statistics of our sample data on walking frequencies of the test subjects, and is tunable based on the subject profile.

After smoothing the raw acceleration data, the algorithm searches for the peaks and valleys of the waveform in order to identify a distinct step. An example of the peak detection is given in Figure 2. Two thresholds, Δ_a and Δ_t, are used to filter out false peaks caused by acceleration jitters that are either too small in magnitude or too short in time duration.

However, not all the detected peaks correspond to meaningful step boundaries, hence the false positives of steps. Accidental bouncing of a phone could produce such a false step. The algorithm further reduces false positives using heuristic constraints and dynamic time warping (DTW) validation.

After analyzing the acceleration magnitudes of over 10,000 real step data points from 40 subjects, we applied two heuristics to reduce erroneous detections: (1) Maximum time duration of a step (e.g., 1 second); and (2) Minimum & maximum changes in acceleration magnitudes during one step (e.g., 0.2g and 2g).

DTW has been widely used as an efficient way to measure the similarity between two discrete waveforms. A lower DTW distance denotes a higher similarity. For step detection, considering in normal walking when a user alternates her left and right foot, it can be expected similar waveform will repeat at every other step. Based on this observation, we designed a DTW validation algorithm to determine whether a step detected via peaks is a real step.

The DTW validation works as follows. Suppose the peak detection yields a series of detected steps $\{S_1, S_2, \ldots, S_n\}$. Then, for S_i, the algorithm calculates the DTW distance between S_i and S_{i-2}. If it is lower than a given threshold, both S_i and S_{i-2} are considered as real steps. Otherwise, S_i will be temporally labeled as false until S_{i+2}, when S_i has another chance to be validated after computing the distance between S_{i+2} and S_i. The distance is calculated over normalized accelerometer waveforms, to reduce the influence of amplitude. In our test cases, the distance threshold is set to 3 based on the statistics gathered over 10,000 step data points mentioned earlier.

User Heading Inference

Heading determination is an important component of inertial sensor based positioning. The phone heading may be different from the user heading during walking, with the difference called phone heading offset. As the orientation of a phone can vary during walking, reliable user heading inference can be an extremely difficult task. To the best of our knowledge, there has been no reliable solution.

In this study, we address the problem by first considering the case where the phone is put in a user's pant pocket, and develop a novel heading estimation method. We then relax the assumptions made about the pocket case, and introduce a more general mechanism for correcting errors in the heading estimation.

The assumptions about the phone-in-pant-pocket case are as follows:

Assumption 1: The initial phone heading offset is known. This is reasonable as we need to initialize the relation between user heading and phone heading so that the user heading can be tracked based on the sensor tracking of the phone heading. An example is when a user looks at her phone, with the phone pointing in the direction of her walking, and then puts the phone in the pocket.

Assumption 2: During the walking, the phone heading is relatively stable with the leg movement. This is also quite reasonable as the phone moves with the leg in the pocket, exhibiting a periodic pattern in heading change.

We use the following example to illustrate how the user heading direction is computed. Let the Euler angles (yaw, pitch, roll) denotes the user heading. The user initially operates on the phone. At this point, the yaw of the phone heading is the same as that of the user heading, while the roll of the phone is zero regardless of the pitch. We use orientation quaternion to represent the phone orientation so that the so-called gimbal lock problem can be avoided [22].

Initial Orientation Inference

After a user puts the phone into her pocket, we need to get its orientation quaternion using the following method:

- If the user remains stationary during the phone motion ending in the pocket, we can directly get the orientation quaternion.
- If not, we can get the orientation quaternion at the so-called inference point of the user's first step. The inference point is a point in time with a particular combination of movement characteristics that repeats at every step. We will explain it in detail below.

Assuming the original orientation quaternion of the phone is V_h, and the orientation quaternion of the phone in the pocket is V_p, we have

$$V_h * Q^{-1} = V_p \qquad (1)$$

where Q is the transformation matrix. Since V_h is similar to the orientation quaternion of the user V_u with the only difference in pitch, they have the same heading direction. Hence, we use Q as the transformation matrix from V_p to V_u, which has no influence on the heading direction inference.

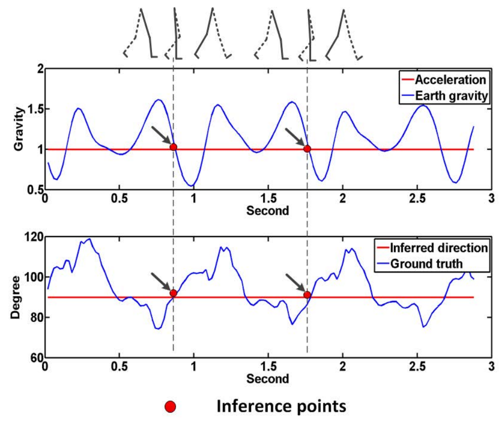


Figure 3. Inference point: user acceleration (top) and inferred heading direction (bottom).

Heading Direction Update

Because of the periodicity of the leg movement during walking, we identify a point during each step when the relative orientation of the phone to the user body is the same as in the original state, i.e., when the phone was just put into the pocket. We call this point the inference point as mentioned earlier; at this point, the transformation matrix from V_P to V_u is the same as the one initially computed. We can thus infer the user heading based on the current phone orientation and the transformation matrix.

The top of Figure 3 shows an example of the inference points at which the acceleration crosses the normal earth gravity value from peak to valley. This corresponds to when the left leg bypasses the right leg during walking. The bottom figure plots the inferred direction computed using the transformation matrix. One can see at these inference points the inferred direction is consistently close to the ground truth heading direction we labeled.

Heading Direction Adjustment

We now relax the previous assumptions about phone heading, and also account for heading biases caused for example by magnetic interference. As we described earlier, the particle filter applies the estimated heading direction to move particles. A perturbation is usually added to account for sensor noise.

Likewise, we develop an algorithm to automatically adjust the particle direction in the particle filtering process, to account for heading estimation errors. At each step, the motion direction of a particle i, θ_i, is computed as

$$\theta_i = \theta_i^e + \delta\theta_i \qquad (2)$$

where θ^e is the estimated heading at current step, and $\delta\theta_i$ the noise following a certain zero-mean distribution such as a zero-mean Gaussian. We now add an additional correction term to the particles at the next step, computed as

$$\theta^a = K * \frac{\sum_{i=1}^{n} \delta\theta_i}{n},$$

where n is the number of surviving particles after the cur-

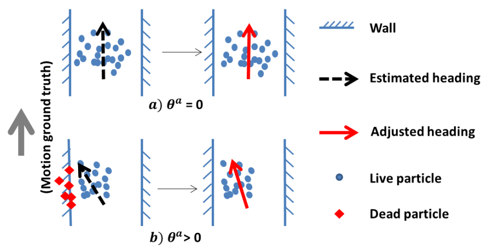


Figure 4. An illustrative example of heading adjustment.

rent step, and K is a scaling factor for how aggressively one wants to adjust the heading direction.

An example is where a magnetic interference causes the heading estimate to consistently biased towards a wall, leading many particles to hit the wall and perish. The surviving particles are those whose perturbation noises have likely helped to correct their course, leading to their survival. Thus, the correction term computed from these surviving particles can help the particles to steer away from the wall, as illustrated in Figure 4.

ADAPTIVE STEP LENGTH ESTIMATION

There are several barriers to getting an accurate step model that works for everyone, all the time.

- It is impractical to ask each user to manually label her step data so that the system can learn her personal step model, before the system can start to work on the user phone. As an alternative, a generic step model should be provided at the time of service delivery.
- People's step models vary a lot, making the generic model less accurate. Physical profiles such as height and weight affect the model accuracy.
- Even for the same person, her step model may change over time due to variations in shoes, ground types, or health status, making model personalization a continuous process.

Addressing the above challenges, we propose a personalization algorithm that starts with a generic step model, and then collects the user data on the fly to learn a personal model. It is designed to adapt to model changes over time.

There are a number of step models proposed in the literature, among which frequency models have been widely used. We set up a system to collect over 4000 steps from 23 users with diverse physical characteristics. Figure 5 gives a snapshot of the tool we have implemented to label the ground truth of collected steps.

After we have got all the ground truth data, we analyze the relationship between the walking speed and actual step length, as shown in Figure 6. There is a clear trend that the step length has a linear relation to the walking frequency. Based on that, we choose a frequency model[28] as our generic step

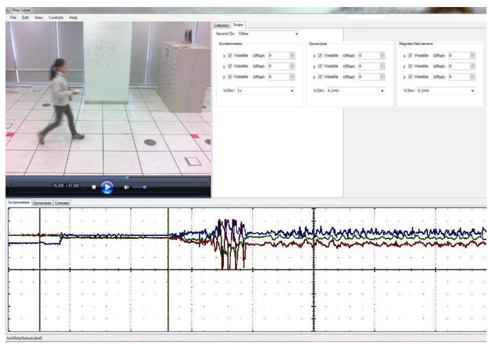


Figure 5. Step data collecting and labeling system.

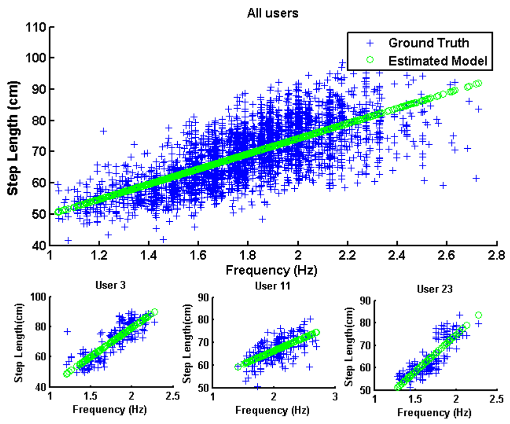


Figure 6. Step length vs. walking frequency.

model, represented as

$$L_g = a * f + b \tag{3}$$

If we take a closer look at each person's data in Figure 6, the linear frequency model fits well, with differences only in the coefficients. With this observation, the goal of our personalization algorithm is to determine the value of the coefficients a and b for each person, as described next.

POSITION UPDATE AND PERSONALIZATION

Position update is performed on a per step basis. Every time a step is detected, our system will acquire the step length based on step model and step frequency, and infer the heading direction of that step as well. Both step length and heading direction are fed into the particle filter module together with the map information.

Before describing the specific algorithm, let's look at how the particle filtering works. Each particle moves according to the dynamics of the model step by step and is constrained by the map. If a particle hits a wall, it will die, and a new particle will be generated around another live particle. Particles whose trajectories align with the observation will have a higher probability to live. Thus, the positions of live particles reflect the estimated position of the user.

We design the step model personalization using the particle filtering over an appropriate state space. As we have stated above, the goal of personalization is to get the pair of the coefficients (a, b) that best describe the ground truth model of the user. Thus, if we associate each particle a different coefficient pair, we can imagine that over time, particles with coefficient pairs closer to ground truth will have higher probabilities to live. As a result, their coefficient pairs can better reflect the ground truth coefficient pair, i.e., the user's personal model.

The customized particle filtering algorithm perform position update as well as personalization. Specifically, the algorithm updates the following state model:

$$\mathbf{x} = \{(x, y), P_{(a,b)}\} \tag{4}$$

where (x, y) represents the coordinate of a particle in a 2D space. $P_{(a,b)}$ represents the coefficient pair, i.e., the step model associated with the particle; we refer to it as the *personalization model* below.

Initially, all particles spread around an initial point specified by the user. During walking, the re-sampling step removes the particles with very low weight, generates new particles from the distribution, and normalizes the weight of each particle. As for the particles representing the model coefficients, initially, they are placed around (a, b), i.e., the generic model. Specifically, for a given particle, its personalization model is represented as:

$$P_{(a,b)} \quad = \quad (a + \delta_a, b + \delta_b) \tag{5}$$

where δ_a and δ_b are noise with a zero-mean uniform distribution. Accordingly, the initial expectation of the personalization model is the generic model.

During walking, if a particle dies and needs to be re-sampled, the algorithm assigns a new personalization model to it. As stated above, the basic idea of personalization is to remove any particle model that is not well aligned with the ground truth. If a particle dies because of inaccuracy in its personalization model, it will be re-sampled with the same personalization model as the nearby live particle; otherwise its personalization model remains unchanged. This is because besides the particle model, heading direction also affects particle propagation. Even with a perfect personalization model, a particle may still hit the side wall due to heading direction error. In this case, the personalization model should not be responsible for the particle death and need not to be replaced.

How do we tell when a personalization model is inaccurate? In our implementation, we use turn detection to determine whether to replace the model. Specifically, while walking along a corridor, heading direction error is the main reason for particles to hit the wall and die. So the personalization model remains unchanged before and after re-sampling. On the other hand, when a user is making a turn, model inaccuracy becomes the main reason for particle death due to deviations from the turn. In this case, the personalization model is replaced after the re-sampling.

Turn detection: The basic idea of turn detection is to compute the change of heading direction, and compare it against a given threshold θ_{th}. When the heading direction change goes above the threshold, a turn is reported. Specifically, the turn detection is performed in three steps:

- Median and mean filtering: An n-point median filter is first applied to the heading direction to filter out sharp direction changes not caused by making turns. Then an m-point mean filter is applied to remove random error of estimated directions. In our implementation, we set n to 9 and m to 3.
- Turn detection: given a filtered direction θ_i for current step i, we first compute the mean direction of all the steps detected since last turn. Then, the direction change at step i is computed as

$$\delta\theta_i = \theta_i - \sum_{j=1}^{i-1} \theta_j \tag{6}$$

  If $\|\delta\theta_i\| \geq \theta_{th}$, a turn is detected.
- Turn initiation and termination: once a turn is detected, we can get the exact point in time by searching for a salient direction change point in a small window around the current step; the size of the window is related to the size of median and mean filter. The turn termination point can likewise be determined by searching for the point when the heading direction goes back to a stable direction again.

Once the turn termination is determined, the re-sampled particles adopt the personalization models of those nearby.

Particle propagation: All particles update their coordinates according to:

$$\begin{aligned} x(t+1) &= x(t) + ((l(t) + \delta l(t)) \cos(\theta(t) + \delta\theta(t)) \\ y(t+1) &= y(t) + ((l(t) + \delta l(t)) \sin(\theta(t) + \delta\theta(t)) \end{aligned} \tag{7}$$

Where $l(t)$ and $\theta(t)$ are the estimated step length and inferred heading direction, while $\delta l(t)$ and $\delta\theta(t)$ zero-mean Gaussian noise on the length and heading, respectively. Additionally, a heading correction term may be added as discussed earlier. The variances of $\delta l(t)$ and $\delta\theta(t)$ are determined by the confidence of step length estimation and heading inference.

Particle correction: This step sets the weight for every particle. If a particle crosses a wall during propagation, we set the weight of that particle to 0. If the particle is in the middle part of the road, we increase the weight of the particle. This is based on the assumption that people tend to walk in the middle of a corridor and cannot walk across a wall.

After the particle correction, the user location is updated to the centroid of the particles. At the end of each session, the personalization model is also summarized and stored for subsequent uses, as follows. We first get p particles with the highest weights. We then choose q frequencies that lie within a normal walking frequency band. Applying the models of these particles to the frequencies generates qp points of (frequency, step length). The summary personalization model is obtained by a linear regression on these points.

Table 1. False positives of step detection (TC1, BM1).

Algorithms	PD	PD+H	PD+H+DTW
FP	101	29	14

Table 2. False positive and false negative of step detection for the in-hand and in-pocket cases (TC2, BM1).

Algorithms	PD	PD+H	PD+H+DTW
FP(%)	7.7 / 8.9	1.54 / 3.48	1.15 / 1.29
FN(%)	0 / 0	0.3 / 0.5	0.4 / 0.6

EVALUATION

In this section, we evaluate the performance of the key modules as well as the end-to-end system for indoor localization we have implemented.

Step Detection

False negative (FN) and false positive (FP) are the key metrics for the step detection algorithm, and directly affect the accuracy of the overall localization. We consider two test cases for the evaluation:

Test Case 1 (TC1): The subject operates on the phone such as making a call, texting, playing games, or moving the phone randomly, without taking any step.

Test Case 2 (TC2): The subject walks in free style with her phone at a certain position. The ground truth will be labeled by recording the entire walking process.

We use the following benchmarks for comparing the performance of the step detection algorithm:

Benchmark 1 (BM1): The baseline peak detection algorithm based on peaks and valleys, described earlier and widely used in the literature. By comparing against it, we examine the effects of the heuristics and DTW validation in our algorithm.

Benchmark 2 (BM2): Off-the-shelf step detection applications on widely used smartphone platforms: NSC (Symbian) - Nokia step counter, PED (iOS) - pedometer, and NW (Android) - Noom Walk.

Table 1 shows the step detection result for *TC1* with respect to *BM1*. PD refers to the baseline peak detection algorithm, PD+H the algorithm with the heuristics, PD+H+DTW the algorithm with the heuristics and DTW validation. In the experiment, 8 people with different physical profiles used their phones in free style for a certain time period without taking any step. We then apply each algorithm to the sensor data collected, and compute the total steps. It can be seen that PD produces 101 false steps, which can be reduced to 29 by applying the heuristics. The DTW validation further reduces the number of false steps to 14.

Table 2 shows the step detection result for *TC2* with respect to *BM1*. All subjects walk with their phones being held in the hand and put in the pant pocket (two separate phones). One can clearly see the trend of decreasing false positive with the heuristics and DTW validation being applied. The cost is a slightly increased false negative since in some rare cases, a real step might be erroneously filtered out by the heuristics or DTW validation. This is acceptable since the benefit more than compensates for the cost here.

Table 3. False positive of step detection (TC1, BM2).

Algorithm	PHD/NSC	PHD/PED	PHD/NW
FP	4/12	3/27	7/18

Table 4. Accuracy of step detection for the in-hand and in-pocket cases (TC2, BM2).

Algorithm	PHD	NSC	PED	NW
Error(%)	1.6 / 1.1	2.4 / 2.9	24.5 / 22	15.2 / 25.6

It should be noted that for *BM2*, since those applications only report the final count instead of a specific time stamp for each detected step, it is impossible to get false positive and false negative for them. Instead, we use the step detection error to evaluate the performance of each application.

Table 3 shows the step detection result for *TC1*, with respect to *BM2*. To ensure the fairness, for any pair of devices under comparison, we bind the devices together, for similar motion input. In the table, PHD is our algorithm with the PD+H+DTW combination. It can be seen our algorithm produces far fewer false positives than any of the three benchmark applications.

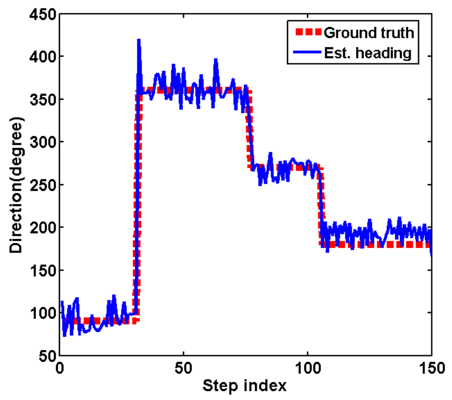
Table 4 shows the step detection result for *TC2*, with respect to *BM2*. As in the earlier case, the subjects walk with their phones being held in the hand or put in the pant pocket (two phones). Among the three benchmark applications, NSC is the only one with a reasonable accuracy at both positions (2.4% for in-hand and 2.9% for in-pocket). Using our algorithm, the error can be further reduced to 1.6% and 1.1%, respectively.

Heading Inference

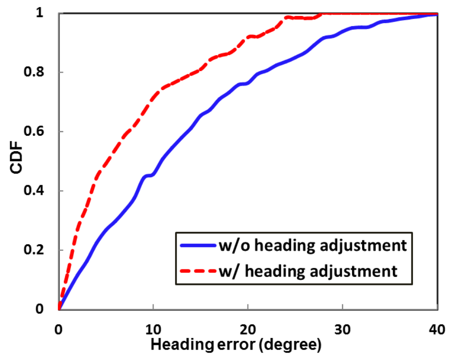
In this experiment, 5 subjects were asked to walk along a given rectangle trajectory. The ground truth is the direction of the pathway. In each round of the test, the user first opened our application, put the phone into the pocket, and then started to walk.

Figure 7(a) shows the heading inference result for one trace. One can clearly see that the inferred heading direction is close to the ground truth direction.

Figure 7(b) shows the CDF of the heading direction inference error on all the runs of the users. In 95% cases, our algorithm yields less than 32 degrees in error. The average error is 12 degrees, while the maximum at 40 degrees. After applying the heading direction adjustment discussed earlier, the 95th percentile error is reduced to 23 degrees, and the maximum error to 28 degrees.



(a) Estimated heading vs. ground truth.



(b) Heading inference error

Figure 7. Heading inference result.

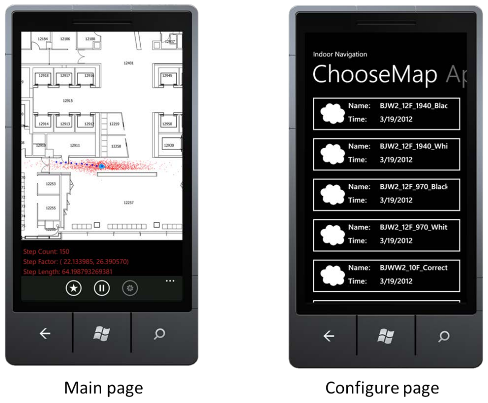

Step Count: 150
Indoor Navigation
ChooseMap
Name: BJW2_12F_1940_Blac Time: 3/19/2012
Name: BJW2_12F_1940_Whi Time: 3/19/2012
Name: BJW2_12F_970_Black Time: 3/19/2012
Name: BJW2_12F_970_Whit Time: 3/19/2012
Name: BJWW2_10F_Correct Time: 3/19/2012
Main page
Configure page

Figure 8. System prototype on Windows Phone.

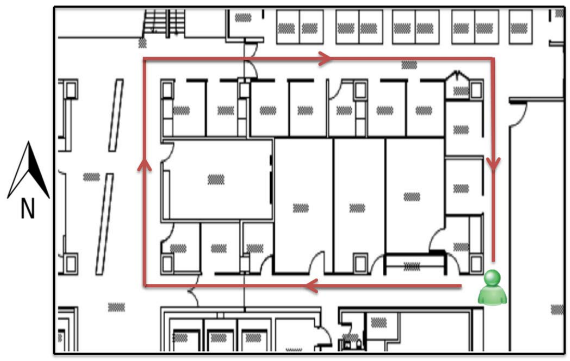


Figure 9. Floor map of the indoor test site.

End-to-End System Evaluation

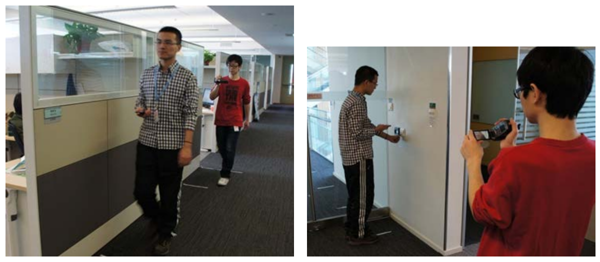
We have built a prototype on Windows Phone 7.5 and tested it on HTC Mazza, as in Figure 8. The floor map is divided into small tiles and each tile is labeled as pathway, room space, or wall. The map is loaded into the phone in the form of an XML file, which contains both the map image and associated metadata. Every time when a user opens the application, a map list will pop out asking the user to select one. After the map is selected and displayed on the screen, the user taps and holds to input the current location. The system then automatically starts to track the user and updates the location on a per step basis on the map. The user can also set parameters such as the sensor sampling rate and particle number, with a tradeoff between positioning accuracy and cost. The default setting in the following experiments are 50Hz sampling rate for all the IMU sensors and 3000 particles for the particle filter.

We have tested our system in an office building, with the floor map shown in Figure 9. Twenty laps of data were collected from 13 subjects with different physical profiles. Each subject walked along a 31×15m area (shown as red solid line on the map), taking two phones with one in the hand and the other in the pant pocket.

To label the ground truth, we place tags on the ground along the walking path at every 50cm, and recorded the entire walking process, as shown in the Figure 10. Then we carefully labeled the coordinates of a subject and the associated timestamp for every step. The positioning error is calculated as the projected distance between the estimated position and labeled ground truth along the walking path.

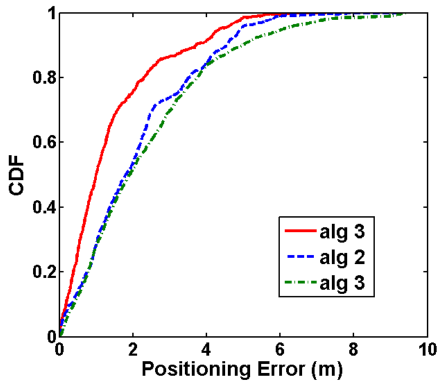
Positioning Accuracy

We evaluate the accuracy of the basic algorithm, and the effect of DTW validation and personalization in improving the accuracy. Figure 11 shows the positioning results of the system. For the in-hand case, with the generic step model, the

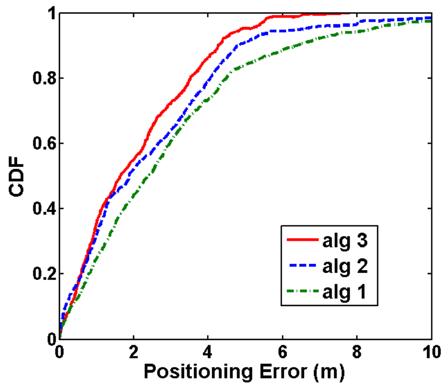


(a) Walking (b) Opening the door

Figure 10. Snapshots of the data collection process.



(a) In-hand



(b) In-pocket

Figure 11. Distribution of positioning error. ***Alg 1*****: generic model, no DTW validation;** ***Alg 2*****: generic model, with DTW validation;** ***Alg 3*****: personal model, with DTW validation.**

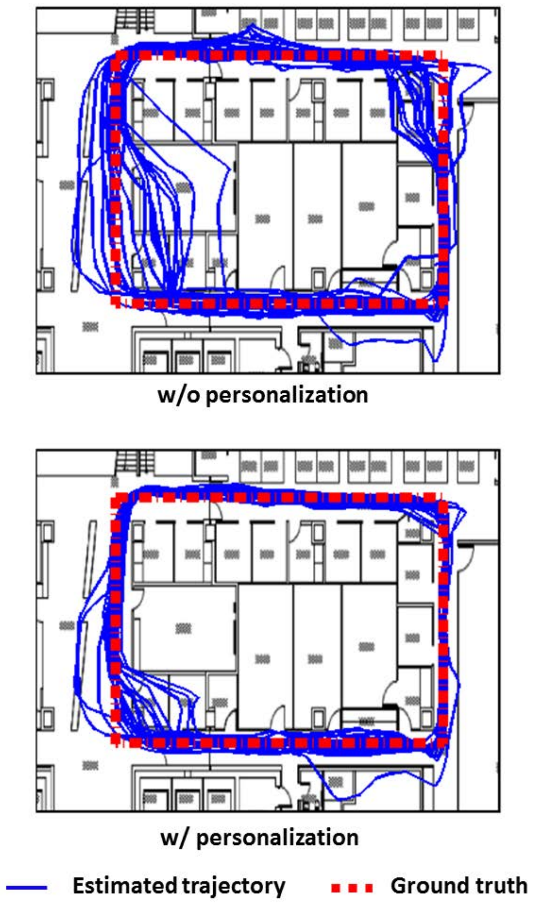


Figure 12. Estimated trajectories vs. ground truth, without (top) or with (bottom) personalization.

mean error is at 2.4m without DTW validation in step detection, and the 95th percentile error at 6.4m. With the DTW validation, the mean error is reduced by 13%, to 2.1m, and the 95th percentile error to 5.1m. When the personalization is applied, the mean error can be further reduced by another 29%, to 1.5m, and the 95th percentile error to 4.3m.

For the in-pocket case, the mean error is at 2.9m using the generic model without the DTW validation, and the 95th percentile error at 8.3m. The DTW validation reduces the mean error by 14%, to 2.5m, and the 95th percentile error to 6.5m. The personalization further reduces the error to 2m (20% lower) and 4.8m, respectively. Compared to the in-hand case, the in-pocket case performs slightly worse since the accuracies of both step detection and heading inference are lower.

It should be noted that in the above experiments, there are cases when the system fails to track users, due to extremely large errors in distance and heading estimation. We did an additional experiment to evaluate the number of failure cases of the system.

We collected 40 laps of data from 10 people with phones held in the hand. The trajectory is still as specified in Figure 9. Each time the system failed to track a subject, the user re-initialized the location to get the system back on track. With the generic model only, the system reports 17 failures. With the help of personalization, the failure number is reduced to 3. The personalization significantly reduces the failure number since most of the failures come from the subjects with extreme step models where the benefit of personalization is more obvious.

Personalization Effectiveness

Personalization improves positioning accuracy by adapting the step model to each user. Figure 12 shows the trajectories estimated with and without personalization. With personalization, the estimated trajectories match the ground truth more closely, although one can still see parts of the trajectories not being well aligned with the ground truth, such as those near the bottom left corner. It should be noted that most of them occur at the first turn, when personalization has not yet been performed.

We also took a deep look at how personalization improves

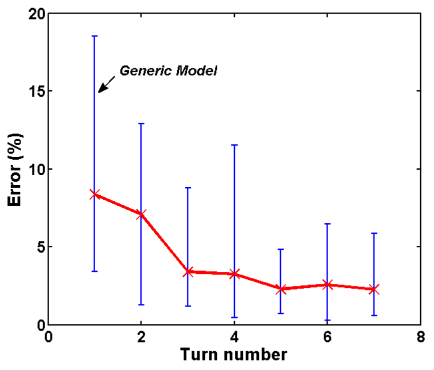


Figure 13. Estimated step length error vs. turn number.

the step length estimation, as more personal data is being collected. Since a personal model is updated every time a turn is made, we group the data of all subjects based on the number of turns they have made, and compute the errors of estimated average step lengths for each group. For example, at the first turn, the estimated average step length is computed based on the generic model since no personal model has been applied yet. Figure 13 shows the the step length estimation error as a function of the turn number. It is clear the step model converges quickly and remains stable afterwards.

DISCUSSIONS

We've developed an end-to-end localization system using inertial sensors on commodity phones, and demonstrated its effectiveness and accuracies in comparison with state-of-the-art.

We plan to improve our systems in several ways. Heading direction inference is still an open problem. In our evaluation, the magnetic interference is one major cause for error. We are exploring ways to calibrate the readings automatically and additional methods to compensate for the error. Although our current implementation does not rely on infrastructure, the system can benefit from infrastructure assistance where it is feasible. We are investigating possibilities of integrating additional signals and constraints such as WiFi or Bluetooth into our system.

REFERENCES

1. All-in Pedometer App. `http://itunes.apple.com/us/app/all-in-pedometer/id368180978?mt=8`.
2. P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-Based user location and tracking system. In *InfoCom*, 2000.
3. K. Chintalapudi et al. Indoor localization without the pain. In *MobiCom*, 2010.
4. D.-K. Cho et al. AutoGait: A mobile platform that accurately estimates the distance walked. In *PerCom*, 2010.
5. S. Y. Cho, C. G. Park, and H. Y. Yim. Sensor fusion and error compensation algorithm for pedestrian navigation system. In *ICCAS*, October 2003.
6. J. Chung et al. Indoor location sensing using geo-magnetism. In *MobiSys*, 2011.
7. I. Constandache et al. Towards mobile phone localization without War-Driving. In *InfoCom*, 2010.
8. L. Fang et al. Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience. *IEEE Transactions on Instrumentation and Measurement*, 2005.
9. J. Kappi, J. Syrjarinne, and J. Saarinen. MEMS-IMU based pedestrian navigator for handheld devices. In *ION GPS*, 2001.
10. N. Karlsson et al. The vSLAM algorithm for robust localization and mapping. In *ICRA*, 2005.
11. L. Klingbeil and T. Wark. A wireless sensor network for real-time indoor localisation and motion monitoring. In *IPSN*, 2008.
12. B. Krach and P. Robertson. Integration of foot-mounted inertial sensors into a bayesian location estimation framework. In *WPNC*, 2008.
13. H. Leppäkoski et al. Error analysis of step length estimation in pedestrian dead reckoning. In *GPS*, September 2002.
14. R. W. Levi and T. Judd. Dead reckoning navigational system using accelerometer to measure foot impacts. US Patent, December 1996.
15. Motion API. `http://msdn.microsoft.com/en-us/library/hh202984(v=vs.92).aspx`.
16. Noom Walk Pedometer. `http://android-apps.com/applications/health-fitness-applications/noom-walk-pedometer-alpha`.
17. NOKIA Step Counter. `http://betalabs.nokia.com/apps/nokia-step-counter`.
18. K. Park et al. Smartphone-based pedestrian tracking in indoor corridor environments. *Personal and Ubiquitous Computing*, Dec 2011.
19. A. Parnandi et al. Coarse in-building localization with smartphones. In *MobiCASE*, pages 343–354, 2009.
20. A. Serra et al. Indoor pedestrian navigation system using a modern smartphone. In *Mobile HCI*, 2010.
21. S. H. Shin et al. Adaptive step length estimation algorithm using low-cost MEMS inertial sensors. *IEEE Sensors Applications Symposium*, February 2007.
22. M. D. Shuster. The quaternion in kalman filtering. In *AAS/AIAA Astrodynamics Conference*, 1993.
23. S. Sprager and D. Zazula. Impact of different walking surfaces on gait identification based on higher-order statistics of accelerometer data, 2011.
24. S. P. Tarzia et al. Indoor localization without infrastructure using the acoustic background spectrum. In *MobiSys*, 2011.
25. O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *UbiComp*, 2008.
26. Yamax Pedometer DIGI-Walker. `http://www.yamaxx.com/digi/index-e.html`.
27. M. Youssef and A. K. Agrawala. The Horus WLAN location determination system. In *MobiSys*, 2005.
28. W. Zijlstra. Assessment of spatio-temporal parameters during unconstrained walking. *European Journal of Applied Physiology*, 92(1), June 2004.