Using Smart-Phones and Floor Plans for Indoor Location Tracking

Kun-Chan Lan and Wen-Yuah Shih

***Abstract*—We implement pedestrian dead reckoning (PDR) for indoor localization. With a waist-mounted PDR based system on a smart-phone, we estimate the user's step length that utilizes the height change of the waist based on the Pythagorean Theorem. We propose a zero velocity update (ZUPT) method to address sensor drift error: Simple harmonic motion and a low-pass filtering mechanism combined with the analysis of gait characteristics. This method does not require training to develop the step length model. Exploiting the geometric similarity between the user trajectory and the floor map, our map matching algorithm includes three different filters to calibrate the direction errors from the gyro using building floor plans. A sliding-window-based algorithm detects corners. The system achieved 98% accuracy in estimating user walking distance with a waist-mounted phone and 97% accuracy when the phone is in the user's pocket. ZUPT improves sensor drift error (the accuracy drops from 98% to 84% without ZUPT) using 8 Hz as the cut-off frequency to filter out sensor noise. Corner length impacted the corner detection algorithm. In our experiments, the overall location error is about 0.48 meter.**

***Index Terms*—Floor plan, map matching, pedestrian dead reckoning (PDR), simple harmonic motion (SHM), waist-mounted, zero velocity update (ZUPT).**

I. Introduction

A PERSONAL dead reckoning or pedestrian dead reckoning (PDR) system is a self-contained technique for indoor localization. PDRs may include wearable inertial sensors, thus eliminating the need for beacon nodes or RF maps/propagation models based on surveys of the environment. These inertial sensors (such as an accelerometer, gyroscope, or digital compass) are used to measure step length and heading direction.

Depending on where the sensors are placed, previous studies generally classified PDR systems into two types: foot- and waist-mounted. The foot-mounted method [3]–[10] uses a double integral on horizontal acceleration to estimate distance and a gyroscope or compass to measure the heading direction. The waist-mounted [13]–[16] method tries to detect each step event to calculate the total number of steps, and then multiplies this by a constant step length which is based on the pedestrian's characteristics (weight, height, and age) to estimate the total moving distance. Some waist-mounted methods use linear regression to find the relationship between the acceleration, walking speed, and step length [20]. The limitation of such approaches is the requirement of a training phase to estimate the step length for each individual. While waist-mounted systems are implementable on a hand-held device, their step length estimation accuracy is typically worse than foot-mounted systems which perform poorly with regard to orientation accuracy [17].

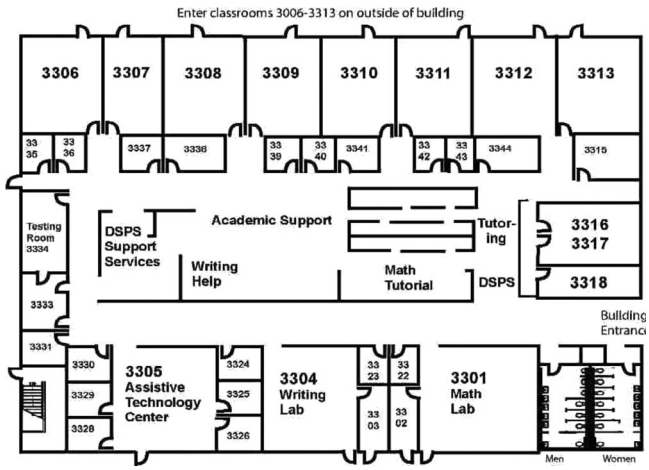

Fig. 1. Floor plan.

Two types of errors can be observed due to PDR sensor hardware: systematic (such as sensor bias, sensitivity, and drift [11]) and random (such as environmental changes [8], [9]). Here, we focus on the sensor drift error and noise. We propose a new zero velocity update (ZUPT) method and a map matching algorithm to calibrate the sensor drift errors from the accelerometer and gyro, and use a low-pass filtering mechanism to reduce the noise in our system.

Here, we consider a scenario in which the user has a smart-phone and can access the floor plans of the building (see Fig. 1). The system utilizes the sensors on the smart-phone to compute the user's moving distance and direction and, in combination with the map and the last-recorded GPS position, estimates the user's current position (see Fig. 2).

We make the following assumptions.

1) The floor plan can be characterized using a link-node model [12] in which pathways are the links and the intersections of pathways are the nodes. The user carries a smart-phone in his or her pocket.
2) The initial heading direction is known (the user can point the phone in the direction headed before putting the phone in his or her pocket).

Manuscript received March 10, 2013; revised August 12, 2013, October 5, 2013, and December 18, 2013; accepted December 20, 2013. Date of publication January 16, 2014; date of current version March 12, 2014. This paper was recommended by Associate Editor A. Karniel.

K.-C. Lan is with the Department of CSIE, National Cheng Kung University, Tainan, Taiwan (e-mail: klan@csie.ncku.edu.tw).

W.-Y. Shih is with the Department of CS, National Chiao Tung University, Hsinchu, Taiwan (e-mail: Todd629.cs01g@nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2013.2296875

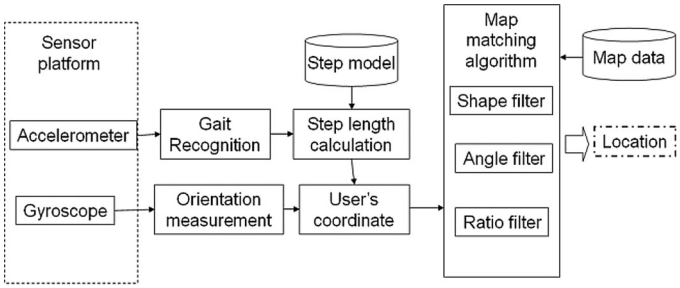

Fig. 2. System architecture.

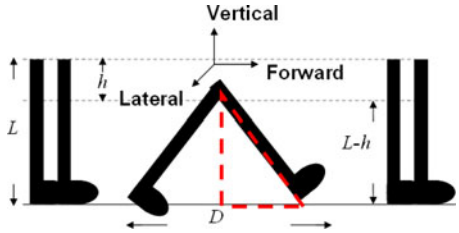

Fig. 3. Walking diagram.

3) During walking, the position of the phone is relatively stable in relation to the leg movements.

Several researchers have investigated using smart-phones for indoor localization. Kaikai *et al*.'s prototype system Guoguo [39] uses acoustic ranging techniques to achieve centimeter-level accuracy. However, their methods require specialized anchor nodes. Wang *et al*. [35] proposed a landmark-based approach to address dead-reckoning error. However, the landmark signature could be different across different phones and their system requires prior knowledge of existing landmarks in the building. Yang *et al*. [40] utilized RSSI and floor plan data for location estimation. However, they did not address the issues that arise when the building has a symmetric structure. In our paper, we complement our trajectory-map similarity analysis with RSSI fingerprints to resolve this problem. Finally, our architecture is similar to that used in Li *et al* [32], but they required the use of a detailed map for the trajectory tracking. In addition, their approach needs a training phase to create a step length model.

Here, we describe our PDR method that estimates the moving distance when a smart-phone is carried in the user's waist level pocket. No training phase is required for initialization. Exploiting geometric similarity between the user trajectory and the floor map, we also present our map-matching algorithm to address sensor errors.

The rest of this paper is structured as follows: the step-length estimation and map-matching algorithms are presented in Sections II and III, respectively. The results of our experiment appear in Section IV. Finally, we conclude in Section V.

II. Waist-mounted PDR Method for Estimating Step Length

Weinberg [16] observed that the upper body moves vertically when walking, and suggested that one can estimate the step length as follows:

$$\text{StepLength} = 2 \times \text{heightchange}/\alpha \tag{1}$$

where α is the swinging angle of the leg from the horizontal plane, and the heightchange can be estimated based on the vertical acceleration. However, he did not discuss how to measure α, and we found that, in practice, it is difficult to measure such a small angle during walking. Based on the Pythagorean Theorem, we thus propose a different way to estimate step length using the change in height. During walking, a person's body moves up and down. If we assume the length of the leg is L, the waist-line will move up-and-down between L and *(L-h)* from the ground, where h is the change in the height of the waist. Consider the triangle in Fig. 3, formed by the person's two feet and his or her step length D. Given that L is known, using the Pythagorean Theorem, we can estimate D if we know the height of this triangle, i.e., *(L-h)*. To obtain *(L-h)*, we need to first calculate h which is the change in height of the waist during walking. Therefore, if we mount an accelerometer on the user's waist, the readings can be used to estimate the height change h, which can then be used to calculate the step length D based on the Pythagorean Theorem (the length of the leg (i.e., L) and half of the step length (i.e., $D/2$) forms a right triangle in which leg length is the hypotenuse).

In this paper, we extend the aforementioned waist-mounted method for a hand-held device, such as a smart-phone. When implementing a PDR system on a hand-held device, two cases can be considered. The first is when the user holds the device (e.g., talking on the phone), and the other is when the device is put in a pocket or a bag. [21] has shown the feasibility of using a waist-mounted method for the first case. Therefore, here we focus on how to extend the results of a waist-mounted method for the second case. To implement a waist-mounted method on a hand-held device, one must consider: the orientation and placement of the device. The orientation of the device may change from time to time when it is put in a pocket or bag during walking. Therefore, we need to determine the orientation of the device with respect to the walking direction to estimate the height change. We employ the technique proposed in [35] to reorient three axes of the accelerometer. In addition, we adopt a method similar to that used in [21] (which used a gyroscope to record changes in orientation) to estimate the height change, as shown in (2). Here, θ is the orientation of the device in relation to the direction of gravity and is between 0 and 90 degrees, and N is the acceleration which is caused by external force

$$\begin{cases} N = (\text{SensorReading} - \text{Gravity} \times \cos\theta) \div \cos\theta \\ \text{HeightChange} = \int\int N dt. \end{cases} \tag{2}$$

Next, in a waist-mounted method, the vertical movement displacement of the device is the key parameter to estimate the step length. When the device is positioned above the waist-line, its vertical displacement during walking will be the same as when it is mounted on the waist. When the device is placed below the waist-line (e.g., in a pant's pocket), we can estimate the vertical displacement of the waist as follows. Assuming the leg length

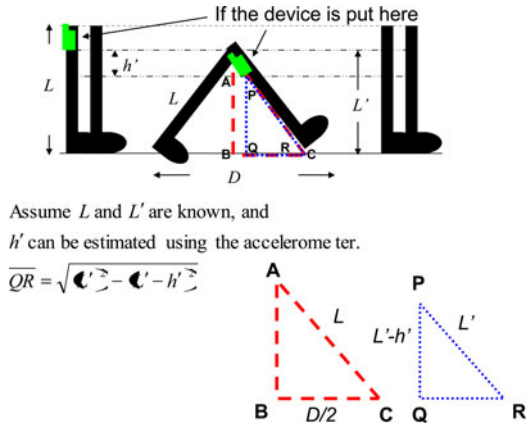

Fig. 4. Use of Pythagorean Theorem for computing the step length when the device is mounted on the lower body.

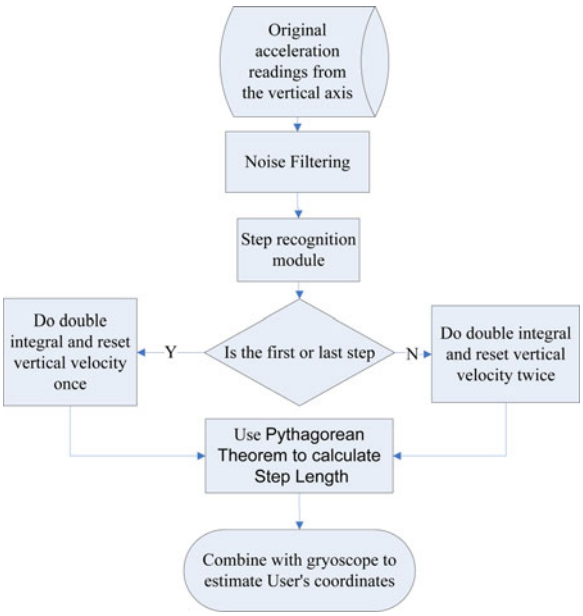

Fig. 5. Flow chart of our PDR system.

(L) and the pocket position from the ground (L') are already known, we can find two similar triangles ΔABC and ΔPQR (see Fig. 4). Based on the Pythagorean Theorem, we can obtain $\overline{QR}$ by first measuring L' and h' (i.e., the vertical displacement of the pocket during walking). Then, using triangle similarity, we find that $\overline{BC} = D/2 = \left(L \times \overline{QR}\right)/L'$. h' can be measured in a similar way to h.

Our system architecture is shown in Fig. 5. We first filter the noise using a low-pass filter. The filtered signal is fed into the step recognition module to identify each new step. We adopt the concept of simple harmonic motion (SHM) [18] to reset the vertical velocity at the beginning of each new step, which prevents sensor drift errors from being accumulated over to the next one. Finally, the step length is estimated based on the filtered sensor data and the foot length.

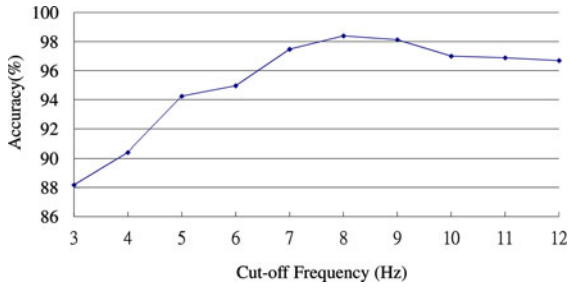

Fig. 6. Accuracy of estimating step length using different cut-off frequencies for the low-pass filter.

A. Noise Filtering

During walking, some unexpected and unpredictable body vibrations might cause some higher-frequency noise in the sensor readings. One can use a low-pass filter and preset a cut-off frequency to filter the noise, and some prior step recognition systems use 3 Hz as their cut-off frequency to detect a step event [2]. However, we found that while a 3-Hz threshold can detect a new step event, it is too low for our purposes and will remove data which is not noise. [19] analyzed the acceleration of the waist during walking, and found the maximum acceleration is 8 Hz. To examine whether 8 Hz can produce the best results, we perform a set of experiments using different frequencies for the low-pass filter, that range from 3 to 12 Hz, and compare their accuracies in estimating step length. The use of 8 Hz as the cut-off frequency produces the most accurate results (see Fig. 6). Therefore, we set our cut-off frequency at 8 Hz to filter the noise, and this threshold worked well under ten repeated experiments using three different subjects with various leg lengths from 68 to 95 cm.

In addition, given that the phone can bounce around in the pocket and create many false peaks in the acceleration waveform [37], we further apply two filters to reduce these acceleration jitters. The first is the minimum and maximum changes in acceleration magnitudes during one step (after analyzing our data, we set the thresholds to 03. and 1.8 g, respectively). For the second filter, we adopt the dynamic time warping technique to detect false peaks based on the assumption that the waveforms of each step are similar [38].

B. Step Recognition Module

In order to recognize a step, we first analyze the components of one step that can cause vertical changes to the body. There are three major events which may affect the height of the waist [see Fig. 7(a)]. The first one is a heel-touching-ground event, which happens when the heel just hits the ground and the waist is in its lowest position during the entire step. The event that comes after this is the stance, which occurs when the foot is flat on the ground. Finally, the heel-off-ground event occurs right after the stance. Generally, as shown in Fig. 7(b), the vertical acceleration of a heel-touching-ground event is the local minimum within a step. Given that human walking frequency is never over 3 Hz [42], the duration between two consecutive heel-touching-ground events must be over 0.33 s. Based on this, we use a

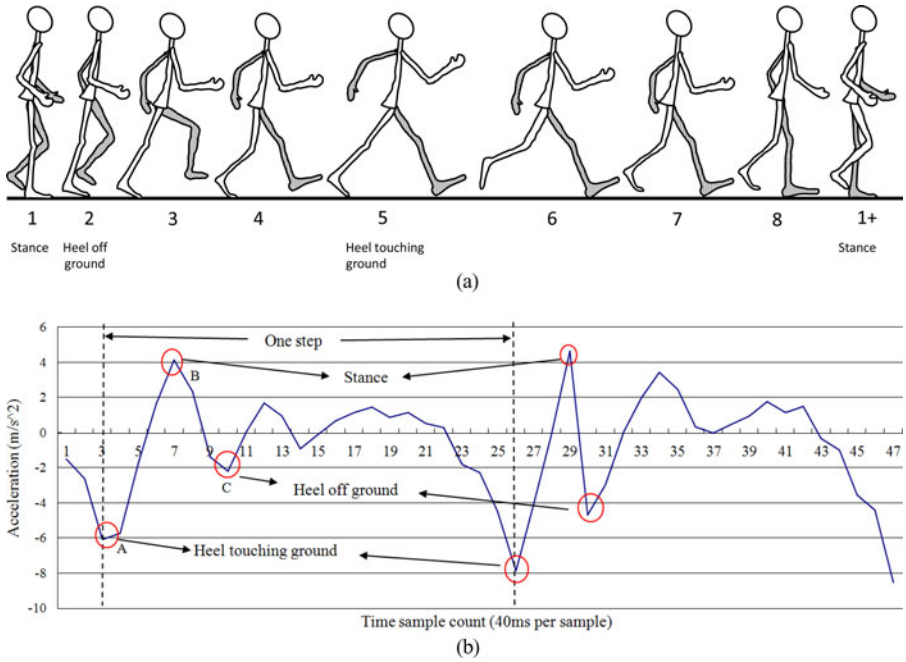

Fig. 7. (a) Walking diagram (Modified from [41]). (b) Vertical acceleration of walking.

sliding window algorithm to detect every heel-touching-ground event, and define one step as from a heel-touching-ground event to the next heel-touching-ground event. Once a new step is identified, the sensor data between two consecutive heel-touching-ground events will be used to estimate the step length.

C. Step Length Estimator

1) Our Step Model: To measure the walking distance from point A to point B, we sum the step length of all steps. We model a complete step as from one heel-touching-ground-event to the next. The first step is defined as from the stance event at point A to the next heel-touching-ground event, as shown in Fig. 7(a). The last step is defined as from the last heel-touching-ground event to the stance event at point B. Both the first step and the last step are considered as half a step. When the first or last step is detected, our system will divide the calculated step length by 2.

2) Avoid Accumulation of Sensor Drift Errors: Once each step can be identified, we can then do the double integral to calculate the change in height of the waist and then use this information to estimate the length of each stride based on the Pythagorean Theorem. However, as discussed previously, if the system only naively does the double integral on the accelerometer data, the sensor drift errors could accumulate from one step to the next. To avoid this, we previously proposed a ZUPT method [1] to calibrate the sensor data: if we mount a sensor on the waist of a pedestrian, then while he or she is walking, the trajectory of the sensor can be approximated by SHM [18]. In addition, given that the velocity of the highest and lowest points of the sensor will be zero, we can utilize these characteristics to detect when the user starts a new step. Finally, once the points with zero velocity in the vertical direction [where the vertical acceleration reaches its local minimum and local maximum, i.e., points A and B in Fig. 7(b)] are identified, we can then reset the vertical velocity to zero before doing the double integral to calculate the change in height of the user's waist, and this can then be used to estimate the step distance based on the Pythagorean Theorem.

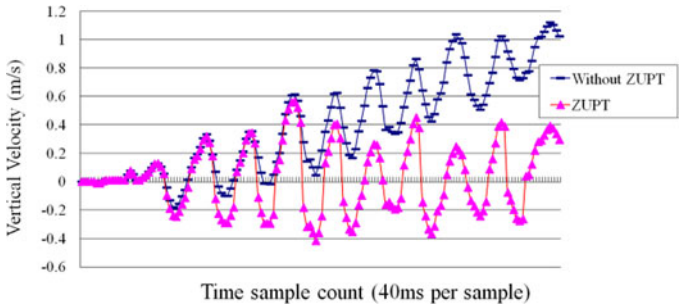

Fig. 8. Effect of ZUPT on the estimated velocity.

As shown in Fig. 7(a), when the stance and heel-touching-ground events occur, the waist has the largest displacement from its equilibrium position. Therefore, we reset the vertical velocity to zero at these points (so-called 'ZUPT'). We performed an experiment to observe the effects of the ZUPT. We recorded the walking velocity computed by the sensor every 40 ms for the cases of enabling ZUPT and disabling ZUPT respectively. Each case was run for 20 s. The implementation of ZUPT can address the accumulation of sensor drift errors (see Fig. 8).

The first and last steps are considered as special cases. There is only one point where the velocity needs to be reset, because the initial velocity of the first step is already zero and the last step does not need to reset this when the body is in its stance state.

3) Moving at High Speed or at the Same Spot: As shown in Fig. 7(b), when one walks at a normal speed, the stance event generally occurs as the first peak after the heel-touching-event. However, during the high speed movement (which is defined

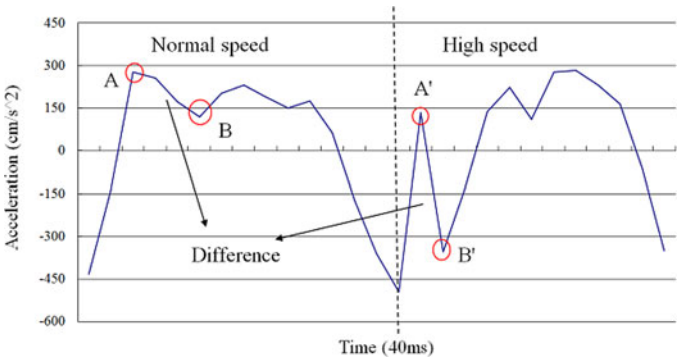

Fig. 9. Effect of counter-force when walking at a high speed.

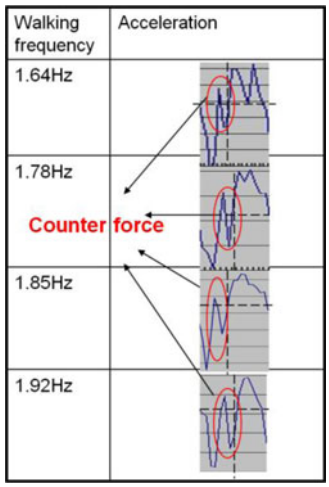

Fig. 10. Acceleration at different high speeds.

as when the step frequency is greater than1.6 Hz), we observe that the counter-force from the ground could introduce *an extra pulse* between the heel-touching-ground and stance events, as shown in Fig. 9 (points A′ and B′) and Fig. 10. Without taking this into consideration, our system could reset the velocity at the wrong point. Studies in kinesiology [30], [31] showed that the duration from the heel-touching-ground event to the stance event normally accounts for at least 16.7% of the time in a step. Based on this observation, we can identify the right point where the stance event occurs and reset the velocity when calculating the step length.

When one is moving in place, the acceleration data should not be considered in the calculation of step length. In addition, when one is walking in place, the height of the waist does not usually change significantly. Therefore, we use a threshold-based filter to detect this phenomenon by looking at whether the acceleration data in all three directions (vertical, horizontal, and lateral) are lower than a certain threshold ε, as shown below, and then reset the vertical acceleration accordingly (in our implementation, ε is set to 1 m/s^2), as shown in

$$\begin{aligned} &\text{verticalAcc} = 0 \\ &\text{if } |\text{lateralAcc}| < \varepsilon \,\&\, |\text{horizontalAcc}| < \varepsilon \,\&\, |\text{verticalAcc}| < \varepsilon. \end{aligned} \tag{3}$$

III. Gyroscope Error Calibration with Map Matching

The effectiveness of a PDR system lies in its success in accurately estimating the user's moving distance and direction. We discussed how to use ZUPT to reduce the sensor drift error when measuring the distance. In a PDR system, the direction in which the user is heading is most commonly obtained from a gyroscope sensor (The smart-phone compass for orientation does not work well in indoor settings [34]). However, a gyroscope can only produce the relative angular displacement (RAD) of a device with respect to a specific direction, and this is not necessarily the absolute direction. Therefore, while we could track the user's trajectory using the gyroscope, this might be biased by the error in its initial direction and appear as a rotated version of the true path, as in Fig. 11(b). When the gyroscope error is significant and left uncorrected, it can make the entire PDR system unusable. Here, we use the map matching technique to address such errors.

Map-matching is the process of comparing the pedestrian's trajectory data with a digital map of the environment to match the trajectory data to the route segment on which the pedestrian is walking, and it can be used to correct the heading error of a PDR system [33]. [31] tries to match the user trajectory to the closest junction and road on the map while [22], [23] utilizes the map to filter out positions where the user is unlikely to move (e.g., walls and obstacles). Both techniques require the use of a detailed scaled map of the building, although in practice this is usually difficult to obtain. Here, we propose a new map-matching method utilizing the more commonly-seen building floor plans to address gyroscope errors. We assume that a floor plan is an "approximate" scaled down version of the physical layout of the floor. Our basic idea is to utilize the geometric similarity between the trajectory data and the floor plan to infer the ***last-visited corner*** by the user. The flow chart of our algorithm is shown in Fig. 12. Before map matching, we convert the floor plan into a link-node model in which information such as the turning angles of the corners and comparative ratios of the lengths between any two corridors can be estimated [12] (see Fig. 13). The link-node model is used to approximate the layout of corridors and corners. We then compare the geometry of the user trajectory with the link-node model to find the possible routes that the user has traveled. Here we consider the map and the trajectory as two independent graphs, say M and T. We list all subgraphs of M and compare T with them to find the most similar one. We define the "similarity" between two graphs by comparing their shapes, vertex angles, and relative edge lengths. Once a unique route is identified, this can then be used to calibrate the trajectory data and identify *the most recently visited corner*. Since the location of every corner is known within the floor plan, the system can then localize the user while he or she moves between corners using the dead-reckoning data from the accelerometer. Given that the link-node model is only an approximation of the physical layout of the building (e.g., the link length might not be an exact scaled-down version of the corridor length), the results of this comparison between the map and trajectory could generate multiple candidate routes. Therefore, we also implement an RSSI-based filter by using

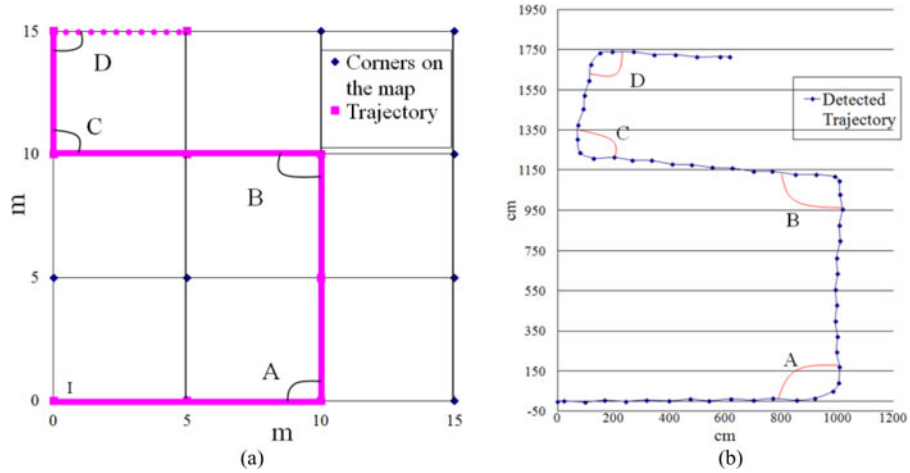

Fig. 11. (a) Link-model of the map and the ground truth. (b) Estimated trajectory from the sensor data.

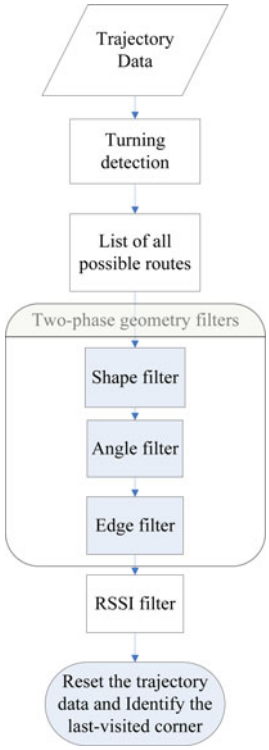

Fig. 12. Flow chart of our map-matching algorithm.

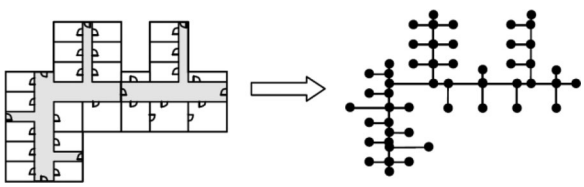

Fig. 13. Link-node model from a floor plan.

the existing WiFi-signal-based landmarks [34] (e.g., a corridor-corner may overhear a unique set of WiFi APs, however the set may change at short distances away from that spot; in addition, some dead spots inside a building may not overhear any WiFi signals, which by itself is a signature). When a WiFi AP signal is

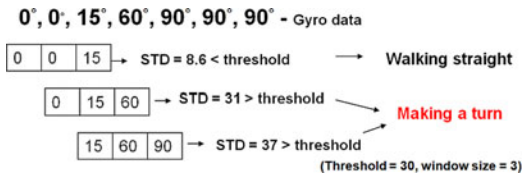

Fig. 14. Example of turning detection.

available, we can use this RSSI-filter to select the correct route from multiple candidates. For example, if we know the user has passed a certain landmark, we can remove the candidate routes without it. This approach is similar to the method used in [34]. The map-matching process is performed every time the system detects that the user makes a turn, and details of the turn detection process are described below.

A. *Turn Detection*

The first step is to identify the turns in the trajectory data, and to use these as a signature to match all possible routes on the map. However, given that the gyroscope can only provide the RAD, we use a sliding-window-based algorithm to infer the user's possible turns. To determine if a person is making a turn, we compare the standard deviation of the window with a threshold estimated during the period when the user is walking forward. A turning event is considered to have occurred when the standard deviation of the window exceeds the threshold. Note that, since it could take several steps to turn around a corridor corner, we record all the turning events that are related to a possible corner-turning event in order to compute the angle of making a complete turn of this corner. One example is shown in Fig. 14. The turning angle (CT) of a possible corner can be estimated using (4)

$$CT = AT - BT. \tag{4}$$

Here, AT is the heading angle after making a turn around a corner, and BT is the heading angle before making a turn around a corner. For example in Fig. 14, $CT = AT - BT =$

90–0 = 90. However, in reality, it is not necessary that one only makes a turn when encountering a corner. For example, one might walk back and forth along the same corridor/aisle. In addition, possible gyroscope drift errors can also produce false turning events. Therefore, detection of a turning event is not necessarily an indication that the user is indeed passing a corner. We consider these kinds of turning events, which are detected when the user is not passing a corner, as 'fake' turnings. Nevertheless, it is difficult to distinguish normal corner turnings from fake ones based only on accelerometer and gyroscope data. Here, we utilize the floor map information to resolve the issue of fake turnings, as follows. We assume that, once the fake turnings are removed from the trajectory data, the trajectory data should be geometrically similar to a possible route on the map.

We first try to find all the possible routes that a user might take (say, R_T) based on all the possible combinations of detected turnings from the trajectory data. We then compare these R_T with all possible routes on the map (say, R_M). The objective here is to find an $((R_T, R_M)$ pair which is "*the most geometrically similar*". We use some methods from image processing theory to solve this problem. We first model the map as a graph, $G_M(V_M, E_M)$. *V_M is the corner of map, such as A in Fig. 11(a), and the E_M denotes the corridor between two adjacent corners. We also define the graph $G_{Mi}(V_{Mi}, E_{Mi})$* as the subgraph of *G_M, which is used to model all possible routes on a map (assuming there are i possible routes). In addition, we model the user trajectory as a graph $G_T(V_T, E_T)$*, where *V_T stands for an ordered set of detected turnings (including fake ones), such as A′,B′,C′,D′ in Fig. 11(b), and E_T is the edge set whose element is the connection between two adjacent detected turnings. That is, assuming $V_T = \{V_1, V_2, \ldots, V_k\}$, V_k is the k-th detected turning, then $E_T = \{\overline{V_1V_2}, \overline{V_2V_3}, \ldots, \overline{V_{k-1}V_k}\}$. We next define the graph $G_{Tj}(V_{Tj}, E_{Tj})$ as follows. There exists a set V' which is the power set of V_T (the set that contains all subsets of V_T), i.e., $V' = \{\Phi, V_1\}, \{V_2\}, \ldots \{V_1, V_2\}, \{V_1, V_3\}, \ldots \{V_1, V_2, V_3, \ldots V_k\}\}$*. Here, we let V_{Tj} be an ordered set, $V_{Tj} \in V'$, $|V_{Tj}| > 1$, and j is the number of elements in V'. The E_{Tj} is the edge set which contains the edge between any two adjacent elements in V_{Tj}. In other words, G_{Tj} is used to model all possible routes that could be generated based on the detected turnings (including fake ones), and G_{Mi} stands for the accessible route on the map.

Again, the idea here is to find a (G_{Mi}, G_{Tj}) pair which is the most geometrically similar. In other words, we want to eliminate those hypothetical routes generated in G_{Tj} that cannot be found on the real map. We consider two graphs as geometrically similar if they have similar shapes, angles (i.e., the angle between two connected edges) and edge lengths (after normalization). We design a two-phase filtering mechanism and employ three filters: shape filter, angle filter, and edge filter. In phase one, we input all (G_{Mi}, G_T) pairs into these three filters to remove those which are not geometrically similar. The purpose of phase two is to remove the nonexisting routes caused by fake turnings (such as those marked with dotted green circles in Fig. 22), and we input all (G_{Mi}, G_{Tj}) pairs into these filters to remove the nonexisting routes.

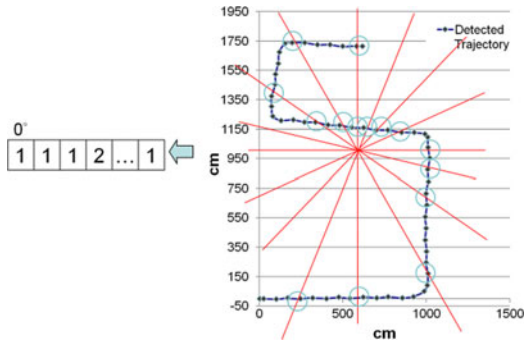

Fig. 15. Example of the shape filter.

The aim of the above two-phase filtering mechanism is to produce a (G_{Mi}, G_{Tj}) pair which is geometrically similar. However, when we do not have "*sufficient*" trajectory data (e.g., when there are no turnings detected in the trajectory data), it is possible that we can still have multiple candidate routes remaining after the geometry-similarity filtering. Therefore, when multiple candidate routes exist, we employ an RSSI-based filter by using the existing WiFi-signal-based landmarks [34], to further select the correct one among multiple candidates. Next, we discuss the details of each filter.

1) The Shape Filter: We adopt the idea of a shape descriptor [24], [25] to implement our shape filter by comparing the shapes of two graphs. Considering the nature of our input data and the computational overhead, we modify the original shape descriptor method to suit our scenario. To reduce the computational overhead, we calculate the centroid [28], [29] of each graph and create a line every 10 degrees from 0° to 180°to pass through this. We then calculate how many crossing points on the edge can be made by each line and record them in a one-dimensional array(see Fig. 15). Finally, we compare the arrays generated based on the two graphs to determine their similarity using Euclidean distance (*L-2* norm) [27]. We set a threshold to judge the similarity between these two graphs, and this is a trained value which can be set based on how similar the graphs should be. If the value of similarity is over the threshold, the system will consider these two graphs are different and thus remove them from the set of candidates.

2) The Angle Filter: We adopt the concept of chain code [26] to implement the angle filter. The chain code uses a sequence of numbers to represent a series of different moving directions and transform a graph to a one-dimensional expression. However, given that what we consider here is a real-time localization system and the computational capability of a smart-phone is limited, we cannot directly apply a full-fledged chain code for two reasons. First, it is difficult to decide the number of sampling points for the trajectory data, since each step distance could be different. Second, the computational overhead of using the chain code is proportional to the number of steps in the trajectory data and the number of candidate routes on the map. We implement the concept of the chain code via two different filters: the angle and the edge ones. Conceptually, the outputs of the chain code include the angle information (e.g., A and A' in Fig. 11), the direction of the edge and the normalized edge ratio (i.e., divide

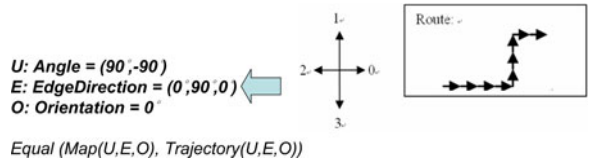

Fig. 16. Inputs of the angle filter: Angle, EdgeDirection, and Orientation.

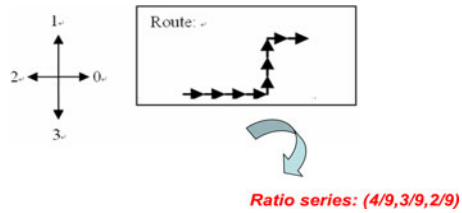

Fig. 17. Input of the edge filter: the distance of each edge divided by the total distance.

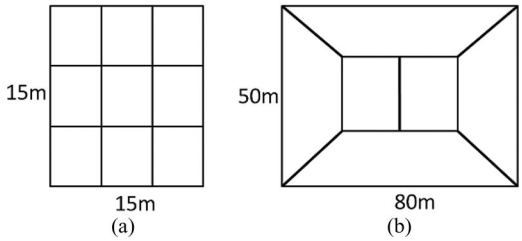

Fig. 18. Two floor plans used in the experiment.

TABLE I
PERFORMANCE OF DIFFERENT PHONE PLACEMENTS

	Phone Location		
	Waist	*Shirt pocket*	*Pant pocket*
Accuracy	98.25%	97.72%	96.97%
Standard deviation	1.29%	2.38%	2.81%

the distance of each edge by the distance of the longest edge). In our angle filter, for example, we compare every matching angle and the direction of each edge between GT and GMi, and use a threshold, which is also a trained value as in the shape filter, to determine whether they are close enough (see Fig. 16). After filtering out those G_{Mi} which do not have similar angles, we then implement the second part of the chain code with an edge filter, as described below.

3) The Edge Filter: With this filter, we check whether the normalized edge ratios between two graphs, for example, G_{Tj} and G_{Mi}, are similar or not. The system calculates the displacement between any two adjacent vertices in the graph and stores these displacements as a vector. Fig. 17 shows an example input to the edge filter. We then use the Euclidean distance and set a threshold to determine whether the vectors produced for G_{Tj} and G_{Mi} are similar or not. If the value of the *L-p* norm of these two graphs is over the threshold, the system then removes the corresponding (G_{Tj}, G_{Mi}) pair from the candidates.

In our experiments, the chosen thresholds are 3.5, 15, and 0.1 for the shape filter, angle filter, and edge filter, respectively. These values are chosen through manual examination of a range of different values.

IV. EVALUATION

In this section, we first discuss the results of our step length estimation method, and then show the performance of the aforementioned map matching algorithm.

A. Experiment Setup

To implement our algorithm, we used a variety of Android phones with embedded accelerometer and gyro sensors from HTC and Samsung. We performed three sets of experiments: one with the devices carried at the waist, one with the device carried in shirt pockets, and one with the device carried in the pants pockets. Each set of experiments was repeated ten times with three different subjects (with height ranging from 158 to 179 cm) for two different maps (see Fig. 18). Here, we present the results with the map in Fig. 18(a). Participants were asked to walk freely along the corridors (i.e., they could wander around or walk back-and-forth). We used a laser distance meter to measure the actual distance traveled. To record the user's actual location, we pasted markers on the ground at known locations. Each marker had a number on it, and the user recorded the number when he or she walked passed it. To estimate the user's leg length L, we created two markers separated by a distance d on the ground; the users stepped on them with the phones in their pockets. The readings of the accelerometer are affected by gravity [1], and are a function of the angle between the measured axis and the direction of gravity. Therefore, by observing the change in accelerometer readings when the leg swings forward, we can obtain the swinging angle (θ) of the leg from the body. We can then calculate L when d and the swinging angle are known based on Fig. 3 (i.e., $L = d/2\sin(\theta/2)$).

B. Step Length Estimation

Some of the state-of-the-art pedometers [43] on the market can also output walking distance. As shown in Table I, we compare our method with these pedometers and find that they only achieve up to 90% accuracy, which is significantly lower the 98% we currently achieved. Note that, this 2% inaccuracy might be because of other sensor errors (such as sensor bias, leveling, and so on), and the imperfect estimation of leg length. Furthermore, as discussed previously, our approach is based on the idea of using the change in height of the waist to estimate step length, which is similar to the method used in Weinberg [16]. In his work, he proposed an equation, as shown in (5), to estimate distance by observing the vertical acceleration during walking. We implemented Weinberg's method and found that its accuracy is about 96.7%. In addition, one limitation of Weinberg's approach is that its system parameters need to be re-trained for every new user

$$\begin{aligned} &\text{StepLength} = k \times n \times \sqrt[4]{\text{Acc}_{\max} - \text{Acc}_{\min}} \\ &k\text{: constant and} n\text{: step number.} \end{aligned} \tag{5}$$

We use a low-pass filter to filter out the noise and employ ZUPT to address the drift error of the accelerometer.

TABLE II
COMPARISON WITH OTHER WAIST MOUNTED METHODS AND THE INFLUENCE OF THE MECHANISM

	Methods					
	Pedometer [43]	***Weinberg***	***Ours (A: LPF, B: ZUPT)***			
Concept	Constant length × Step number	Equation (5)	Ours (A+B)	A	B	none
Accuracy (Estimation/Ground Truth)	90.1%	96.8%	98.3 %	84.1 %	95.3 %	75.5%
Standard deviation	1.8%	3.2%	1.3%	7.4%	2.2 %	16.3%

To understand the effects of these mechanisms, we employ each, one at a time. ZUPT has a greater impact on the system performance than the implementation of the low-pass filter; when ZUPT is disabled the accuracy drops from 98.25% to 84.1% (see Table I). We also compare the performance of our system when placing the phone on different locations of the body (see Table II). The performance of our system degrades when the phone is in the pocket as compared with on the waist (note that our phone-in-the-pocket results are still better than Weinberg's results, which were based on waist-mounted devices). This is not surprising, since the phone generally has less acceleration jitters when it is mounted on the waist compared with when it is in a pocket.

C. Map Matching

As discussed, we use a sliding-window-based algorithm to detect the user's possible turnings from the gyroscope data. To determine if a person is making a turn, we compare the standard deviation of the window with a threshold. Generally, when the chosen window size is too small, all the walking steps that happen during a turn might not be able to be included within a window. When the chosen window size is too big, two different turnings can be included in the same window if the number of walking steps between two corners is less than the window size. The accuracy of detecting turnings is lower when the chosen window size is too small or too big (here "accuracy" is the true-positive rate of detecting a corner) (see Figs. 19 and 20). Therefore, in our experiments, we choose the window size from the range below, in which D is the shortest distance between two adjacent corners, as shown in

$$2 < \text{WindowSize} < \frac{D}{\text{AvgStepLength}}. \tag{6}$$

The threshold is obtained using the standard deviation of the window during the period when the user is walking straight.

In the shape filter, we set a line, every ten degrees from 0° to 180°, that passes through the centroid of the graph, and use the crossing points on the edges obtained this way to determine if two graphs have similar shapes. To understand the effects of the gaps between two crossing lines (default 10°) on determining shape similarity, we vary this gap from 0° to 70°. Generally, the system will have less computational overhead when the gap is larger. However, a larger gap also suggests that poorer results might be obtained, since fewer crossing points will be generated for the comparison of shape similarity. We start getting inaccurate results when the gap is larger than 15° (see Fig. 21).The accuracy here is defined as the success ratio in identifying two graphs with similar shapes.

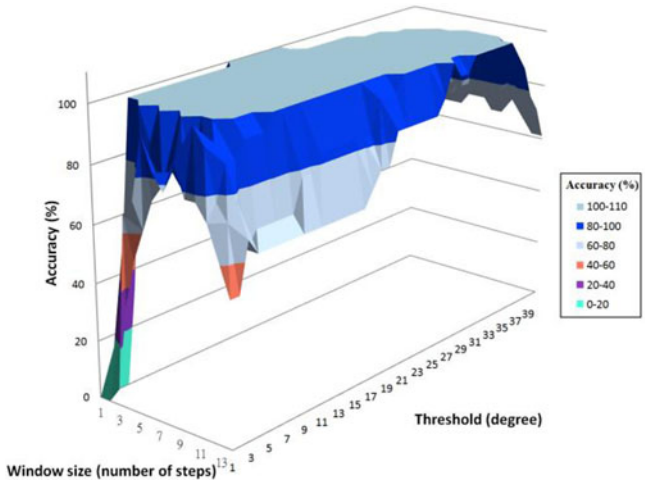

Fig. 19. Number of walking steps between two corners is larger than the window size.

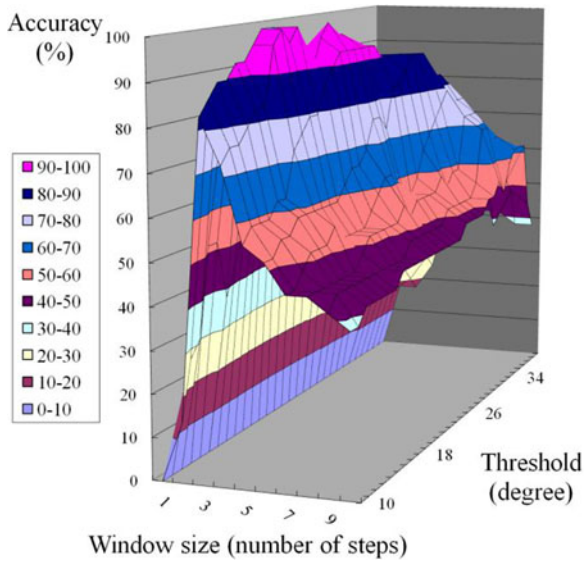

Fig. 20. Number of walking steps between two corners is smaller than the window size.

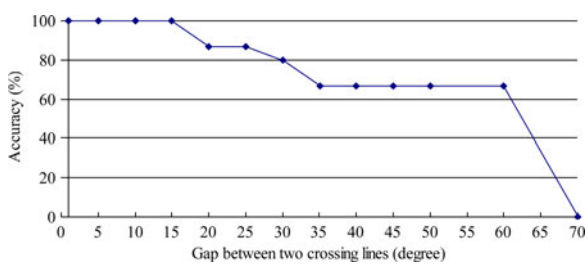

Fig. 21. Effect of gap size on the accuracy of in identifying shape similarity.

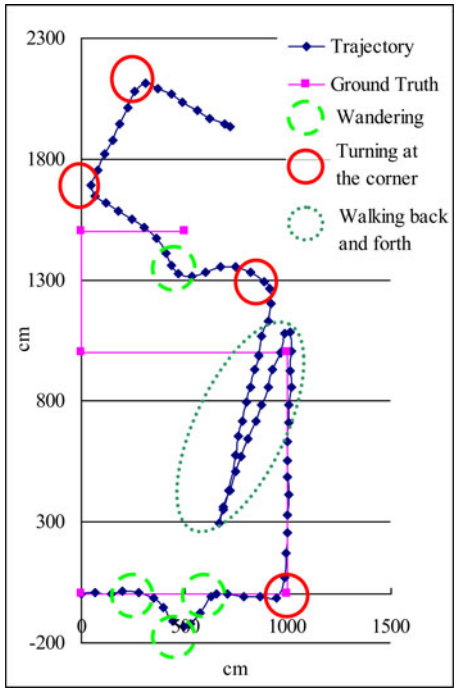

Fig. 22. Testing environment and route.

Fig. 22 shows one of the user trajectories in our experiments, which is about 40 meters long and includes four corridors and corners. In this experiment, the walker deliberately created some "fake turnings" by wandering about around at certain spots, shown as the green dashed circles in Fig. 22. The solid circles in Fig. 22 indicate when this walker made a 'real' turn at the corner. Our system is able to identify the last-visited corner in all our experiments. The location errors are mainly because of the inaccuracy in step length estimation. Therefore, the location error might increase as the length of the corridor increases, since the inaccuracy in step length will accumulate over the walking distance between two corners. In our experiments, the average location error is about 0.48 meter, and the standard deviation is about 0.43 meter. Note that, in our experiment scenarios, we were able to use an RSSI filter to help find the correct route out of multiple candidate routes in 53 cases out of 180 cases (i.e., about 30%).

V. Conclusion

In this paper, we design and implement a waist-mounted PDR method that estimates moving distance when a smart-phone is carried in the user's waist level pocket. Furthermore, by exploiting geometric similarity between the user trajectory and the commonly-seen building floor map, we present a novel map matching algorithm to calibrate the direction errors from the gyroscope. Our results show that we can achieve about 98% accuracy in estimating the user's walking distance, and the average location error is about 0.48 meter in our experiment scenarios.

One limitation of this paper is that we assume that the floor plan is a scaled-down version of the physical layout of the floor. In some cases, this assumption might not be always true. Here, we propose a bootstrap phase based on the participatory sensing approach [35], [36] to obtain the scale information of the map. In this bootstrap phase, the system first collects the users' trajectories, and then compares them with the link-model of the map to estimate the relative distance of every corridor. Note that, while the first few users may experience inferior location accuracy, when more data are obtained the system will converge. In addition, such a bootstrapping phase only needs to be performed once for each building. The duration of this bootstrapping phase depends on the visiting frequency of the participants and how long the participatory sensing data converges to the true scaling value. We carried out an experiment in a local mall by recording the interarrival time between two consecutive visitors for every corridor on its first floor (30 corridors in total) for three days. The mean interarrival time is 2.3 min. To understand how the scaling converges to the true value with the number of participants, we had five people walk around the whole floor at different times to simulate a number of different participants. Here, we define Δas the difference of the estimated length and the real length (L) of a corridor and $s = 1 - \Delta/L$. The average s is close to 1 when we have more than 65 participants in our test scenario, as shown in Fig. 23. In addition, we observe the positioning accuracy (which is defined as the probability of successfully locating the last-visited corner) for our test scenario can reach 100% when the scaling accuracy is higher than 99% (which needs around 35 participants in our experiments. Note that we do not claim that our results are representative since the length of bootstrapping phase is highly dependent on the map topology as well as participants' mobility patterns and visiting frequency which could vary at different times and in different places). Nevertheless, one limitation of this study is its reliance on such a participatory sensing system to collect enough scale information for the map, and we are currently investigating how to overcome this. In addition, our current link-node model does not consider open indoor spaces (e.g., a big hotel lobby), and we leave this for future work.

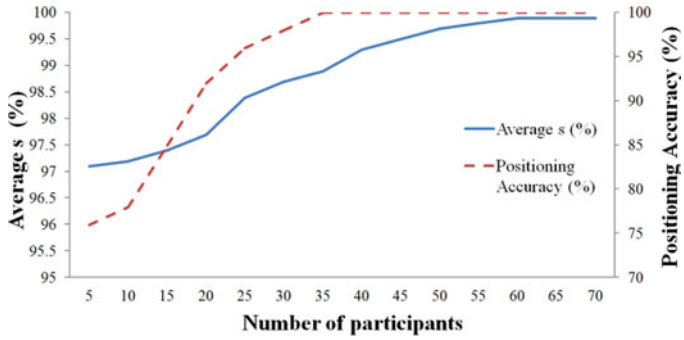

Fig. 23. Influence of participants.

References

[1] K.-C. Lan and W.-Y. Shih, "Using simple harmonic motion to estimate walking distance for waist-mounted PDR," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2012, pp. 2445–2450.

[2] M. Martin and M. Michael, "A step counter service for Java-enabled devices using a built-in accelerometer," presented at the 1st Int. Workshop Context-Aware Middleware Services: Affiliated 4th Int. Conf. Commun. Syst. Softw. Middleware, Dublin, Ireland, 2009.

[3] L. Ojeda and J. Borenstein, "Personal Dead-reckoning System for GPS-denied Environments," in *Proc. IEEE Int. Workshop Safety, Security Rescue Robot.*, 2007, pp. 1–6.
[4] J. C. Alvarez, R. C. Gonzalez, D. Alvarez, A. M. Lopez, and J. Rodriguez-Uria, "Multisensor approach to walking distance estimation with foot inertial sensing," in *Proc. IEEE 29th Annu. Int. Conf. Eng. Med. Biol. Soc.*, 2007, pp. 5719–5722.
[5] M. Dippold, "Personal dead reckoning with accelerometers," presented at the 3rd Int. Forum Appl. Wearable Comput., Bremen, Germany, Mar. 2006.
[6] F. Eric, "Pedestrian tracking with shoe-mounted inertial sensors," in *Proc. IEEE Comput. Graph. Appl.*, 2005, vol. 25, pp. 38–46.
[7] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU," in *Proc. IEEE Int. Symp. Intell. Signal Process.*, 2009, pp. 37–42.
[8] E. Z. RaúlFeliz and J. G. García-Bermejo, "Pedestrian tracking using inertial sensors," *J. Phys. Agents*, vol. 3, pp. 35–42, 2009.
[9] K. Sagawa, H. Inooka, and Y. Satoh, "Non-restricted measurement of walking distance," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2000, vol. 3, pp. 1847–1852.
[10] Y. Xiaoping, E. R. Bachmann, H. Moore, and J. Calusdian, "Self-contained position tracking of human movement using small inertial/magnetic sensor modules," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 2526–2533.
[11] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*. London, U.K.: Peter Peregrinus Ltd, 1997.
[12] P.-Y. Gillieron and B. Merminod, "Personal navigation system for indoor applications," presented at the 11th IAIN World Congr. Smart Navigat., Syst. Serv., Berlin, Germany, 2003.
[13] D. Alvarez, R. C. Gonzalez, A. Lopez, and J. C. Alvarez, "Comparison of step length estimators from weareable accelerometer devices," in *Proc. IEEE 28th Annu. Int. Conf. Eng. Med. Biol. Soc.*, 2006, pp. 5964–5967.
[14] F. Lei, P. J. Antsaklis, L. A. Montestruque, M. B. McMickell, M. Lemmon, S. Yashan, F. Hui, I. Koutroulis, M. Haenggi, X. Min, and X. Xiaojuan, "Design of a wireless assisted pedestrian dead reckoning system – the NavMote experience," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 6, pp. 2342–2358, Dec. 2005.
[15] S. H. Shin, C. G. Park, H. S. Hong, and J. M. Lee, "MEMS-Based personal navigator equipped on the user's body," presented at the ION GNSS 18th Int. Tech. Meeting Satellite Division, Long Beach, CA, USA, Sep. 13–16, 2005.
[16] H. Weinberg, "Using the ADXL202 in Pedometer and Personal Navigation Applications," Appl. Notes An-602, Analog Devices, Inc., Norwood, MA, USA., 2002.
[17] R. Stirling, J. Collin, K. Fyfe, and F. Lachapelle, "An innovative shoe-mounted pedestrian navigation system," presented at the Eur. Navigat. Conf. GNSS, Graz, Austria, Apr. 22–25, 2003.
[18] C. Speaks, "Simple harmonic motion," in *Introduction to Sound*, USA: Springer, 1992, pp. 43–90.
[19] K. Y. Chen and D. R. Bassett, Jr., "The technology of accelerometry—based activity: monitors: Current & future" *Med. Sci. Sports Exerc.*, vol. 37, pp. S490–S500, 2005.
[20] S. H. Shin, C. G. Park, J. W. Kim, H. S. Hong, and J. M. Lee, "Adaptive step length estimation algorithm using low-cost MEMS inertial sensors," in *Proc. IEEE Sensors Appl. Symp.*, Feb. 6–8, 2007, pp. 1–5.
[21] C.-M. Su, J.-W. Chou, C.-W. Yi, Y.-C. Tseng, and C.-H. Tsai, "Sensor-aided personal navigation systems for handheld devices," in Proc. 39th Int. Conf. Parallel Process. Workshops., Washington, DC, USA: IEEE Computer Society, 2010, pp. 533–541.
[22] T. Ishikawa, M. Kourogi, T. Okuma, and T. Kurata, "Economic and synergistic pedestrian tracking system for indoor environments," in *Proc. Int. Conf. Soft Comput. Pattern Recog.*, 2009, pp. 522–527.
[23] C. Ascher, C. Kessler, M. Wankerl, and G. F. Trommer, "Dual IMU Indoor Navigation with particle filter based map-matching on a smartphone," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, 2010, pp. 1–5.
[24] Z. Hao and J. Malik, "Learning a discriminative classifier using shape context distances," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog.*, 2003, vol. 1, pp. I-242–I-247.
[25] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.
[26] J. A. Saghri and H. Freeman, "Analysis of the precision of generalized chain codes for the representation of planar curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 5, pp. 533–539, Sep. 1981.
[27] N. Dunford and J. T. Schwartz, *Linear Operators, vol. I.* New York, NY, USA: Interscience Publishers, 1958.
[28] R. A. Johnson, *Modern Geometry: An Elementary Treatise on the Geometry of the Triangle and the Circle*. Boston, MA, USA: Houghton Mifflin, 1929, pp. 173–176, 249–250, 268–269.
[29] "Calculating the area and centroid of a polygon," (1988). [Online]. Available: http://paulbourke.net/geometry/polygonmesh/
[30] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor Localization Without the Pain," in *Proc. 16th ACM Int. Conf. Mobile Comput. Netw.*, Chicago, IL, USA, Sep. 2010, pp. 173–184.
[31] D. Gusenbauer, C. Isert, X. Kro, and J. Sche, "Self-contained indoor positioning on off-the-shelf mobile devices," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, 2010, pp. 1–9.
[32] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proc. 14th ACM Int. Conf. Ubiquitous Comput.*, Sep. 2012, pp. 421–430.
[33] P. Aggarwal, D. Thomas, L. Ojeda, and J. Borenstein, "Map matching and heuristic elimination of gyro drift for personal navigation systems in GPSdenied conditions," *Measurement Sci. Technol.*, vol. 22, p. 025205, 2011.
[34] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 197–210.
[35] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, New York, NY, USA, 2008, pp. 323–336.
[36] K. G. Raghu, P. Nam, A. Hossein, N. Saurabh, and F. A. Tarek, "GreenGPS: a participatory sensing fuel-efficient maps application," presented at the 8th Int. Conf. Mobile Syst., Appl. Services, San Francisco, CA, USA, 2010.
[37] M. Alzantot and M. Youssef, "UPTIME: Ubiquitous pedestrian tracking using mobile phones," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 1–4, 2012, pp. 3204–3209.
[38] R. Liu, Z. Duan, J. Zhou, and M. Liu, "Identification of individual walking patterns using gait acceleration," in *Proc. The 1st Int. Conf. Bioinformatics Biomed. Eng.*, Jul. 6–8, 2007, p. 543, 546.
[39] K. Liu, X. Liu, and X. Li, "Guoguo: Enabling fine-grained indoor localization via smartphone," presented at the 11th Int. Conf. Mobile Syst., Appl. Services, Taipei, Taiwan, Jun. 25–28, 2013.
[40] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 269–280.
[41] E. G. Lutz, *Animated Cartoons*, Charles Scribner & Sons: New York, NY, USA, reprinted by Applewood Books, Bedford, MA, 1920, 105 pp., ISBN 1-55709-474-8.
[42] G. Thüer and T. Verwimp, "Step detection algorithms for accelerometers," E-Lab Master's Thesis, from the Artesis University College of Antwerp, Antwerp, Belgium, 2008–2009.
[43] Footsteps, (2013). [Online]. Available: https://itunes.apple.com/us/app/footsteps-pedometer-free/id364911801.

Kun-Chan Lan received the Master's degree in computer science from State University of New York, Stony Brook, NY, USA, in 1997. He received the Ph.D. degree in computer science from University of Southern California, Los Angeles, CA, USA, in 2004.

From 1998 to 2004, he joined ISI's Computer Networks Division as a Graduate Research Assistant. During this period, he maintained and contributed codes to the popular network simulation software ns-2. From 2004 to 2007, he joined the Network and Pervasive Computing Program at NICTA (National ICT Australia) in Sydney, Australia, as a Researcher. He is currently an Associate Professor with the Department of Computer Science and Information Engineering of National Cheng Kung University, Tainan, Taiwan. His research interests include intelligent transport system (ITS), sensor network, and pervasive healthcare.

Wen-Yuah Shih received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2009. He received the M.S. degree in computer science and information engineering from National Cheng Kung University, Tainan, Taiwan, in 2012. He is currently working toward the Ph.D. degree at the Department of Computer Science of National Chiao Tung University, Hsinchu, Taiwan.

His areas of research include wireless sensor network, and ad hoc network.