Expert Systems with Applications 41 (2014) 2443–2456

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An intelligent driver location system for smart parking

Kun-Chan Lan [a], Wen-Yuah Shih [b,*]

[a] Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, University Road, Tainan City 701, Taiwan, ROC
[b] Department of Computer Science, National Chiao Tung University, No. 1001, University Road, Hsinchu 30010, Taiwan, ROC

ARTICLE INFO

Keywords:
Pedestrian dead reckoning
Simple harmonic motion
ZUPT
Map matching
Floor plan
Smart parking

ABSTRACT

It is often frustrating for drivers to find parking spaces, and parking itself is costly in almost every major city in the world. Here we propose a crowdsourcing solution by exploiting sensors in smart-phones to collect real-time parking availability information. We design a phone-based system to track a driver's trajectory to detect when they are about to leave their parking spot. We focus on the efficiency and accuracy of using a phone to monitor the driver's walking trajectory, applying a waist-mounted PDR method that can measure the driver's moving distance with a high accuracy. In addition, we design a map matching algorithm to calibrate the direction errors when the driver is in an indoor environment, using widely-available building floor plans. The results of our experiment show that we can achieve about 98% accuracy in estimating the user's walking distance, with an overall location error of about 0.48 m.

1. Introduction

Searching for street parking in crowded urban areas creates many problems and frustrations for drivers. It has been shown that over 40% of the total traffic volume in urban areas is composed of vehicles cruising for parking (Shoup, 2006). A long queue of cruising vehicles can cause serious congestion with the blocking of only a few streets. In addition, low speed cruising can produce significant amounts of automobile emissions (Arnott & Inci, 2006), increasing air pollution. A prior study (Mathur et al., 2010) found that in one area of Los Angeles vehicles searching for parking produced 730 tons of carbon dioxide, and burned 47,000 gallons of gasoline (Ayala, Wolfson, Xu, Lin, & Dasgupta, 2011) over one year.

In this work, we propose a solution that utilizes the sensors (such as a GPS, accelerometer, gyroscope, and digital compass) in a smart-phone to detect the driver's parking/un-parking activities. Such information can then be broadcast (e.g. through the Internet) to people who are trying to find a parking space. To detect parking, a prior work has shown that it is possible to detect the driver's transportation mode (e.g., driving, stationary, and walking) (Stenneth, Wolfson, Yu, & Xu, 2011) using the sensors in a smart-phone. For example, if we detect a transition pattern like driving → stationary → walking, we may conclude the car has been parked at the stationary point. In this paper, we focus on how to detect the unparking activity by tracking the walking trajectory of the driver using the smart-phone's sensors. The idea is a simple one. If the phone detects that the driver is approaching the place where they parked their car, it is likely the driver is about to leave the area and the parking space will become available very soon.

In this paper, we consider a social network formed by drivers, similar to the Crowdpark (Yan et al., 2011) platform. In our architecture, a driver who is currently parked can provide advance notification about when they plan to leave, and this information may be sold to another driver who is willing to pay (via a virtual currency (Lan & Wang, 2013), such as BitCoin) to reserve the parking spot. The buyer arrives at the reserved parking spot close to the leaving time of the seller, and can occupy the spot when the seller leaves. Since the drivers transact only parking availability information, our paradigm presents a loose reservation model for the parking spot, and the buyer is charged only when they successfully park their car in the focal location. Our system provides an incentive for sellers to contribute their leaving information, and also encourages the buyers to re-sell the parking spaces when they leave. Each participant in this system is assigned a random unique ID when joining the social network, and thus their real identity will not be revealed to the other users.

While crowdsourcing-based approaches to parking are not new (Yan et al., 2011), but, as far as we know, all of the prior works in this area (such as Google OpenSpot (OpenSpot) require the drivers to manually report when they leave their parking spots. In contrast, we utilize the sensors in the user's smart-phone to automatically infer the parking space's availability in advance by tracking the trajectory of the driver. How to accurately and efficiently monitor the walking trajectory of the driver is thus the key issue in our method, and the focus of this paper.

Tracking the walking trajectories of people in an environment has long been considered an important component of ubiquitous networking. Generally, in an outdoor environment, the driver's

* Corresponding author. Tel.: +886 35712121.
E-mail addresses: klan@csie.ncku.edu.tw (K.-C. Lan), Todd629.cs01g@nctu.edu.tw (W.-Y. Shih).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.09.044

walking trajectory can be obtained through GPS data, although it is more challenging to localize the driver in an indoor environment. In this paper, we consider the use of a personal dead reckoning or pedestrian dead reckoning (PDR) system to localize the driver, particularly when they are in a building. The PDR technique only requires a couple of inertial sensors to be put on the user, so that it can be used in any building without pre-installing beacon nodes or pre-building RF maps/propagation models based on surveys of the environment. These inertial sensors (such as an accelerometer, gyroscope, or digital compass, which smart-phones usually have) are used to measure step length and heading direction.

Generally, depending on where the sensors are placed, previous studies classified PDR systems into two types: foot- and waist-mounted. The foot-mounted method (Alvarez, Gonzalez, Alvarez, Lopez, & Rodriguez-Uria, 2007; Dippold, 2006; Eric, 2005; Feliz, Zalama, & García-Bermejo, 2009; Jimenez, Seco, Prieto, & Guevara, 2009; Ojeda & Borenstein, 2006; Ojeda & Borenstein, 2007a; Ojeda & Borenstein, 2007b; Oliver & Robert, 2008; Sagawa, Inooka, & Satoh, 2000; Xiaoping, Bachmann, Moore, & Calusdian, 2007) uses a double integral on horizontal acceleration to estimate distance, and a gyroscope or compass to measure the heading direction. On the other hand, the waist-mounted (Alvarez, Gonzalez, Lopez, & Alvarez, 2006; Lei et al., 2005; Shin, Park, Hong, & Lee, 2005; Weinberg, 2002) 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 (Shin, Park, Kim, Hong, & Lee, 2007). But when the pedestrian's walking pattern is different from the predefined step length model, the accuracy in distance estimation could be adversely affected. Finally, although a waist-mounted method is generally more feasible to be implemented on a hand-held device, its accuracy in estimating the step length is typically worse than that of a foot-mounted method which, in contrast, performs poorly with regard to obtaining an accurate orientation (Stirling, Collin, Fyfe, & Lachapelle, 2003).

Sensor drift (Titterton & Weston, 1997) is a well-known problem in PDR systems. Given that the hardware used in such systems is not perfect, the inertial sensors constantly have some small errors when estimating the distance and direction, and signal noise

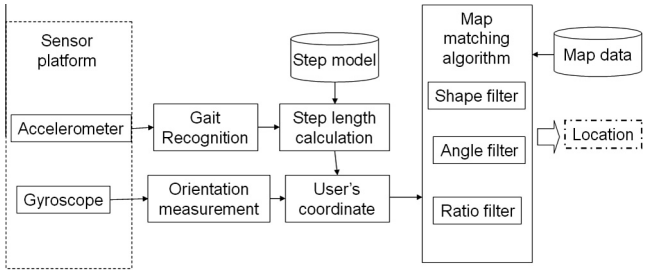

Fig. 2. The entire system architecture.

(such as the vibrations of the user's body) can further exacerbate this problem (Feliz et al., 2009; Sagawa et al., 2000). In an outdoor environment, sensor drift can be mitigated through the use of GPS measurements (Godha, Lachapelle, & Cannon, 2006; Ladetto & Merminod, 2002), while in an indoor environment some PDR systems use a map matching mechanism to calibrate these errors (Ascher, Kessler, Wankerl, & Trommer, 2010; Gusenbauer, Isert, & Krösche, 2010; Ishikawa, Kourogi, Okuma, & Kurata, 2009), and these can be categorized into two types. One tries to matches the user trajectory to the closest junction and road on the map (Gusenbauer et al., 2010), while the other one utilizes the map to filter out positions where the user is unlikely to move (e.g., walls, obstacles, and so on) (Ascher et al., 2010; Ishikawa, 2009). Both techniques require the use of a detailed scaled map of the building, as shown in Fig. 1(a), although in practice this is usually difficult to obtain.

In this paper, we consider a scenario in which the user has a smart-phone and can access the floor plan of the building, like the one shown in Fig. 1(b), as these are widely available. The system utilizes the sensors on the smart-phone to compute the user's moving distance and direction. Combining the walking trajectory with the map allows the system to estimate the driver's current position in a building. The system architecture is shown in Fig. 2. Our system can also be implemented as an add-onto a traditional outdoor navigation app (e.g., Google Latitude, although this service has now been discontinued) so that the starting position of the user in the building can be estimated using the last-recorded GPS position.

We make the following assumptions in this paper.

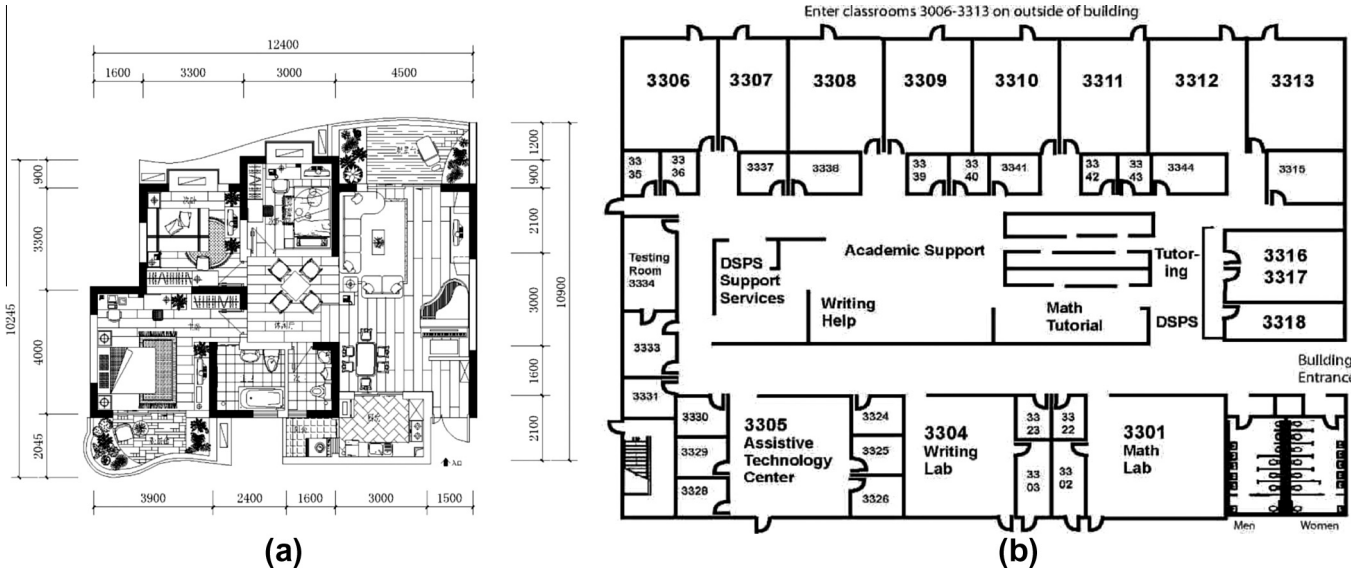

Fig. 1. (a) A detailed map and (b) a floor plan.

1. The floor plan can be characterized using a link-node model (Gillieron & Merminod, 2003), in which pathways are the links and the intersections of pathways are the nodes. Note that the limitation of a link-node model is that it does not consider open indoor spaces (e.g., a big hotel lobby), and we leave this for future work.
2. The phone is placed inside the pocket of the user, which is often the case in real life (however, some prior works, such as (Mohan, Padmanabhan, & Ramjee2008), discussed the situation of when the phone is held in the hand).
3. We assume the initial heading direction is known. This can be achieved by having the user point the phone in the direction they are heading before putting the phone in their pocket.
4. During walking, the position of the phone is relatively stable in relation to the leg movements.
5. In this work, we do not consider the multi-floor building scenario, since most of smart-phones do not have a barometer to detect the change to a different floor, and we leave this as a further direction for future work.

The contribution of this paper is threefold. First, unlike previous crowdsourcing-based methods that required the drivers to manually report when they leave their parking spots, we utilize the phone sensors to automatically infer the parking space availability and notify the other drivers in advance by tracking the trajectory of the driver. Second, we implement a waist-mounted-based PDR method on the smart-phone to accurately estimate the driver's moving distance with no need for a training phase. The accuracy of our distance estimation is about 98%. Finally, when the driver is indoors, based on the geometric similarity between the driver's trajectory and the floor map, we design a map-matching algorithm to calibrate sensor errors using building floor plans, which are widely available. We find that the location error is about 0.48 m in our test scenarios.

The rest of this paper is structured as follows: In Section 2, we describe the related work. We discuss the details of our step-length estimation and map-matching algorithms in Sections 3 and 4, respectively. The results of our experiment are shown in Section 5. Finally, we conclude this paper in Section 6.

2. Related work

Our work is built on some previous studies of parking guidance, PDR, step length estimation, and map matching, and these are outlined below.

2.1. Parking-guidance

Online real-time parking information systems have recently attracted significant interest in both industry and academia. They can generally be classified into infrastructure-based and crowdsourcing-based approaches. The infrastructure-based methods, such as SFpark and ParkNet, require the installation of occupancy sensors and wireless transceivers on the parking spot and/or vehicle. For example, ParkNet installs ultra-sonic sensors on vehicles to detect parking availability when vehicles drive by. However, these approaches are generally costly and not scalable. On the other hand, a number of mobile crowdsourcing applications have been developed that allow drivers to share empty parking spot information, such as OpenSpot, Roadify, Primospot, and SpotScout. These work by having the drivers report when they are leaving their parking spots. That data gets fed into the applications, and then drivers can see where people have recently left an open parking spot. However, such data could have a *very* short lifespan, since empty spots in crowded areas are quickly used. In other words, by the time when the driver figures out how to get to an open spot, it is likely to have already been taken. In contrast to these prior works, we utilize the sensors on the phone to *automatically* infer parking space availability by tracking the trajectory of the driver, which requires no additional infrastructure or man-power support, and is more practical for deployment in the real world. Furthermore, our system provides an incentive for the driver to offer parking availability information in advance to others, by selling it, so that the potential buyer can arrive at the parking spot close to the leaving time of the seller.

To illustrate the usefulness of such advance parking availability notification, we performed the following experiments in an urban area. We had two cars (A and B) travel around in the area. We performed two sets of experiments using our system and OpenSpot. Each set of experiments was executed for three days. The idea here is to have car A update its parking information online, and then observe how many times car B can successfully occupy the parking spot that car A has previously taken. We found the success rate for car B to arrive at car A's spot in time was 87% using our application, and only 59% using OpenSpot.

2.2. PDR system

Some prior studies use a PDR system to estimate the trajectory of a user by placing some sensors on the body. Inertial sensors, such as accelerometers, gyroscopes, and compasses, are commonly used in such systems. Some PDR systems also include GPS sensors (Godha et al., 2006; Ladetto & Merminod, 2002), and use these to calibrate the PDR drift as long as a GPS signal is available. When the GPS signal is obstructed, the system can then be changed to the PDR mode and continue to record the trajectory. Our study is based on the PDR system, which has the advantage of avoiding the deployment overhead of the signal-based methods. On the other hand, the performance of our system can be enhanced by a signal-based system if available. For example, one limitation of our map-matching approach is its need to collect "enough" trajectory data before it can uniquely identify the user's position on the map. When there is not enough trajectory information, we make use of the known locations of existing WiFi base stations in the building to help estimate the user's location (Ji, Biaz, Pandey, & Agrawal, 2006; Krishna, Anand Padmanabha, & Venkata, 2010; Lim, Kung, Hou, & Luo, 2006).

2.3. Step length estimation

A PDR system can be classified into two types depending on where the sensor is mounted: foot-mounted (Alvarez et al., 2007; Dippold, 2006; Eric, 2005; Feliz et al., 2009; Jimenez et al., 2009; Ojeda & Borenstein, 2006; Ojeda & Borenstein, 2007a; Ojeda & Borenstein, 2007b; Oliver & Robert, 2008; Sagawa et al., 2000; Xiaoping et al., 2007) and waist-mounted (Alvarez et al., 2006; Lei et al., 2005; Shin et al., 2005; Weinberg, 2002). To calculate the step length, the foot-mounted methods typically perform a double integral on the horizontal acceleration. However, without further calibration, the problem of sensor drift (Titterton & Weston, 1997) could introduce serious inaccuracies when estimating the step length. One way to calibrate the sensor drift error is called zero velocity update (ZUPT). When the swinging-foot touches the ground, the angular velocity of this foot will be close to zero, which can be used to reset the system to avoid sensor drift errors accumulating into the length estimation of the next step.

On the other hand, for a waist-mounted PDR system, ZUPT is not directly applicable, since one will not be able to find zero velocity in the horizontal direction. Some waist-mounted PDR systems use a constant step length (Jorgensen, 2008; Martin & Michael, 2009; Schneider, Crouter, Lukajic, & Bassett, 2003; Shih,

2010) while the others (Li et al., 2012; Shin et al., 2005) use a trace-driven approach, by first collecting empirical data from multiple users and then using linear regression to find the relation between step length, walking frequency, and the variance of acceleration. The limitation of this approach is that one might need to collect new training data for a new user. Weinberg (2002) observed that the upper body moves vertically when walking, and suggested that one can estimate the step length as follows

$$StepLength = 2 \times heightchange/\alpha$$

where α is the swinging angle of the leg from the body, and the *heightchange* can be estimated based on the vertical acceleration. However, he did not discuss how to measure α. Our idea is similar to Weinberg's, but we estimate the step length based on the height change and length of the leg using the Pythagorean Theorem. In addition, we use the concept of Simple Harmonic Motion (SHM) to find the zero velocity in the vertical direction, and apply ZUPT to avoid the accumulation of sensor drift errors on the vertical-axis.

2.4. Map-matching algorithm

In some prior PDR systems, a map-matching mechanism is used to match the user trajectory onto the map (Ascher et al., 2010; Gusenbauer et al., 2010; Ishikawa, 2009) in order to calibrate the sensor errors. There are two ways this is achieved. The first tries to match the user trajectory to the closest junction and road on the map (Gusenbauer et al., 2010), while the other utilizes the map information to filter out positions where the user is unlikely to walk (e.g., walls and obstacles) (Ascher et al., 2010; Ishikawa, 2009). However, both techniques require the use of a detailed scaled map of the building (i.e., with detailed distance information for each route on the map), which is usually not easy to obtain. In our work, we utilize the more widely available building floor plans instead of detailed scaled maps. Based on the geometric similarity between the trajectory data and the map, we propose a new map-matching method that uses the floor plan to locate the user. As shown later in Section 5, our approach has similar performance to those methods that rely on detailed scaled map information.

3. Implement a waist-mounted PDR method using a smart-phone

Weinberg (2002) proposed that one can estimate the step length using the height change of the waist and the *swinging angle of the leg* during walking (Weinberg, 2002). However, he did not discuss how to measure this angle, and we found that, in practice, it is difficult to measure such a small angle during walking. Based on Pythagorean Theorem, in our prior work Lan and Shih, (2012) we proposed 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. Considering the triangle in Fig. 3, formed by two feet of a person and their 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 first need to 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 of this 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 above-mentioned waist-mounted method for a smart-phone. When implementing a PDR system on a smart-phone, two cases can be considered. The first is when the user holds the phone (e.g., when talking on the phone), and the other is when the phone is put in a pocket or a bag. Prior research (Su, Chou, Yi, Tseng, & Tsai, 2010) has shown the feasibility of using a waist-mounted method for the first case. Therefore, in this section we focus on how to extend the results of a waist-mounted method for the second case. To implement a waist-mounted method, two issues need to be considered: the orientation and placement of the phone. The orientation of the phone may change from time to time when it is put in a pocket or bag during walking, and this affects the influence of gravity on the three axes of the accelerometer, and results in different readings from the sensor. To resolve this problem, we adopt a method similar to that in an earlier work, Su et al. (2010), which used a gyroscope to record changes in orientation and to calibrate the system, as shown in Fig. 4. Next, in a waist-mounted method, the vertical movement displacement of the phone is the key parameter to estimate the step length. When the phone is positioned above the waist-line, its vertical displacement during walking will be the same as when it is mounted on the waist. On the other hand, when the phone is placed below the waist-line (e.g., in a pants pocket), we can estimate the vertical displacement of the waist as follows. Assuming the leg length (L) and the pocket position from the ground (L') are already known, we can find two similar triangles ΔABC and ΔPQR, as shown in Fig. 5. 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 the walking). Then, using triangle similarity, we can find $\overline{BC} = D/2 = (L \times \overline{QR})/L'$. h' can be measured in a similar way as to measure h .

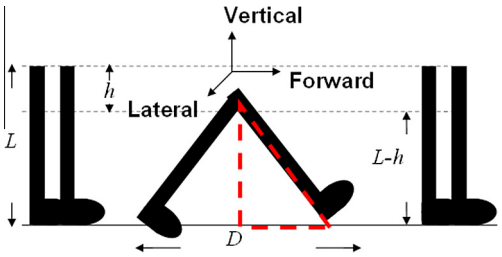

Fig. 3. Walking diagram.

To use the smart-phone to estimate the walking distance, the following practical issues need to be considered:

1. *How to remove signal noise, such as vibrations from the body?*
2. *How to prevent sensor drift errors accumulating from one step to the next?*
3. *How to deal with when the user moves at irregular speeds?*

Our system architecture is shown in Fig. 6. 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) to reset the vertical

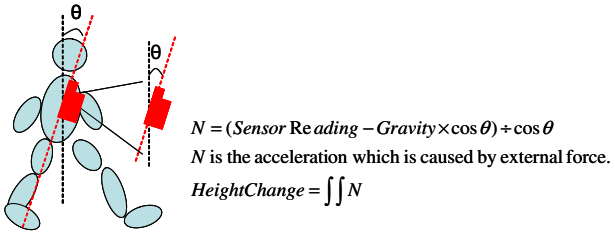

Fig. 4. The orientation of the device.

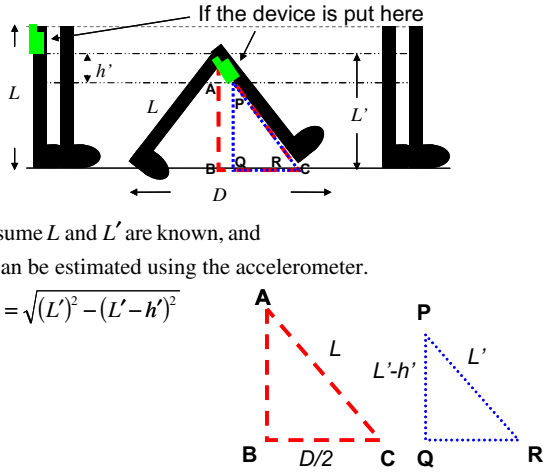

Fig. 5. When the device is mounted on the lower body.

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 foot length.

3.1. Noise filtering

During walking, some unexpected and unpredictable body vibrations might cause some higher-frequency noise in the sensor readings. In this section, we discuss how to filter such noise.

Intuitively, one can use a low pass filter and preset a cut-off frequency to filter noise. Some prior work has shown that the frequency of human muscle movement is lower than 16 Hz (Gerald & Andreas, 2008), and the human step frequency is never higher than 3 Hz (Lei et al., 2005). Therefore, some step recognition systems use 3 Hz as their cut-off frequency to filter noise signal (Martin & Michael, 2009). Initially, we also used 3 Hz, but then found our results became worse. This shows that, while it is good enough to detect a new step event, a 3 Hz threshold is too low for our purposes, and will remove data which is not noise. Some prior research (Chen & Bassett, 2005) analyzed the acceleration of the waist during walking, and found the maximum acceleration is 8 Hz. We thus set our cut-off frequency at 8 Hz for filtering the noise, and this threshold worked well under repeated experiments with different subjects.

3.2. 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, as shown in Fig. 7.1. 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. 7.2, the vertical acceleration of a heel-touching-ground event is the local minimum within a step. In addition, previous studies (Gafurov, Snekkenes, & Bours, 2007; Shih, 2010) showed that human walking frequency is never over 3 Hz. Therefore, the duration between two consecutive

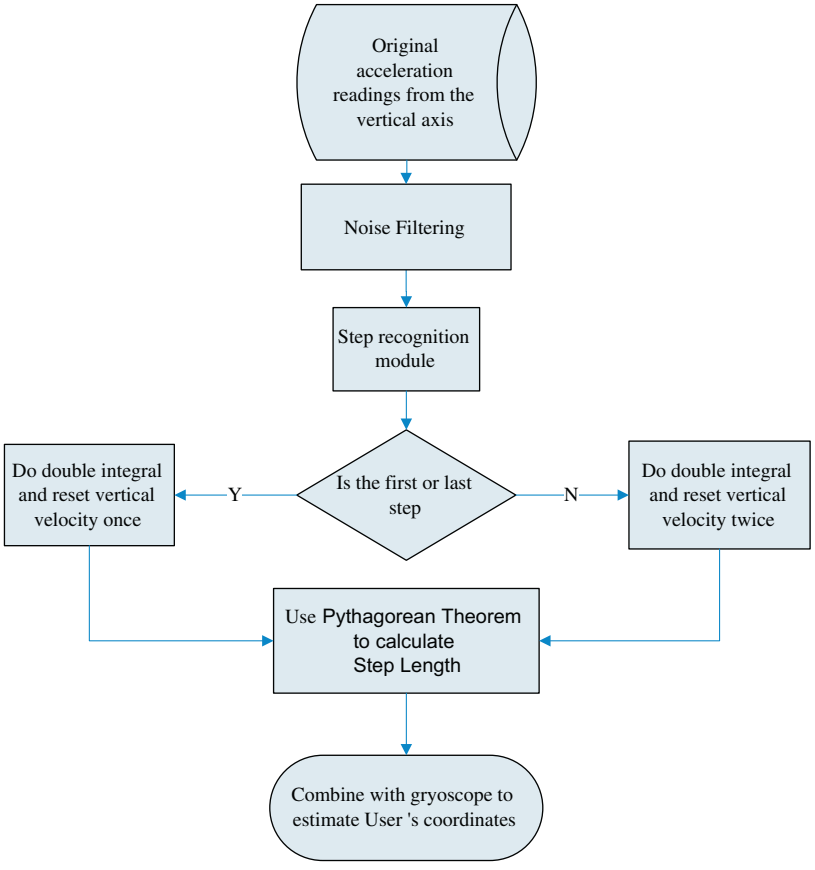

Fig. 6. Flow chart of our PDR system.

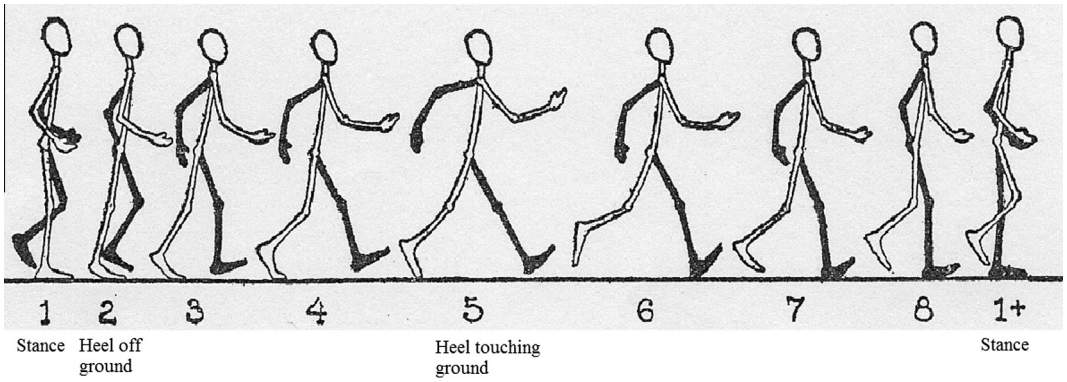

Fig. 7.1. Walking diagram (modified from Walking diagram).

heel-touching-ground events must be over 0.33 s. Based on the above, we use a 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.

3.3. Step length estimator

1) *Our step model*

To measure the walking distance from point A to point B, we sum up the step length of all steps. We model a complete step as from one heel-touching-ground-event to the next. Therefore, we consider the first step (i.e., from the stance event to the heel-touching-ground event, as shown in Fig. 7.1) and the last step (i.e., from the heel-touching-ground event to the stance event) 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 height change 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 drifts errors could accumulate from one step to the next. To avoid this, we previously proposed a zero velocity update method (Lan & Shih, 2012) to calibrate the sensor data: if we mount a sensor on the waist of a pedestrian, then while they are walking the trajectory of the sensor can be approximated by Simple Harmonic Motion (SHM). 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 maximum, i.e., points A and B in Fig. 7.2) are identified, we can then reset the vertical velocity to zero before doing the double integral for calculating the height change of the user's waist. The height change can then be used to estimate the step distance based on the Pythagorean Theorem.

Specifically, there are three major events during one step, namely heel-touching-ground, stance, and heel-off-ground. As shown in Fig. 7.2, the lowest valley (point A) of the wave indicates when the heel is touching the ground (corresponding to the fifth posture in Fig. 7.1), the first peak occurs (point B) when the walker is in the stance state (corresponding to the first posture in Fig. 7.1). Following the stance event, the body starts to lean forward and the foot is now on its toes, which will give a force to push the body up, so that the vertical acceleration will change to the opposite direction and cause the second valley in Fig. 7.2 (i.e., point C, corresponding to the second posture in Fig. 7.1), due to the law of inertia.

As shown in Fig. 7.1, 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. We performed an experiment to observe the effect of

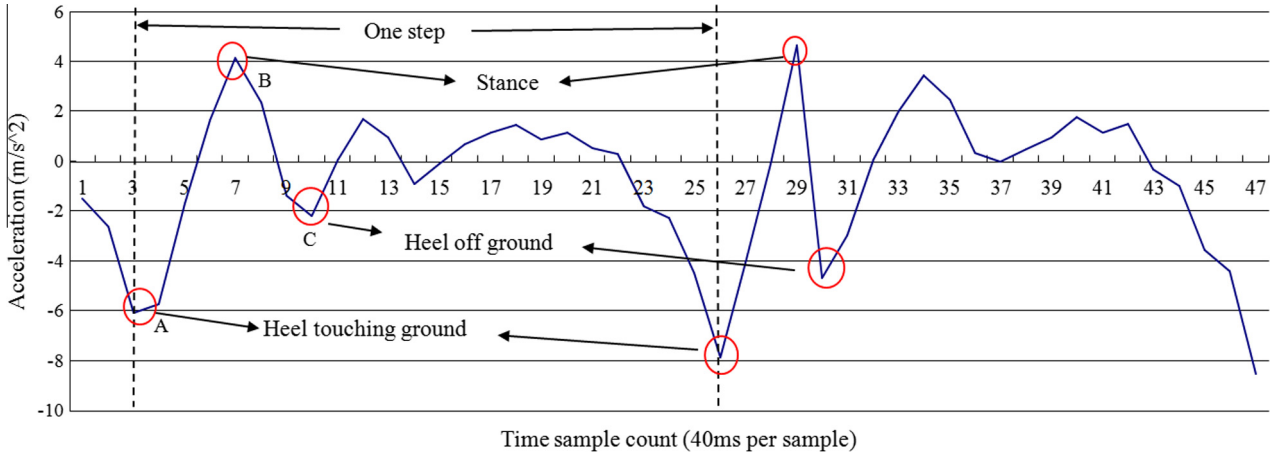

Fig. 7.2. The vertical acceleration of walking.

doing such a zero velocity update (ZUPT). As shown in Fig. 8, implementing ZUPT can indeed avoid the accumulation of sensor drift errors.

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 on the same spot*

As shown in Fig. 7.2, when one walks at a normal speed, the stance event generally occurs as the first peak after the heel-touching-event. However, during high speed movement, 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. Previous studies in kinesiology (Chintalapudi, Iyer, & Padmanabhan, 2010; Gusenbauer et al.,2010) 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 on the same spot, the acceleration data should not be considered in the calculation of step length. In addition, when one is walking on the same spot, the height of the waist does not usually change significantly. Therefore, we use a simple 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).

$$verticalAcc = 0,$$
$$if\ |lateralAcc| < \varepsilon\ \&\ |horizontalAcc| < \varepsilon\ \&\ |verticalAcc| < \varepsilon$$

4. 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*. In the previous section, 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 moving is usually obtained from a gyroscope sensor. 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 trajectory might be biased by the error in its initial direction, and appear as a rotated version of the true path, as in the example shown in Fig. 11. When the error of the gyroscope is significant, and if left uncorrected, it can make the entire PDR system unusable. 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 (Aggarwal, Thomas, Ojeda, & Borenstein, 2011). In this study we propose a new map-matching method utilizing widely available building floor plans (instead of a detailed scaled map, as in some earlier works (Ascher et al., 2010; Ishikawa, 2009) to calibrate the 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 starting the map matching, we adopt an approach similar to that in a prior work (Gillieron & Merminod, 2003) by first converting the floor plan into a link-node model. The turning angels of the corners and comparative ratios of the lengths between any two corridors are then estimated, as shown in 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 travelled. Here we consider the map and the trajectory as two independent graphs, say M and T. We list all the sub-graphs of M and compare T with all these to find the most similar one. We define the "similarity" between two graphs by comparing their shapes, vertex angles (i.e., the angle between two connected edges) and relative edge lengths (normalized). 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 they move between corners using the dead-reckoning data from the accelerometer, as previously discussed in Section 3. Note that, 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 the existing WiFi-signal-based landmarks (e.g., a corridor-corner may overhear a unique set of WiFi APs, but the set may change at short distances away from that spot, and some dead spots inside a building may not overhear any WiFi signals, which by itself is a signature). When a WiFi AP signal is 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 those routes that do not contain it. This approach is similar to the method used by a recent study (Wang et al., 2012). The map-matching process is performed every time the system detects that the user makes a turn.

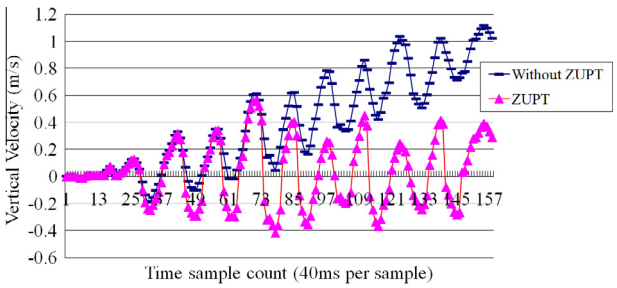

Fig. 8. The velocity with ZUPT versus without ZUPT.

4.1. Turn detection

The first step in our map matching process is to find out all the turnings in the trajectory data and use these as a signature to match all possible routes on the map. However, given that the gyroscope can only provide the relative angular displacement (RAD), we use a sliding-window-based algorithm to infer the user's possible turnings from the data. To determine if a person is making a turn, we compare the standard deviation of the window with a threshold which is estimated during the period when the user is walking straight. 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 as

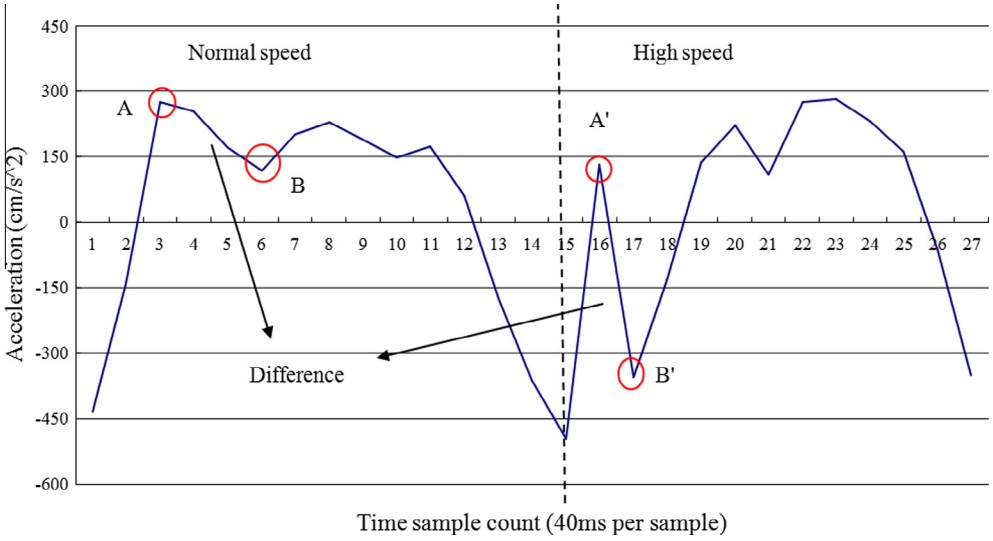

Fig. 9. The difference between normal and high speeds.

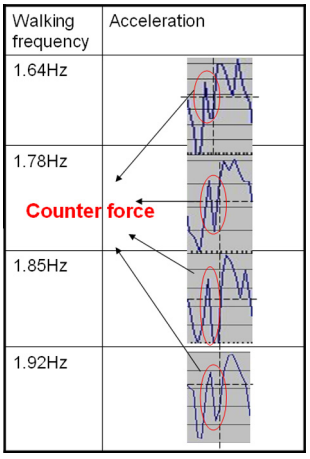

Fig. 10. The acceleration at different high speeds.

$CT = AT - BT$ 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 the 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. In this study we thus utilize the floor map information to resolve the issue of fake turnings, as follows. The assumption we make here is 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.

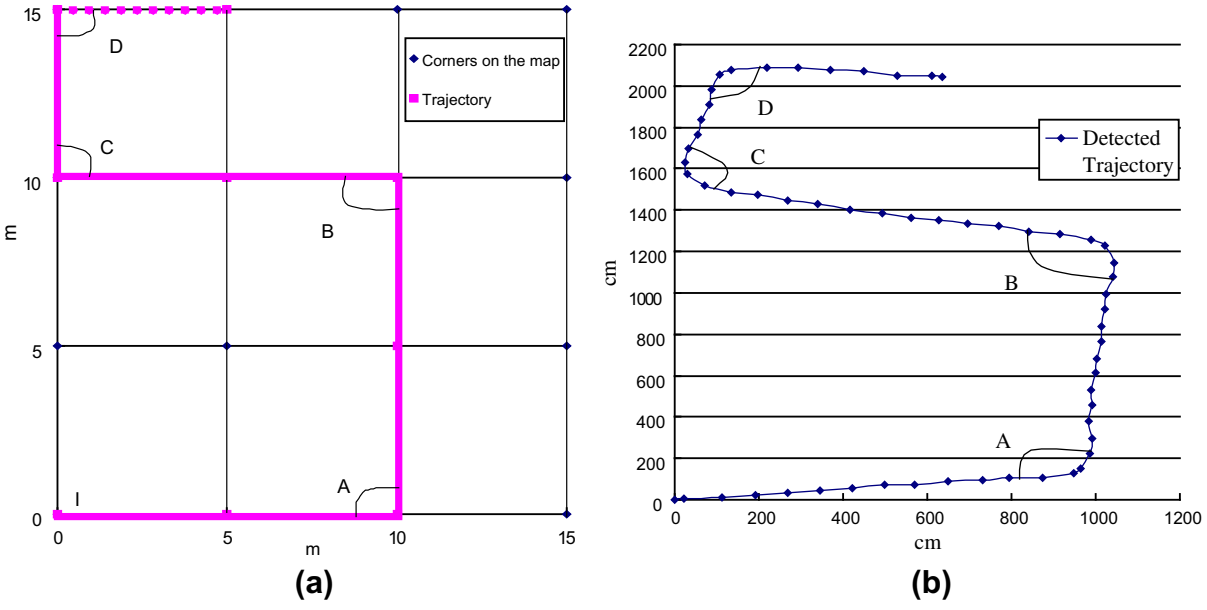

Fig. 11. (a) Link-model of the map and the ground truth, (b) the estimated trajectory from the sensor data.

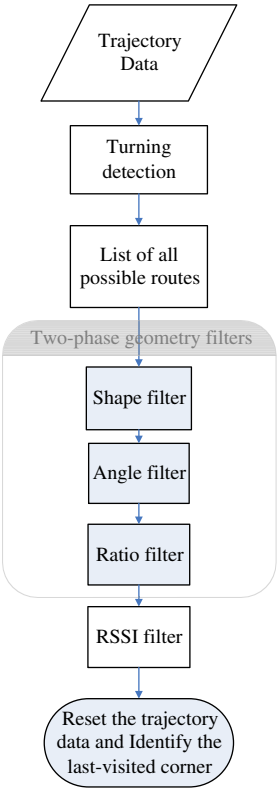

Fig. 12. The flow chart of our map-matching algorithm.

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 routes (i.e. R_T) with all the 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 sub-graph of G_M, which is used to model all possible routes on a map. 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, $V_{Tj} \in V'$ and $|V_{Tj}| > 1$. 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 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 non-existing routes caused by fake turnings by using these filters for all possible (G_{Mi}, G_{Tj}) pairs.

4.2. Two-phase filtering mechanism

One possibility is that we might not have "*sufficient*" trajectory data (e.g., when there is no detected turning in the trajectory data), and that multiple candidate routes remain after the geometry-similarity filtering. Therefore, we also employ an RSSI-based filter by using the existing WiFi-signal-based landmarks, as described previously, 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 (Belongie, Malik, & Puzicha, 2002; Hao & Malik, 2003) to compare 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; Johnson, 1929) 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, as shown in Fig. 15. Finally, we compare the arrays generated based the two graphs to determine their similarity using Euclidean distance (*L-2* norm) Dunford & Schwartz, 1958. We set a threshold to judge the similarity between these two graphs. If the value is over the threshold, the system will consider these two graphs are different and remove them from the set of candidates.

1) *The angle filter*

We adopt the concept of a chain code (Saghri & Freeman, 1981) 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 into a one-dimensional expression. However, we cannot directly apply a full-fledged chain code to our system. 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. Given that what we

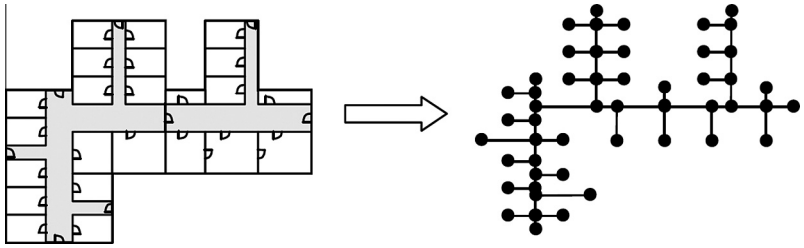

Fig. 13. The link-node model from a floor plan.

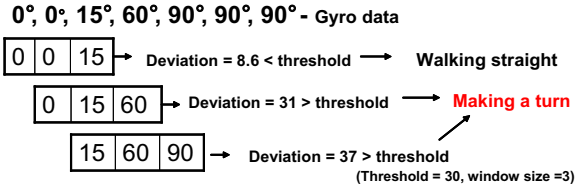

Fig. 14. An example of turning detection.

consider here is a real-time localization system and the computational capability of a smart-phone is limited, we cannot just naively use the chain code. Therefore, we adopt the concept of the chain code and implement it separately via two different filters: the angle filter and the edge filter. 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 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 G_T and G_{Mi}, and use a threshold to determine whether they are all close enough, as shown in 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.

2) *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.

5. Evaluation

In this section, we discuss the performance of our PDR-based localization system. We first discuss the results of our step length estimation method, and then show the performance of the above-mentioned map matching algorithm.

5.1. Experiment setup

To implement our algorithm, we use a variety of Android phones from HTC and Samsung. We perform two set of experiments on the smart-phones. One is placing the smart-phone in a shirt pocket and the other is putting it in a pants pocket. We use a laser distance meter to measure the actual travel distance of the user. In addition, to record the user's actual location, we pasted markers on the ground at precisely known locations. Each of these markers had a number on it, and the user recorded the numbers when they walked passed them.

5.2. Step length estimation

As discussed previously, we use a low-pass filter to filter out the noise in our step length estimation algorithm. We choose 8 Hz as our cut-off frequency for filtering the noise. To examine whether this frequency produces the best results, we perform a set of experiments using different frequencies for the low-pass filter, ranging from 3 Hz to 12 Hz, and compare their accuracies in estimating step length. As shown in Fig. 18, the use of 8 Hz as the cut-off frequency produces the most accurate results.

Some of the state-of-the-art pedometers on the market can also output walking distance. We compare our method with these pedometers, and find that they only achieve up to 90% accuracy, which is significantly lower than our results (98%). 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 (2002). In his work, he proposed an equation to estimate distance by observing the

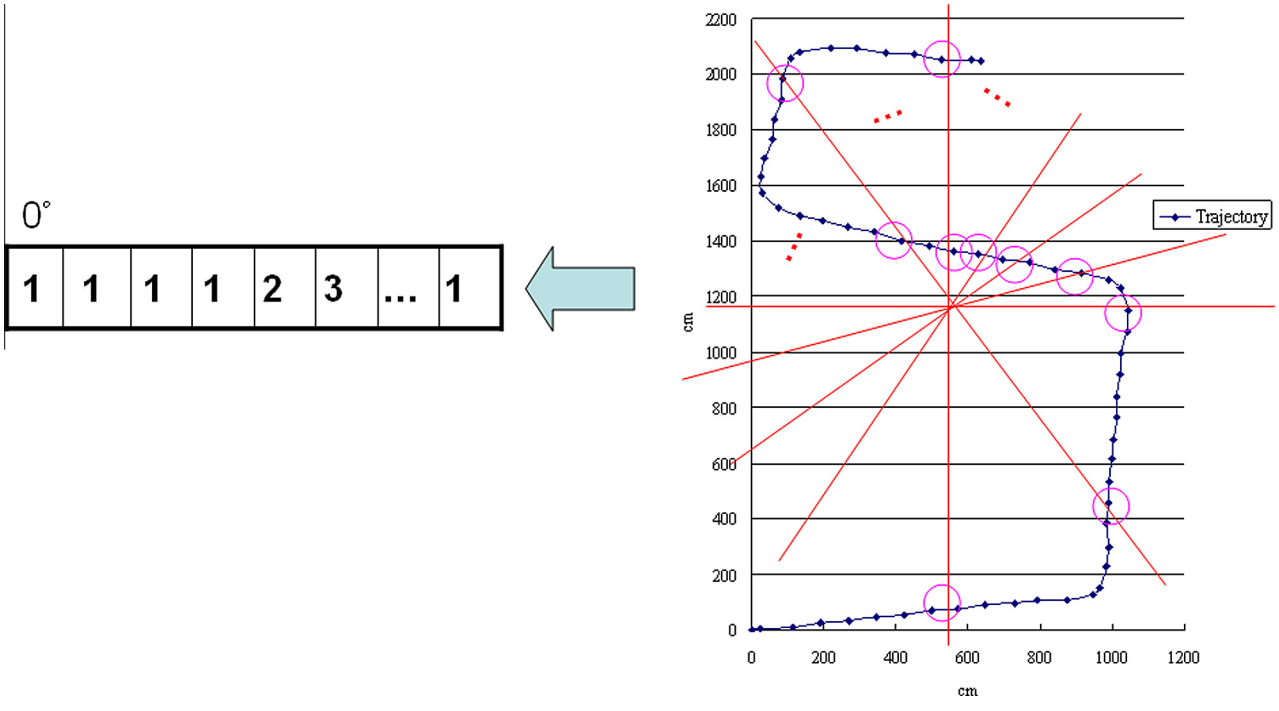

Fig. 15. Example of the shape filter.

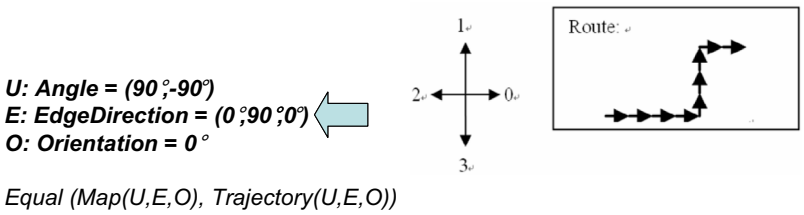

Fig. 16. Features of the angle filter.

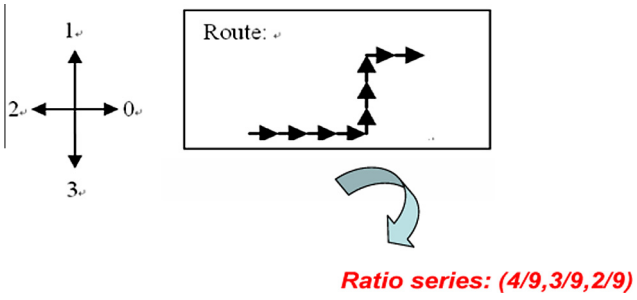

Fig. 17. The input to the edge filter.

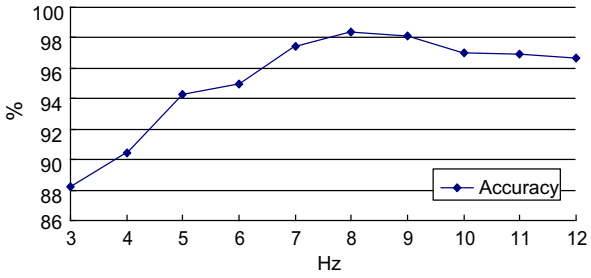

Fig. 18. The accuracy using different cut-off frequencies for the low pass filter.

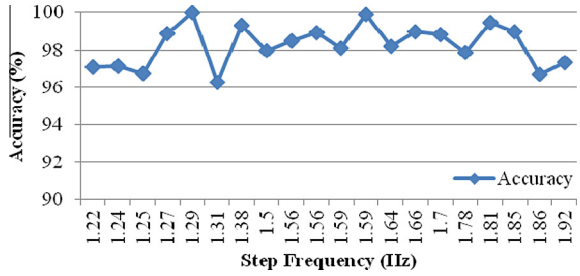

Fig. 19. Accuracy of estimated distance under different walking speeds.

vertical acceleration during walking. We implement Weinberg's method and find 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.

In our system, we use a low-pass filter (LPF) to filter out the noise and employ zero velocity update (ZUPT) to calibrate the drift

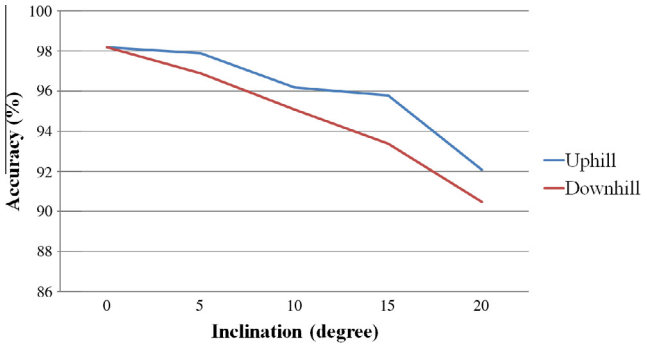

Fig. 20. Accuracy under different inclinations.

error of the accelerometer. To understand the effects of these mechanisms, we enable only one of them at a time. As shown in Table 1, ZUPT has a higher 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%.

Considering the variations that might exist in different individuals' walking gaits, we performed an experiment in which we asked the subject to walk with different speeds, as shown in Fig. 19. Here the step frequency is defined as the number of walking steps in one second. The average accuracy is about 98.26% and standard deviation is about 1.09%. In addition, we asked two other people (one male and one female) to perform the same experiment for the walking distance measurements. Again, each experiment was repeated for 10 times. The average accuracies and standard deviations are (98.42%, 0.98%) and (97.93%, 1.86%) respectively, which are similar to the results shown in Table 1.

Finally, to understand the effects of moving uphill of downhill on the accuracy of walking distance estimation, we carried out a set of experiments in which the user walked on with different slope inclinations of 0°, 5°, 10°, 15° and 20°. The results show that the step length is shorter when the slope becomes steeper. The accuracy of walking distance estimation also degrades as we increase the level of inclination from 0° to 20°, as shown in Fig. 20. In addition, the accuracy in the downhill direction is generally lower than that in the uphill direction.

5.3. Map matching

As discussed above, 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

Table 1
comparison with other waist mounted methods and the influence of the mechanism.

	Pedometer	Weinberg	Ours (A: LPF, B: ZUPT)			
Method	Constant step length × step number	$k \times n \times \sqrt[4]{Acc_{max} - Acc_{min}}$	Original	A	B	None
Accuracy (Estimation ground truth)	90.09%	96.78%	98.25%	84.1%	95.29%	75.53%
Standard deviation	1.83%	3.18%	1.29%	7.41%	2.22%	16.3%

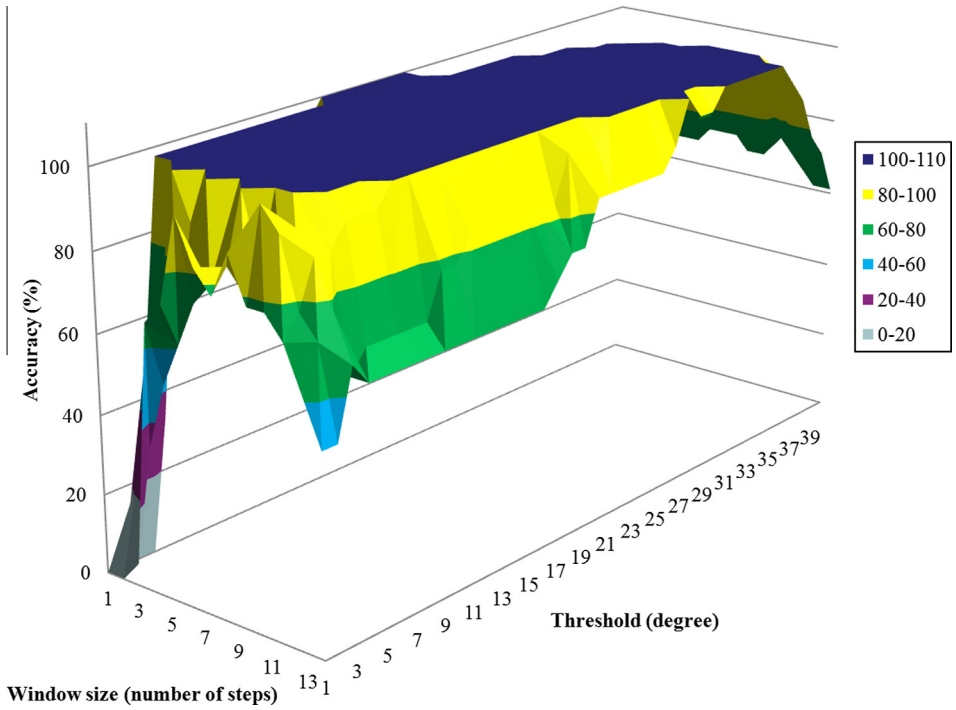

Fig. 21. The number of walking steps between two corners is larger than the window size.

deviation of the window with a threshold. Generally, when the chosen window size is too small, all the walking steps that happen during a turning might not be able to be included within a window. On the other hand, 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. As shown in Figs. 21 and 22, the accuracy of detecting turnings becomes lower when the chosen window size is too small or too big. Therefore, in our experiments we set the window size from the range defined below, in which D is the shortest distance between two adjacent corridors.

$$2 < Window\ Size < \frac{D}{Avg\ Step\ Length}$$

In addition, 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 10° from 0° to 180°, that passes through the centroid of the graph, and use the crossing points on the edges obtained in 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 will be obtained, since fewer crossing points will be generated for the comparison of shape similarity. As shown in Fig. 23, we start getting inaccurate results when the gap is larger than 15°.

Finally, to test the performance of our localization system, we choose a route which is about 40 m long and includes four corridors and corners, as shown in Fig. 24. We deliberately created some 'fake turnings' by having a walker walk straight down the route but wander about at certain spots (we put markers on these

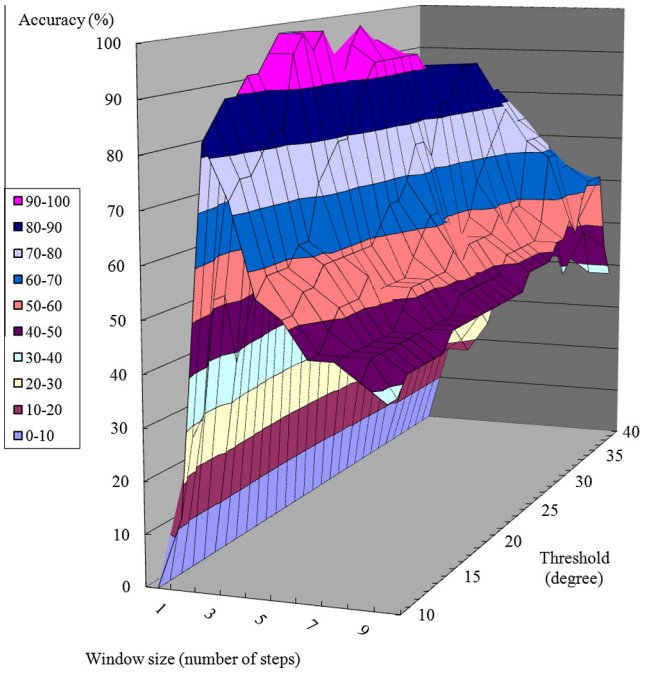

Fig. 22. The number of walking steps between two corners is smaller than the window size.

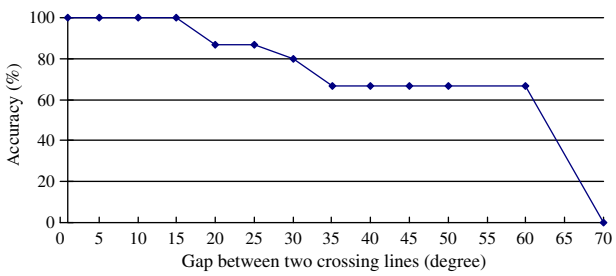

Fig. 23. The discrimination of different slopes.

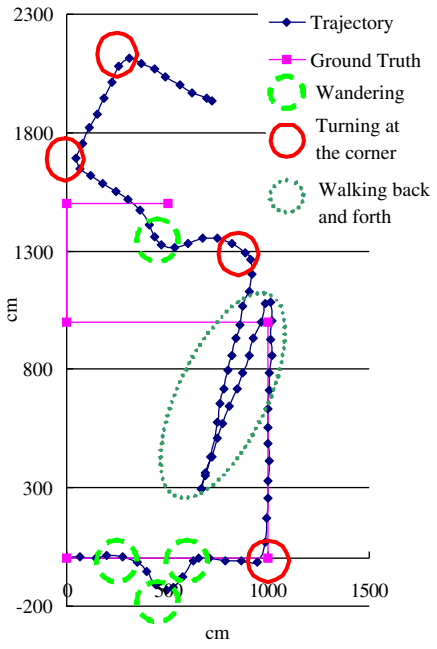

Fig. 24. The testing environment and route.

spots before the experiment started), shown as the green dashed circles in Fig. 24. The solid circles in Fig. 24 indicate when this walker made a 'real' turn at the corner. We repeated this experiment 10 times, and our results show that our location error is about 0.48 m, and the standard deviation is about 0.43 m.

6. Discussion and conclusion

In this work, 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 (Mohan et al., 2008; Raghu, Nam, Hossein, Saurabh, & Tarek, 2010) to obtain the scale information of the map. The idea is simple: 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, a little more data will bring the system to convergence. In addition, such a bootstrap phase only needs to be performed once for each building. Nevertheless, one limitation of this work is its reliance on a participatory sensing system to collect enough scale information for the map, and we are currently investigating how to overcome this. Furthermore, in our map matching algorithm, we compare all possible (R_T, R_M) pairs to find the one which is "*the most geometrically similar*". Two factors could contribute to the system's computational overhead: the frequency of performing the map matching process and the number of possible (R_T, R_M) pairs to be considered. In our implementation, we only perform map matching when the system detects the turning event (including 'fake turnings'). In other words, we only need to run the step estimation module when the user walks on the same corridor if there no fake turning events are detected. The number of possible (R_T, R_M) pairs depends on the complexity of the floor map and the user's movement patterns. The number of possible R_T after the last-visited corner is generally limited unless the walker wanders about frequently. However, the number of possible R_M could be significantly large depending on the complexity of the map. For example, a map with a high node degree distribution could potentially generate a larger number of possible R_M, and result in a higher computation overhead. We are currently studying this issue by considering utilizing the user's trajectory history data to reduce the space of possible R_M.

To conclude this paper, we've made the following contributions in this work. First, unlike previous crowdsourcing-based methods that required the drivers to manually report when they leave their parking spots, we utilize the phone sensors to automatically infer the parking space availability and notify the other drivers in advance by tracking the trajectory of the driver. Second, we implement a waist-mounted PDR method on a smart-phone by using the sensors in the phone to track the driver's walking trajectory. Second, we design a map matching algorithm to calibrate the direction errors from the gyroscope using building floor plans, which are readily available. Finally, our map matching algorithm implements three filters, namely the shape filter, angle filter and edge filter, to infer the user's last-visited corner. Our results show that we can achieve about 98% accuracy in estimating the user's walking distance, while the overall location error is about 0.48 m in our experiment. We found that it is important to have a mechanism such as ZUPT to calibrate sensor drift error, as the accuracy drops from 98% to 84% without ZUPT. Furthermore, we show that one might need to consider corridor length in order to detect corners.

One limitation of this work is its reliance on a participatory sensing system to collect enough scale information for the map, and we are currently investigating how to overcome this. In addition, we assume that the position of the phone is relatively stable in relation to the leg movements and do not take into account of the possibility of running in this study. Finally, we do not consider the multi-floor building scenario, since most of smart-phones do not have a barometer to detect the change to a different floor, and we leave this as a further direction for future work.

Acknowledgments

This work was supported by the National Science Council of Taiwan under Grants NSC 101-2220-E-006-010 and NSC 101-2221-E-006-098-MY3.

References

Aggarwal, P., Thomas, D., Ojeda, L., & Borenstein, J. (2011). Map matching and heuristic elimination of gyro drift for personal navigation systems in GPS denied conditions. *Measurement Science and Technology, 22*, 025205.

Alvarez, D., Gonzalez, R. C., Lopez, A., & Alvarez, J. C. (2006). Comparison of step length estimators from weareable accelerometer devices. In *28th Annual international conference of the IEEE in engineering medicine and biology society (EMBS '06)* (pp. 5964–5967).

Alvarez, J. C., Gonzalez, R. C., Alvarez, D., Lopez, A. M., & Rodriguez-Uria, J. (2007). Multisensor approach to walking distance estimation with foot inertial sensing. In *29th Annual international conference of the IEEE engineering in medicine and biology society (EMBS 2007)* (pp. 5719–5722).

Arnott, R., & Inci, E. (2006). An integrated model of downtown parking and traffic congestion. *Journal of Urban Economics, 60*, 418–442.

Ascher, C., Kessler, C., Wankerl, M., & Trommer, G. F. (2010). Dual IMU indoor navigation with particle filter based map-matching on a smartphone. In *Indoor positioning and indoor navigation (IPIN), 2010 international conference on* (pp. 1–5).

Ayala, D., Wolfson, O., Xu, B., Lin, J., & Dasgupta, B. (2011). Parking slot assignment games. In ACM SIGSPATIAL GIS.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24*, 509–522.

Centroid. Available from http://en.wikipedia.org/wiki/Centroid#cite_ref-0.

Chen, K. Y., & Bassett, D. R. (2005). The technology of accelerometry – Based activity: monitors: current & future. *American College of Sports Medicine*.

Chintalapudi, K., Iyer, A. P., & Padmanabhan, V. N. (2010). Indoor localization without the pain. In *16th ACM international conference on mobile computing and networking (MobiCom 2010)* (pp. 173–184). Chicago, IL, September 2010.

Dippold, M., (2006). Personal dead reckoning with accelerometers. In *Presented at the 3rd international forum on applied wearable computing 2006 (IFAWC2006)*. Bremen, Germany, March 15–16.

Dunford, N., & Schwartz, J. T. (1958). *Linear operators*. New York: Interscience Publishers. Vol. I.
Eric, F. (2005). Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications, 25*, 38–46.
Feliz, R., Zalama, E., & García-Bermejo, J. G. (2009). Pedestrian tracking using inertial sensors. *Journal of Physical Agents, 3*, 35–42.
Gafurov, D., Snekkenes, E., & Bours, P. (2007). Gait authentication and identification using wearable accelerometer sensor. In *2007 IEEE workshop on automatic identification advanced technologies* (pp 220–225).
Gerald, B., & Andreas, T. (2008). DiaTrace – Neuartiges assistenz-system für die gesundheitsprävention zur nahrungsaufnahme und bewegungserfassung. In *Deutscher kongress mit ausstellung/technologien – anwendungen – management.* Berlin, Germany.
Gillieron, P. -Y., & Merminod, B. (2003). Personal navigation system for indoor applications. In *Proceedings of the 11th IAIN world congress on smart navigation, systems and services.* Berlin.
Godha, S., Lachapelle, G., & Cannon, M. E. (2006). Integrated GPS/INS system for pedestrian navigation in a signal degraded environment. In *Presented at ION GNSS* (pp. 26–29).
Google Latitude. Available from http://www.google.com/intl/en/mobile/latitude/.
Gusenbauer, D., Isert, C., & Krösche, J. (2010). Self-contained indoor positioning on off-the-shelf mobile devices. In *Indoor positioning and indoor navigation (IPIN), 2010 international conference on* (pp 1–9).
Hao, Z., & Malik, J. (2003). Learning a discriminative classifier using shape context distances. In *Computer vision and pattern recognition, 2003. proceedings. 2003 IEEE computer society conference on* (Vol. 1, pp. I-242–I-247).
Ishikawa, T., Kourogi, M., Okuma, T., & Kurata, T. (2009). Economic and synergistic pedestrian tracking system for indoor environments. In *2009 International conference of soft computing and, pattern recognition* (pp. 522–527).
Ji, Y., Biaz, S., Pandey, S., Agrawal, P. (2006). ARIADNE: A dynamic indoor signal map construction and localization system. In *MobiSys*.
Jimenez, A. R., Seco, F., Prieto, C., & Guevara, J. (2009). A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. In *2009 IEEE international symposium on intelligent signal processing (WISP 2009)* (pp. 37–42).
Johnson, R. A. (1929). *Modern geometry: An elementary treatise on the geometry of the triangle and the circle*. Boston, MA: Houghton Mifflin, pp. 173–176, 249–250, and 268–269.
Jorgensen, F. (2008). Accurate step counter. U.S. Patent 0 172 203 A1, Jul. 17.
Krishna, C., Anand Padmanabha, I., & Venkata, N. P. (2010). Indoor localization without the pain. In *Presented at proceedings of the sixteenth annual international conference on Mobile computing and networking.* Chicago, Illinois, USA.
Ladetto, Q., & Merminod, B. (2002). *In step with INS-navigation for the blind, tracking emergency crews*. GPS World, pp. 30–38.
Lan, K. -C., & Shih, W. -Y. (2012). Using simple harmonic motion to estimate walking distance for waist-mounted PDR. In *2012 IEEE wireless communications and networking conference (WCNC)*.
Lan, K. -C., & Wang, H. -Y. (2013). On providing incentives to collect road traffic information. In *International wireless communications & mobile computing conference (IWCMC'13).* Cagliari, Sardinia, Italy, 1–5 July.
Lei, F., Antsaklis, P. J., Montestruque, L. A., McMickell, M. B., Lemmon, M., Yashan, S., et al. (2005). Design of a wireless assisted pedestrian dead reckoning system – the NavMote experience. *IEEE Transactions on Instrumentation and Measurement, 54*, 2342–2358.
Li, F., Zhao, C., Ding, G., Gong, J., Liu, C., & Zhao, F. (2012). A reliable and accurate indoor localization method using phone inertial sensors. In *14th ACM international conference on ubiquitous, computing (Ubicomp'12).*
Lim, H., Kung, C. L., Hou, J. C., Luo, H. (2006). Zero configuration robust indoor localization: Theory and experimentation. In *Infocom*.
Martin, M., & Michael, M. (2009). A step counter service for Java-enabled devices using a built-in accelerometer. In *Presented at proceedings of the 1st international workshop on context-aware middleware and services: affiliated with the 4th international conference on communication system software and middleware (COMSWARE 2009).* Dublin, Ireland.
Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., et al. (2010). ParkNet: Drive-by sensing of road-side parking statistics. *ACM MobiSys*.
Mohan, P., Padmanabhan, V. N., & Ramjee, R. (2008). Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on embedded network sensor systems. SenSys '08* (pp. 323–336). New York, NY, USA: ACM.
Ojeda, L., & Borenstein, J. (2006). Non-GPS navigation for emergency responders. In *International joint topical meeting: Sharing solutions for emergencies and hazardous environments*. Salt Lake City, UT, February 12–15.
Ojeda, L., & Borenstein, J. (2007a). Personal dead-reckoning system for GPS-denied environments. In *2007 IEEE international workshop on safety, security and rescue robotics (SSRR 2007)* (pp 1–6).
Ojeda, L., & Borenstein, J. (2007b). Non-GPS navigation for security personnel and first responders. *Journal of Navigation, 60*, 391–407.
Oliver, W., & Robert, H. (2008). Pedestrian localisation for indoor environments. In *Presented at Proceedings of the 10th international conference on Ubiquitous computing*. Seoul, Korea.
OpenSpot. Avaiable from http://techcrunch.com/2010/07/09/google-parking-open-spot/.
ParkNet. Avaiable from https://parknet.dk/.
Primospot. Avaiable from http://www.primospot.com/.
Raghu, K. G., Nam, P., Hossein, A., Saurabh, N., & Tarek, F. A. (2010). GreenGPS: A participatory sensing fuel-efficient maps application. In *Presented at proceedings of the 8th international conference on Mobile systems, applications, and services*. San Francisco, California, USA.
Roadify. Avaiable from http://www.roadify.com/.
Sagawa, K., Inooka, H., & Satoh, Y. (2000). Non-restricted measurement of walking distance. In *2000 IEEE international conference on systems, man, and, cybernetics* (Vol. 3, pp. 1847–1852).
Saghri, J. A., & Freeman, H. (1981). Analysis of the precision of generalized chain codes for the representation of planar curves. In *Pattern analysis and machine intelligence, IEEE transactions on* (Vol. PAMI-3, pp. 533–539).
Schneider, P. L., Crouter, S. E., Lukajic, O., & Bassett, D. R. (2003). Accuracy and reliability of ten pedometers for measuring steps over a 400-m walk. *Medicine & Science in Sports & Exercise, 35*, 1779–1784.
SFpark. Avaiable from http://sfpark.org/.
Shih, C. -C. (2010). The implementation of a G-sensdor-based pedometer. M.S. paper, Dept. C.S.I.E., National Central Univ., Taoyuan County, Taiwan.
Shin, S. H., Park, C. G., Hong, H. S., & Lee, J. M. (2005). MEMs-based personal navigator equipped on the user's body. In *Presented at ION GNSS 18th international technical meeting of the satellite division.* Long Beach, CA, 13–16 September.
Shin, S. H., Park, C. G., Kim, J. W., Hong, H. S., & Lee, J. M. (2007). Adaptive step length estimation algorithm using low-cost MEMS inertial sensors. In *Sensors applications symposium, 2007. SAS '07. IEEE* (pp. 1–5), 6–8 Feb.
Shoup, D. (2006). Cruising for parking. *Transport Policy, 13*(6), 479–486.
Simple Harmonic Motion. Available from http://en.wikipedia.org/wiki/Simple_harmonic_motion.
SpotScout. Avaiable from https://www.spotscout.com/.
Stenneth, L., Wolfson, O., Yu, P., & Xu, B. (2011). Transportation mode detection from mobile phones and GIS information. In *ACM SIGSPATIAL GIS*.
Stirling, R., Collin, J., Fyfe, K., & Lachapelle, F. (2003). An innovative shoe-mounted pedestrian navigation system. In *Presented at proceedings of European navigation conference GNSS 2003*. Graz, Austria, 22–25 April.
Su, C.-M., Chou, J.-W., Yi, C.-W., Tseng, Y.-C., & Tsai, C.-H. (2010). Sensor-aided personal navigation systems for handheld devices. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, ser. ICPPW '10* (pp. 533–541). Washington, DC, USA: EEE Computer Society.
Titterton, D. H., & Weston, J. L. (1997). *Strapdown inertial navigation technology*. Peter Peregrinus Ltd.
Walking diagram. Available from http://www.juliecallahan.me/walkcycle.jpg.
Wang, H., Sen, S., Elgohary, A., Farid, M., Youssef, M., & Choudhury, R. R. (2012). No need to war-drive: unsupervised indoor localization. In *Mobisys*.
Weinberg, H. (2002). Using the ADXL202 in pedometer and personal navigation applications. In *Application notes American devices*.
Xiaoping, Y., Bachmann, E. R., Moore, H., & Calusdian, J. (2007). "Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In *2007 IEEE international conference on robotics and automation* (pp. 2526–2533).
Yan, T., Hoh, B., Ganesan, D., Tracton, K., Lwuchukwu, T., & Lee, J. (2011). CrowdPark: A crowdsourcing-based parking reservation system for mobile phones. In *UMASS Technical Report., Tech. Rep. UM-CS-2011-001*.