ShoesLoc: In-Shoe Force Sensor-Based Indoor Walking Path Tracking

TUO YU, University of Illinois at Urbana-Champaign

HAIMING JIN, Shanghai Jiao Tong University, China

KLARA NAHRSTEDT, University of Illinois at Urbana-Champaign

Currently, in-shoe force sensors have been widely used for step counting and gait analysis. However, it has not been realized that in-shoe force sensors are also capable of tracking walking paths. In this paper, we present ShoesLoc, an indoor walking path tracking method based on in-shoe force sensors. We show that, based on the force signals from a user's shoes, it is possible to estimate the walking direction change and the stride length of each step with machine learning techniques. We further apply a particle filter to combine this information with the constraint of barriers in floor maps, and thus can determine the walking path and the current position of the user. To solve the problem of the low accuracy caused by cumulative walking direction errors, we improve the particle filter by designing the direction correction algorithm. Moreover, we propose the weight normalization method to handle the impact of handbags and backpacks. Our experimental results show that, after a convergence phase, ShoesLoc achieves the average location error of 0.9-1.3 m. Compared with traditional indoor tracking technologies, ShoesLoc does not require the installation of wireless anchors, and has good robustness to environment changes such as the magnetic interference.

CCS Concepts: • **Information systems** → **Location based services**; • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**.

Additional Key Words and Phrases: Indoor tracking, walking path tracking, force sensors, smart shoes

ACM Reference Format:

Tuo Yu, Haiming Jin, and Klara Nahrstedt. 2019. ShoesLoc: In-Shoe Force Sensor-Based Indoor Walking Path Tracking. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 31 (March 2019), 23 pages. <https://doi.org/10.1145/3314418>

1 INTRODUCTION

In recent years, the rapid evolution of mobile sensing technology has triggered the rise of smart shoes [4, 20, 35], the devices that can measure the motion of users' feet and analyze their activities. One typical design of smart shoes is based on force or piezoelectric sensors. For instance, Nike+ sensors [31] are used to count the steps of runners, and Stridalyzer [35] is a typical force mapping system that helps analyze the gaits of users. In this paper, we further explore the capability of in-shoe force sensors, and prove that it is possible to track the walking paths of users in indoor spaces based on force changes in shoes.

Indoor tracking and localization have been popular research areas. Most of the previous approaches are based on wireless anchors [3, 8, 18, 21, 30, 36, 37, 39, 44, 50]. For instance, Wi-Fi access points generate specific signal strength distributions in indoor space, and mobile devices such as cellphones measure the signal strength to locate themselves [2, 5, 14, 15, 33, 49]. However, the anchors need to be pre-installed, and site surveys are usually

Authors' addresses: Tuo Yu, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, tuoyu2@illinois.edu; Haiming Jin, Shanghai Jiao Tong University, Shanghai, China, jinhaiming@sjtu.edu.cn; Klara Nahrstedt, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, klara@illinois.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

2474-9567/2019/3-ART31 \$15.00

<https://doi.org/10.1145/3314418>

necessary, which lead to extra cost. In some large-scale indoor spaces such as shopping malls and airports, the number of Wi-Fi access points that cover each location is often limited, because they are deployed to provide network access instead of localization service [52]. Moreover, in many cases such as power outage, the anchors do not work. To solve these problems, some previous works proposed inertial sensor-based methods [24, 34, 47]. The accelerometers, gyroscopes and compasses in cellphones or Inertial Measurement Units (IMUs) are used to measure the motion of users and thus estimate the walking paths. However, the electronic compasses can be interfered by magnetic field changes [13, 23, 32], especially for the scenarios where the magnetic field changes significantly [6, 38], such as in manufacturing plants [27]. In contrast with the previous approaches, if we use force sensors alone to locate users, the cost of the anchor installation and site survey can be eliminated. Moreover, the in-shoe force sensors are hardly interfered by magnetic field changes.

In this paper, we present ShoesLoc, an indoor walking path tracking method which is based on in-shoe force sensors. The main idea is that, based on the signals from the force sensors deployed in insoles, we estimate the walking direction change and the stride length of each step made by the user. We then combine this information with the constraint of barriers on floor maps to determine the walking path and the current position of the user.

The main challenge is that, without the compass, the walking direction of the user is unknown, which makes it hard to track the walking trajectory. To solve this problem, ShoesLoc estimates the *relative angle change* of the walking direction instead. We extract features from the force signals, and train a Support Vector Machine (SVM) regression model to estimate the direction change made by each step of the user. Then, by accumulating the direction changes of the steps, we can track the change of the user's heading direction. The stride length of each step can also be estimated in a similar way. To further determine the absolute walking path of the user, we apply a particle filter to utilize the constraint of the barriers on floor maps. The wrong walking trajectories that go across barriers such as walls are eliminated in this way.

Moreover, based on the observation that a long straight walking trajectory must be parallel to one of the straight hallways on the map, we propose the direction correction algorithm to improve the performance of the particle filter, which reduces the cumulative direction error and the computation time. Furthermore, to handle the common case that carrying handbags or backpacks changes the force distribution in shoes, we propose the weight normalization method to normalize the force data, and thus reduce the impact of carrying bags.

The main contributions of our paper are:

- To the best of our knowledge, we are the first to explore the possibility of using in-shoe force sensors to track the walking paths of users, which has low deployment cost and is resistant to magnetic field changes.
- To address the major challenge of estimating walking trajectories without inertial sensors, we design the direction change and stride length estimation methods based on force signals, and apply the particle filter to determine walking paths.
- We propose the direction correction algorithm to further improve the performance of the particle filter. We also propose the weight normalization method to handle the impact of handbags and backpacks.
- We implement a prototype system and conduct extensive evaluations. Experimental results show that ShoesLoc can achieve the average location error of 0.9-1.3 m.

Besides being used as a new user tracking technique for smart shoes, ShoesLoc is also specifically effective for the scenarios where the anchors for localization are not available or the magnetic interference is non-negligible, such as in power outages or in the radiology departments of hospitals.

We organize the rest of our paper as follows. After introducing the related work in Section 2, we present an overview of ShoesLoc's framework in Section 3. Then, we successively introduce the methodologies and algorithms used by the system components in Section 4-7. In Section 8 and 9, we evaluate the performance of ShoesLoc with comprehensive experiments. We have further discussion in Section 10, and conclude our paper in Section 11.

2 RELATED WORK

Indoor localization and tracking: The research on indoor user tracking is tightly related to the indoor localization technology, and they have similar technical foundations. Multiple technologies have been applied in the field of indoor localization. For instance, there are systems based on radio frequency identification [19, 51], ultra-wideband transmission [21, 44], Bluetooth [3, 8], ultrasonic [36, 39] and video signals [29]. However, these methods require the installation of anchors in indoor spaces to communicate with users' devices or generate specific signal distributions. Therefore, they do not work in the regions where no anchors are deployed. The Wi-Fi received signal strength (RSS)-based localization technology solves this problem by using the widely installed Wi-Fi access points as the anchors [2, 5, 15]. However, there are cases where the access points are out of power or do not work, such as in power outages. Moreover, a site survey is usually needed to learn the RSS distribution on the whole floor, which leads to extra cost.

Compared with the indoor localization methods, the indoor tracking methods [18, 30, 37, 50] usually achieve a higher location accuracy because they can make better use of historical localization results by further applying techniques such as Kalman filters [14, 33, 49]. However, with the same fundamentals as the indoor localization technology, the previous indoor tracking approaches suffer the same aforementioned problems. On the other hand, the surveillance camera-based indoor tracking methods [7, 22] require the installation of cameras, which are usually even more expensive than wireless anchors. Some other approaches [16, 42] use cellphone cameras instead to provide indoor navigation services. However, the use of cellphone cameras is energy-expensive and would also cause privacy issues.

In our paper, ShoesLoc can work in a larger range of application scenarios because it uses the force sensors in users' shoes and does not require extra anchors or cameras. Moreover, except the floor map, ShoesLoc does not require any site survey or wireless signal measurements, which reduces the cost of the system setting up.

Motion sensor-based walking tracking: To eliminate the need of anchors, some previous works proposed to use the inertial and magnetic sensors to measure the motion of users, and thus track the walking trajectories. For instance, in [34], Zee used the motion sensors in cellphones to estimate the walking direction and count the steps of users, which can be used to recognize the walking paths if the floor map is available. In [24], based on the inertial sensors in cellphones, the algorithms for step detection, heading direction detection and step length estimation are also proposed to construct the walking paths of users.

However, the accuracy of these approaches is usually impacted by the position or the orientation of the phone carried by the user, which makes them impractical for real-life use [46]. Some other previous works instead utilize the foot-mounted IMUs. For instance, in [43], IMUs are used to measure the motion of users' feet, based on which the system can provide absolute positioning. In [25, 26], sensor tags are embedded into shoes, which can perceive the user's moving trace. However, the IMU-based approaches are usually vulnerable to magnetic interference, because the electronic compasses can be impacted by magnetic changes [13, 23, 32]. For example, in [34] it is shown that, the presence of magnetic materials such as metal in buildings can disturb a compass's perception of North, and the direction error of compasses in an office building can be as high as 30°. Thus, these approaches are not applicable for the regions where the magnetic field changes significantly [6, 27, 38]. In contrast, in ShoesLoc we use force sensors, which are resistant to most environment changes. Therefore, compared with the IMU-based approaches, ShoesLoc can work in more diverse scenarios.

To improve the localization accuracy, many previous motion sensor-based works applied the particle filters [34, 43, 46], which help eliminate the invalid trajectories that go across barriers in maps. However, the particle filters were applied based on the condition that the absolute walking direction can be measured by the sensors. Since the force sensors cannot provide the accurate walking direction, the classic particle filters do not work well because the lower accuracy of the estimated heading direction makes valid particles dropped easily, which causes huge computation burden. To solve this problem, we improve the particle filter by designing the direction

correction algorithm, which can correct the walking direction based on the map information and further reduce the computation time.

Force sensor-based walking tracking: Some previous works also considered the usage of in-shoe force sensors. However, most of them only use force sensors to count steps or analyze the gaits of users. In [40], the force sensors are used to count the steps made by the user, and thus help the inertial and magnetic sensors to track the gait states. Some commercial products such as the ReTiSense Stridalyzer [35] use force sensors to generate the force maps in shoes, which help doctors analyze the walking patterns of users. To the best of our knowledge, our work is the first to use in-shoes force sensors to track the walking paths of users.

3 SYSTEM OVERVIEW

In this section, we give an overview of the sensor deployment and the system architecture of our tracking method.

3.1 Sensor Deployment

Ideally, to monitor the force change on the whole insole, we should deploy a mass of force sensors to cover each point on the insole. However, because of the cost restriction, most of the commercial smart shoes and foot force mapping systems deploy only a limited number of force sensors for each insole. One typical deployment is shown in Fig. 1. For each insole, five force sensing resistors are deployed at the positions of inside metatarsals, outside metatarsals, outside midfoot, arch, and heel.

In our paper, we use the ReTiSense Stridalyzer insoles [35] to measure the force changes at the aforementioned five positions. We assign indices $i = 1, \dots, 5$ to the five sensors on each insole. When the user is walking, each sensor keeps measuring the force with the sampling rate of 15 Hz. Therefore, the inputs for our system are 10 discrete time series, denoted by $\{u_i^l(t), u_i^r(t) | i = 1, \dots, 5, t \geq 0\}$, where $u_i^l(t)$ is the data sequence received from sensor i in the left insole, and $u_i^r(t)$ is from sensor i in the right insole. Although $u_i^l(t)$ and $u_i^r(t)$ are discrete time series, we write them as continuous time series for convenience, which does not impact our following analysis. We will show that it is possible to further relax the sensor number requirement during our experiments in Section 8.1 and 8.2.

Fig. 1. Force sensor deployment for each insole.

3.2 System Architecture

As shown in Fig. 2, our system mainly consists of the following components.

Data preprocessor: This component receives the raw data $\{u_i^l(t), u_i^r(t) | i = 1, \dots, 5, t \geq 0\}$ from the force sensors, and then prepares the data for analysis. It first segments the data for each step made by the left or the right foot, and then adjusts the amplitude of the data to reduce the impact of the changing weight of the user. The outputs are the *step segments* from the left and the right insoles. Each step segment contains the force sequence from each sensor in each shoe for each step made by the user.

Fig. 2. System architecture of ShoesLoc.

Direction change estimator: This component estimates the direction change made by each left or right step. For the step segments of each step, it first extracts a set of features, and then calculates the angle change based on SVM. Note that the output is the relative angle change made by each step instead of the user's absolute walking direction.

Stride length estimator: To handle the impact of the changing walking speed, this component tracks the change of the stride length based on the step segments. Similar to the direction change estimator, it roughly estimates the stride length of each step based on SVM.

Walking path recognizer: This component determines the walking path with an improved particle filter. The inputs are the angle change sequence and the stride length sequence, as well as the map of the walking region. To handle the issue of high cumulative errors, we further design the direction correction algorithm to better utilize the map information.

Before the system can be used, a user needs to provide training data to train the components during the *offline stage*. The training format will be shown in Section 5.3 and 6.2. During the *online stage*, our method can track the position of the user after a convergence phase.

Next, we introduce the algorithms of these components in detail, respectively.

4 DATA PREPROCESSOR

The key design point of this component is to crop the force data sequences for each step, and further process the data to remove the effect of the changing body weight caused by carrying backpacks or handbags.

4.1 Step Segmentation

We process $\{u_i^l(t) | i = 1, \dots, 5\}$ and $\{u_i^r(t) | i = 1, \dots, 5\}$, respectively. To accurately detect the start and end point of each step, we focus on the total foot force on the insoles, i.e., $u^l(t) = \sum_{i=1}^5 u_i^l(t)$ and $u^r(t) = \sum_{i=1}^5 u_i^r(t)$. One example is shown by the last subgraph in Fig. 3. A naive method is finding the peaks in the signal sequence and each peak denotes one step. However, due to the various walking state, sometimes two peaks appear for each step (e.g., peak 1 and 2 in Fig. 3), which is especially the case when the user is walking at a low speed.

Inspired by the fact that the force made by a foot tends to be 0 when the foot has left the ground, we segment steps by detecting the low values of force data. Take the data from the left shoe as an example, a potential start point of a step is detected if $u^l(t) \geq \gamma_1$, and a potential end point of the step is detected if $u^l(t) < \gamma_1$, where γ_1 is a constant parameter and should be much smaller than the body weight of the user. In our implementation, we set $\gamma_1 = 10$ kg. To avoid the impact of the small noise peaks that are higher than γ_1 , a step is detected only if the max force value between its potential start and end point is higher than a constant threshold γ_2 . Clearly, γ_2 controls the robustness to the noise peaks, and we set $\gamma_2 = 20$ kg. If the max force value is lower than γ_2 , we discard this potential step. One discarded potential step is shown in Fig. 3. For each step, we crop the $u_i^l(t)$ (or $u_i^r(t)$) between its start and end point, and denote the step segments by $s_{i,k}^l(\tau)$ (or $s_{i,k}^r(\tau)$), where $i = 1, \dots, 5$, and $k = 1, 2, \dots$ is the index for each left (or right) step. τ denotes the time since the start point of the step, i.e., $\tau = 0$

Fig. 3. Data from the sensors while the user is walking. The data of the right foot between the two dashed straight lines show the force changes during a right turn.

at the beginning of the step segment. One example of the step segments for a single left step is labeled by five red boxes in Fig. 3.

4.2 Weight Normalization

It is common that people would carry handbags or backpacks when they are walking, which changes body barycenters and thus changes the amplitude and distribution of the force on the sensors. To address this issue, we design a method to normalize the force data from each sensor during the online stage.

Note that it is invalid to normalize the force segments by simply dividing the signals by their maximum or average values, because the force distribution is not proportional to the user's weight. Since we mainly focus on the amplitude changes of the force peaks in the step segments, we normalize the *peak amplitudes* only.

During the offline stage, the user is required to provide training data to train the direction change estimator and the stride length estimator, which will be introduced in Section 5.3 and 6.2. We further show that the training data can also provide information for normalization. We denote the cropped step segments in the whole training data by $\tilde{s}_{i,k}^l(\tau)$ and $\tilde{s}_{i,k}^r(\tau)$, where $i = 1, \dots, 5$, $k = 1, \dots, K$ and τ is in the range of the segment's time length. Then, we calculate the maximum and minimum value of the peak amplitudes for each sensor by

$$\tilde{s}_i^{l,\max} = \max_{k=1,\dots,K} \max_{\tau} \tilde{s}_{i,k}^l(\tau), \quad i = 1, \dots, 5, \quad (1)$$

$$\tilde{s}_i^{l,\min} = \min_{k=1,\dots,K} \max_{\tau} \tilde{s}_{i,k}^l(\tau), \quad i = 1, \dots, 5. \quad (2)$$

Fig. 4. Examples for possible ways to make a right turn.

Fig. 5. Training format for angle regression.

In case the training data contains outliers, we can delete the smallest and the largest 2% of $\max_t \tilde{s}_{i,k}^l(\tau)$ during the calculation. $\tilde{s}_i^{r,\max}$ and $\tilde{s}_i^{r,\min}$ can be calculated in the same way.

During the online stage, while the user is taking the k th left or right step, we calculate $s_i^{l,\max}$, $s_i^{l,\min}$ or $s_i^{r,\max}$, $s_i^{r,\min}$ among the previous G steps (i.e., step $k - G$ to $k - 1$). The parameter G must be large enough to guarantee that the user makes multiple turns in the G steps. We set $G = 100$ during our implementation. The calculation is similar to Equation (1) and (2), and the same outlier elimination method is applied. For the k th left step, the normalized force segments are

$$\tilde{s}_{i,k}^l(\tau) = \left(s_{i,k}^l(\tau) - s_i^{l,\min} \right) \frac{\tilde{s}_i^{l,\max} - \tilde{s}_i^{l,\min}}{s_i^{l,\max} - s_i^{l,\min}} + \tilde{s}_i^{l,\min}, \quad i = 1, \dots, 5. \quad (3)$$

For the k th right step, $\tilde{s}_{i,k}^r(\tau)$, $i = 1, \dots, 5$ are calculated in a similar way. The intuition is that, the range of the force peak amplitudes measured in the online stage is normalized to be the same with that of the peak amplitudes in the training data. Note that after the normalization, the force value around the start and end point would be smaller than 0. However, it does not impact the feature extraction in the following components.

One common scenario is that, the user is carrying a bag with a single hand or a single shoulder, which causes the unbalance of the force changes on different sensors. This unbalance can be complex and vary with different walking habits. For instance, when the user is carrying a heavy bag with the left hand, the body can either tilt to the left or right, which leads to different distributions of the signals' peak amplitudes on the ten sensors. As shown by Equation (3), our normalization method handles this problem by normalizing the force data from different sensors separately. After weight normalization, the range of the force peak amplitudes on each sensor is the same with that in the training data. Therefore, the unbalance of the force changes is eliminated.

For the k th step of the left/right foot, the data preprocessor outputs the normalized force sequence for each sensor i in the shoe as $\tilde{s}_{i,k}^l(\tau)$ or $\tilde{s}_{i,k}^r(\tau)$, which is sent to the direction change estimator and the stride length estimator.

5 DIRECTION CHANGE ESTIMATOR

When a user makes a turn during walking, the force distribution in the shoes changes. In this section, we show that it is possible to estimate the walking direction changes based on the force changes on the insoles. We first show our observations for the turning patterns of human, and then introduce how the direction change estimator extracts features and estimates the direction changes.

5.1 Turning Pattern Observation

It is intuitive that when a person makes a turn, the force made by the feet on insoles changes. Fig. 3 shows an example of the force changes during turning. The person makes a right turn around the 3,000th millisecond. We can observe that the amplitude of the total force also changes. This is caused by the fact that when the person makes the right turn using the right foot, the maximum force on the right inside metatarsals drops, as shown by the first subgraph of Fig. 3. Therefore, it is possible to recognize the turning actions according to the force changes on insoles. We further explore the possibility of estimating the angle changes of the walking direction based on the force data.

In ShoesLoc, we keep measuring the force data from both shoes. This is based on the observation that, the data from one shoe alone are not sufficient enough for turning detection. As illustrated by Fig. 4(a) and 4(b), there are at least two possible ways for a user to make a right turn. In Fig. 4(a), during the turning the user turns the body by putting extra force on the left foot. We define a *key step* as the step on which the user puts extra force to change the body direction. In Fig. 4(a) the left foot makes the key step, while in Fig. 4(b), the right foot makes the

key step. Apparently, the force data of key steps contain more information for direction change measurement. However, the non-key steps do not have distinct force change and thus would not help the turning detection. For instance, in the case shown by Fig. 3, the person makes the right turn following the turning type shown in Fig. 4(b). The right foot makes the key step, and the data from the left shoe do not show obvious change. Therefore, it is necessary to measure the force change in both shoes, otherwise some turning actions would not be detected.

Moreover, simply classifying the walking actions as “going straight” and “making a left/right turn” is not sufficient for walking path tracking. One example is shown in Fig. 4(c). When the user is making a 90° right turn at a low speed, it is possible that there are multiple key steps, and each key step makes a direction change smaller than 90°. However, by accumulating the angle changes of the key steps, we can still get the correct direction change. On the other hand, the paths in maps are not necessarily latticed. For instance, in Fig. 4(d) there are two possible walking paths after the user makes a right turn. Therefore, it is necessary to estimate the *direction change* made by each step, instead of simply classifying the step as “straight” or “turning”.

In the direction change estimator, we extract features from the step segments of each left/right step, and then estimate the direction change made by the step based on a trained SVM.

5.2 Feature Extraction

For the step segments of each step, we extract a set of features. The pattern in the features should not be highly impacted by the user’s walking speed, and thus the features we use are based on the amplitude of the force signals. Specifically, for the normalized step segment $\bar{s}_{i,k}^l(\tau)$ or $\bar{s}_{i,k}^r(\tau)$, the features are

$$F_{i,k}^l = \max_{\tau} \bar{s}_{i,k}^l(\tau), \quad i = 1, 2, \dots, 5, \quad (4)$$

$$F_{i,k}^r = \max_{\tau} \bar{s}_{i,k}^r(\tau), \quad i = 1, 2, \dots, 5. \quad (5)$$

Therefore, for the k th step made by the left or the right foot, the feature set $\{F_{i,k}^l | i = 1, 2, \dots, 5\}$ or $\{F_{i,k}^r | i = 1, 2, \dots, 5\}$ are sent to the subcomponent of angle regression.

5.3 Angle Regression

Training format: We first show how the training data for the angle regression are collected. The range of the direction change is $[-90^\circ, 90^\circ]$, where 0° denotes that the user’s walking direction does not change. The positive and negative values respectively denote the angles of the left and right turns made by the user. During the offline training, the user needs to make turns with 9 different angles, i.e., -90° to 90° with the interval of 22.5° , as shown by Fig. 5. Each time the user makes a turn, only one key step should be made. The training data for each angle include both the cases where the left or the right foot makes the key step. The training data are labeled with the corresponding angles, and are used to train a SVM regression model.

Based on our observations during the experiments, we only consider the direction change within $[-90^\circ, 90^\circ]$ for each footstep. This is because people usually do not make a turn larger than 90° within a single step. Instead, they make multiple shorter steps and the direction change of each step is smaller than 90° . Even if a person makes a 180° turn within a single step, it is natural that the velocity direction needs to be reversed, which leads to a stop or a pause in stepping. This stop or pause can be easily detected and ShoesLoc will restart the tracking process with the current location.

Ideally, the force changes made by the left and right foot should be symmetric, and the size of the training data could be reduced by half. However, in practice people’s walking patterns would be asymmetric [45], which means the feature patterns of left and right foot are asymmetric. Moreover, the diversity of the hardware used for measurement makes the models trained for left and right foot unexchangeable. Therefore, it is necessary to collect training data from both insoles.

SVM regression: In the offline stage, we extract the features from the training data following the method in Section 5.2, and then train two linear kernel-based SVM regression models for the left and right foot, respectively.

In the online stage, for the k th left or right step made by the user, the angle regression subcomponent receives the extracted features $\{F_{i,k}^l | i = 1, 2, \dots, 5\}$ or $\{F_{i,k}^r | i = 1, 2, \dots, 5\}$. Then, it predicts the angle change caused by the step with the corresponding trained model. The results are bounded in the range of $[-90^\circ, 90^\circ]$. While the user keeps walking, we get two sequences of angle changes caused by the left and right steps. We then merge the two sequences, and index them according to their time stamps. Thus, the output of the angle regression is a single angle change sequence, denoted by $\{\Delta\theta_n | n = 1, 2, \dots\}$. The left subgraph of Fig. 6 shows one example of $\{\Delta\theta_n\}$. The performance of the direction change estimator will be evaluated in Section 8.1.

Fig. 6. Example of the angle change sequence and the stride length sequence.

6 STRIDE LENGTH ESTIMATOR

To track the walking path of the user, counting steps and estimating walking direction are insufficient, because the user's stride length would change. Most of the previous approaches estimate the stride length by using inertial sensors [28]. With ShoesLoc, we prove that it is possible to estimate the stride length based on the force changes on insoles. To give an estimate for the stride length of each step, the stride length estimator extracts another set of features and also applies an SVM regression algorithm.

6.1 Feature Extraction

We use the normalized step segments $\bar{s}_{i,k}^l(\tau)$, $i = 1, \dots, 5$ from the left foot as examples to show how the features are extracted. The same process also applies to the step segments from the right foot.

Since the change of stride length is often related to the change of walking speed, the stride frequency is an important feature for stride length estimation. We use the time length of each step segment $\bar{s}_{i,k}^l(\tau)$ as a metric of the stride frequency. Note that for the same step k , the time lengths of $\bar{s}_{i,k}^l(\tau)$ for different sensor i are the same. The time length for the k th step is denoted as L_k^l .

Moreover, we calculate each step's loading rate and unloading rate, which denote the speeds at which the foot force reaches its peak and falls back to near-zero values, respectively:

$$\hat{L}_{i,k}^l = \frac{\max_{\tau} \bar{s}_{i,k}^l(\tau)}{\arg \max_{\tau} \bar{s}_{i,k}^l(\tau)}, \quad i = 1, 2, \dots, 5, \quad (6)$$

$$\check{L}_{i,k}^l = \frac{\max_{\tau} \bar{s}_{i,k}^l(\tau)}{L_k^l - \arg \max_{\tau} \bar{s}_{i,k}^l(\tau)}, \quad i = 1, 2, \dots, 5. \quad (7)$$

Note that for each $\bar{s}_{i,k}^l(\tau)$, τ is in the range of $[0, L_k^l]$. Since $\arg \max_{\tau} \bar{s}_{i,k}^l(\tau) \geq L_k^l$ never happens, $\check{L}_{i,k}^l$ is always positive. We also calculate the loading and unloading rate of the total force, i.e., we repeat the calculation in Equation (6) and (7) for the signal $\sum_{i=1}^5 \bar{s}_{i,k}^l(\tau)$. The results are denoted as $\hat{L}_k^l$ and $\check{L}_k^l$. We also reuse the features extracted in Section 5.2. Therefore, the features extracted from the k th left step for the stride length regression are

$$[L_k^l, \hat{L}_k^l, \check{L}_k^l, \hat{L}_{1,k}^l, \dots, \hat{L}_{5,k}^l, \check{L}_{1,k}^l, \dots, \check{L}_{5,k}^l, F_{1,k}^l, \dots, F_{5,k}^l]. \quad (8)$$

The features of the right steps can be calculated in a similar way.

6.2 Stride Length Regression

Training format: During the offline stage, the user is required to walk the same distance at different speeds, which lead to different stride lengths. The number of steps are counted and thus we get the average stride length by dividing the distance by the number of steps. For the test with each speed, we label the step segments with the average stride length in this test. The reason why we do not measure the actual stride length of each individual footstep is that, the cost of training data collection would be too high if the user needs to measure each stride length. Therefore, considering the practicality of ShoesLoc, we use the average stride length instead, which is easy to calculate.

SVM regression: Similar to the idea in Section 5.3, we train two linear kernel-based SVM regression models for the left and right foot to estimate the stride length. In the online stage, for the k th left/right step made by the user, the stride length regression subcomponent receives the features as in Formula (8), based on which it estimates the stride length of the step. We merge the two sequences of stride length from the two feet, and denote it by $\{d'_n | n = 1, 2, \dots\}$. We further smooth the sequence of $\{d'_n\}$ with the exponential moving average (EMA), i.e., $d_1 = d'_1$ and $d_n = \alpha \cdot d'_n + (1 - \alpha) \cdot d_{n-1}$ when $n > 1$. The coefficient α is a constant smoothing factor ($0 < \alpha < 1$), which controls the weights of the current and the previous observations, e.g., a higher α discounts older stride length values faster. We set $\alpha = 0.6$, which is fine-tuned based on the performance of the stride length estimator on a validation dataset. Then the stride length sequence $\{d_n | n = 1, 2, \dots\}$ is the output. One example is shown in the right subgraph of Fig. 6. Note that d_{n-1} and d_n denote the estimated values for different feet. We will evaluate the performance of the stride length estimator in Section 8.2.

7 WALKING PATH RECOGNIZER

The component of walking path recognizer receives the angle change sequence $\{\Delta\theta_n\}$, the stride length sequence $\{d_n\}$, as well as the map of the localization region, and runs a map-based particle filter to locate the user.

7.1 Key Idea of Walking Path Recognizer

The key idea is shrinking the possibilities for the user's location by constraining the estimated walking path with the barriers (e.g., walls) on the map. Initially, the user's location is unknown to ShoesLoc. However, based on the angle change sequence and the stride length sequence, we can estimate the possible walking trajectories of the user. Only the trajectories on the correct walking path can fit into the map, while the others are eliminated because they imply that the user has run into walls or other barriers. For each eliminated trajectory, we recreate a new valid trajectory based on the surviving ones. Therefore, once all the walking trajectories converge to a unique path, ShoesLoc can locate the user's location.

Some previous approaches have similar ideas, and apply augmented particle filters to solve the localization problem [34, 46]. Most of them use the accelerometer, gyroscope and compass to measure the user's walking direction, which helps determine the walking trajectory. However, in our paper we are facing a more complex case because we only know the relative angle changes instead of the user's absolute walking direction. Therefore, we can only have a roughly estimated walking trajectory which has cumulative direction errors. If the cumulative

errors are not eliminated, even the correct trajectory could be dropped because of the direction errors. Therefore, the particle filters proposed in the previous works cannot be directly applied, because they are based on absolute direction measurement and cannot handle large cumulative errors, which lead to a low location accuracy and a long computation time. To address this challenge, we improve the particle filter by proposing the direction correction algorithm, which makes better use of the map information than the previous approaches.

Direction Correction Algorithm: It is based on the natural observation that when the user is walking straight, the walking direction must be the same with that of some straight hallway on the map. Therefore, when we detect that the user is walking straight, we can correct the current estimated walking direction to the most similar hallway direction. This will efficiently reduce the cumulative errors of the walking direction. Moreover, since most of the hallways are horizontal or vertical on the map, there are usually only four possible directions of straight paths, which significantly reduces the computation time of the particle filter and makes realtime tracking and localization possible. We discuss the case of irregular-shape hallways in Section 10.2. The details of the algorithm are included in the workflow of the particle filter (Step 2 and 4) in the following subsection.

7.2 Improved Particle Filter

We first give the definition of particles used in the particle filter.

Particle: In our algorithm, a particle is defined as a possible walking state of the user. For the n th step made by the user, we denote a particle by the vector

$$p_n^m = [x_n^m, y_n^m, \theta_n^m], \quad m = 1, \dots, M, \quad n = 0, 1, \dots, \quad (9)$$

where (x_n^m, y_n^m) denotes the 2D location of the user on the map, θ_n^m is the absolute walking direction, m is the index of particles, and M denotes the total number of particles, which is a constant for a given map. $n = 0$ means that the user has not started to move and the particles are at their initial positions. Both (x_n^m, y_n^m) and θ_n^m are defined in an orthogonal coordinate system, where $\theta_n^m = 0$ is the positive direction of the x-axis and $\theta_n^m = 90^\circ$ is the positive direction of the y-axis.

Since the user's location is initially unknown, we distribute the particles randomly on the map. Based on the estimated direction change and stride length, we keep updating the particles for the user's each step. Then, the location changes of the M particles show M possible walking trajectories of the user. For the invalid trajectories blocked by the barriers on the map, we drop the corresponding particles and regenerate new ones on the valid trajectories. All the M trajectories will converge to a single path, and thus we can determine the correct location of the user.

The algorithm of the improved particle filter is shown in Algorithm 1. It has the following steps. Note that Step 1-5 in the algorithm do *not* refer to the physical footsteps made by the user.

(Step 1) Initialization: The algorithm takes the floor map MAP as an input, which includes the locations of hallways as well as barriers such as walls. As we do not know the user's initial location, we uniformly select U locations in the regions where the user can stay from the map with a constant density (line 1). For each location, we select V initial walking directions, i.e., $0, 360^\circ/V, \dots, 360^\circ(V-1)/V$, and for each direction, we generate W particles (line 2-7). Thus, the total number of particles is $M = UVW$. Larger U , V and W would provide higher localization accuracy, but also increase the computation time. Typically, V should be no less than 4. During our implementation, we set the density of the initial locations as $25/\text{m}^2$, and $V = 12$, $W = 2$. One example of the initial particles are shown in Fig. 7(b).

Moreover, based on MAP , we can get the direction set D of the hallways on the map (line 9). Typically, if all the hallways are latticed, the direction set is $D = \{0^\circ(360^\circ), 90^\circ, 180^\circ, 270^\circ\}$. Note that D can be modified according to the map. For example, if MAP contains the hallways in Fig. 4(d), 45° and 225° are also added to D .

After the initialization, the execution of Step 2-5 is triggered by each step made by the user.

Fig. 7. Example showing how the particle filter works based on the data in Fig. 6. (a) shows the real walking path of the user. (b)-(e) show how the particles converge. (e)-(f) show how the location of the user is tracked after the convergence. (g) shows the traceback of the walking path.

(Step 2) Straight trajectory detection: For the n th step made by the user ($n \geq 1$), the filter receives $\Delta\theta_n$ and d_n . Our observation shows that when the user makes a turn, it is unlikely that the degree change made by the key step is smaller than 40° . Therefore, the estimated result $|\Delta\theta_n| < 20^\circ$ almost means that the user is walking along a straight path. We set $\Delta\theta_n = 0^\circ$ in this case (line 11). Moreover, for the n th step, if $\Delta\theta_{n-3}, \dots, \Delta\theta_n$ are all 0° , we label this step as “straight” (line 12), which implies that the user is walking along a straight hallway.

(Step 3) Particles update: We update all the M particles for each new footstep. For the m th particle, we first set $m' = m$ and update particle $p_{n-1}^{m'}$ to p_n^m following line 14-20, where δ_n and λ_n are Gaussian random variables with zero means. In our prototype system, the standard deviation of δ_n is 0.1 m, while that of λ_n is 10° . Note that the cumulative error caused by the calculation in line 19 will be canceled by the direction correction. Then, we check whether the trajectory from $p_{n-1}^{m'}$ to p_n^m intersects any barriers (line 21). If so, p_n^m is dropped. A new m' is randomly selected from $\{1, \dots, M\}$ (line 22), based on which we pick a new particle $p_{n-1}^{m'}$ and update it to p_n^m following line 16-20. This process (line 15-23) is repeated until a valid p_n^m is generated.

(Step 4) Direction correction: Recall that we have gotten the direction set D in Step 1. If the n th step is labeled as “straight”, we change the value of θ_n^m of each particle to the closest value in D (line 24). Take $D = \{0^\circ(360^\circ), 90^\circ, 180^\circ, 270^\circ\}$ as an example. If the 10th step is “straight” and $\theta_{10}^{100} = 169^\circ$, we set $\theta_{10}^{100} = 180^\circ$; if $\theta_{10}^{101} = 350^\circ$, we set $\theta_{10}^{101} = 0^\circ$. Note that both Step 2 and 4 form the direction correction algorithm.

(Step 5) Convergence check: When Step 2-4 are repeated, eventually the particles converge to have similar locations. In line 25-26, we calculate the location center $(\tilde{x}_n, \tilde{y}_n)$ of the particles, which is the estimated user location. Moreover, we also calculate the standard deviation of the location differences to the center (line 27). If the standard deviation is lower than the accepted threshold σ_T , the particles have converged and we label the estimated location as “converged”, otherwise the location is labeled as “unconverged” and would have a low accuracy (line 28-29). In our implementation, we set $\sigma_T = 2$ m. For the n th footstep, $(\tilde{x}_n, \tilde{y}_n)$ and its label are the final outputs of our system. Fig. 7(b)-(e) show how the particles keep converging while the user is walking. Note that in Fig. 7(c), the particles in the horizontal routes cannot be eliminated because the absolute walking direction is unknown yet.

ALGORITHM 1: Procedure of Walking Path Recognizer

Input: Floor map MAP ; Initial location number U ; Initial direction number V ; Particle redundancy factor W ; Angle change sequence $\{\Delta\theta_n | n = 1, 2, \dots\}$; Stride length sequence $\{d_n | n = 1, 2, \dots\}$;

Output: Location sequence $\{(\tilde{x}_n, \tilde{y}_n) | n = 1, 2, \dots\}$ and “converged”/“unconverged” labels;

// (Step 1) Initialization

- 1 Uniformly select U initial locations in the walking region of MAP ;
- 2 $m \leftarrow 0$;
- 3 **foreach** selected initial location (x, y) **do**
- 4 **foreach** $v = 1, \dots, V$ **do**
- 5 **foreach** $w = 1, \dots, W$ **do**
- 6 $m \leftarrow m + 1$;
- 7 $p_0^m \triangleq [x_0^m, y_0^m, \theta_0^m] \leftarrow [x, y, 360^\circ(v-1)/V]$;
- 8 $M \leftarrow UVW$;
- 9 Generate direction set D based on MAP ;
- 10 // (Step 2-5) Run the following loop once for each footstep made by the user
- 10 **foreach** footstep $n = 1, 2, \dots$ **do**
- 11 // (Step 2) Straight trajectory detection
- 11 **if** $|\Delta\theta_n| < 20^\circ$ **then** $\Delta\theta_n \leftarrow 0^\circ$;
- 12 **if** $\sum_{j=0}^3 |\Delta\theta_{n-j}| = 0^\circ$ **then** Label step n as “straight”;
- 13 // (Step 3) Particles update
- 13 **foreach** $m = 1, \dots, M$ **do**
- 14 $m' \leftarrow m$;
- 15 **while true do**
- 16 // Update particle $p_{n-1}^{m'}$ to p_n^m
- 16 Generate random variables $\delta_n \sim N(0, 0.1^2)$, $\lambda_n \sim N(0, (10^\circ)^2)$;
- 17 $x_n^m \leftarrow x_{n-1}^{m'} + (d_n + \delta_n) \cdot \cos(\theta_{n-1}^{m'} + \Delta\theta_n + \lambda_n)$;
- 18 $y_n^m \leftarrow y_{n-1}^{m'} + (d_n + \delta_n) \cdot \sin(\theta_{n-1}^{m'} + \Delta\theta_n + \lambda_n)$;
- 19 $\theta_n^m \leftarrow (\theta_{n-1}^{m'} + \Delta\theta_n) \bmod 360^\circ$;
- 20 $p_n^m \triangleq [x_n^m, y_n^m, \theta_n^m]$;
- 21 **if** line segment between $(x_{n-1}^{m'}, y_{n-1}^{m'})$ and (x_n^m, y_n^m) intersects with barriers in MAP **then**
- 22 Randomly select $m' \in \{1, \dots, M\}$;
- 23 **else Break**;
- 24 // (Step 4) Direction correction
- 24 **if** step n is labeled as “straight” **then** $\theta_n^m \leftarrow \arg \min_{\omega \in D} \{180^\circ - |\theta_n^m - \omega| - 180^\circ|\}$;
- 25 // (Step 5) Convergence check
- 25 $\tilde{x}_n \leftarrow \frac{1}{M} \sum_{m=1}^M x_n^m$;
- 26 $\tilde{y}_n \leftarrow \frac{1}{M} \sum_{m=1}^M y_n^m$;
- 27 $\sigma \leftarrow \sqrt{\frac{1}{M} \sum_{m=1}^M ((x_n^m - \tilde{x}_n)^2 + (y_n^m - \tilde{y}_n)^2)}$;
- 28 **if** $\sigma < \sigma_T$ **then** Label $(\tilde{x}_n, \tilde{y}_n)$ as “converged”;
- 29 **else** Label $(\tilde{x}_n, \tilde{y}_n)$ as “unconverged”;
- 30 **return** $\{(\tilde{x}_n, \tilde{y}_n) | n = 1, 2, \dots\}$ and “converged”/“unconverged” labels;

After the particles have converged, the system can still track the location change of the user. Fig. 7(e)-(f) show how ShoesLoc continues locating the user after convergence. If the walking path of the user before convergence is needed, we can trace back the trajectories of the particles, as shown in Fig. 7(g).

If the particle filters used in the previous works are directly applied, a large amount of particles on valid paths will be dropped in Step 3 due to the cumulative direction error, which leads to long computation time. The direction correction in Step 4 significantly reduces the cumulative direction error and prevents the valid particles from being dropped, and thus reduces the computation time.

8 EVALUATION CONSIDERING CONTINUOUS WALKING

In this section, we first evaluate the accuracy of the direction change estimator and the stride length estimator. Then, we evaluate the performance of ShoesLoc in the case of continuous walking by conducting experiments in two different indoor spaces. The case of frequent walking state changes such as making frequent turns and pauses will be considered in Section 9. The training and test data are collected from 5 male and 2 female volunteers with different ages (23-52), heights (1.58-1.83 m) and weights (53-98 kg).

For the sensors in the shoes, we use the product of ReTiSense Stridalyzer [35] to collect the force data. Each insole contains five force sensors, and continuously transmits the data to our host device via a Bluetooth connection. The other components are implemented on a laptop host (MacBook Pro with 2.7 GHz Intel Core i7 and 16 GB RAM) using Matlab script. However, as Section 9 will show, the computation can be done on mobile devices and it does not impact the practicality of ShoesLoc.

8.1 Performance of Direction Change Estimator

In this subsection, we evaluate the performance of the direction change estimator. We collect training data from each of the 7 volunteers following the format in Section 5.3. For each turning direction in Fig. 5, each volunteer repeats walking 12 times. For 6 times the left foot makes the key steps, while the right foot makes the key steps for the other 6 times. We manually label the key steps, which are used to train the direction change estimator.

During the test, each of the volunteers is asked to walk in free style and make 40 turns. We record the walking of the volunteers with cameras, and measure the direction change made by each key step with a protractor. Moreover, the steps are manually divided into two classes, i.e., “straight” and “turn”. The probability distributions of the errors are shown in Fig. 8. Note that the sign of the errors follows Fig. 5, i.e., a positive error means that the estimated direction change shifts leftward compared with the ground truth, while a negative error means the estimated value shifts rightward. The average signed errors for the two classes are -2.30° and -1.95° , while the average absolute errors for the two classes are 16.79° and 16.04° , respectively. Moreover, the results also show that, when the user keeps moving, the cumulative error of the walking direction is not negligible, which confirms the necessity of applying the direction correction to cancel the cumulative error by making full use of the map information.

We further analyze the performance when the sensor number changes. We restrict the sensor number in each insole to be 1-5 respectively, and select the sensors that can achieve the best performance. The metric is the average error of both the “straight” and “turn” steps, and the two classes have the equal size. The results are shown in Fig. 9. For instance, if only 3 sensors can be deployed in each shoe, they should be at the inside metatarsals, the outside metatarsals and the outside midfoot to achieve the best performance. The average absolute error is 21.37° . It is shown that, by deploying only 4 sensors we can achieve a similar performance to the case of 5 sensors. Thus, for the purpose of walking direction change estimation, it is possible to relax the requirement of 5 sensors if not all of them can be deployed. Moreover, Fig. 9 implies that the sensor at the inside metatarsals makes the biggest contribution, which accords with our intuition that there are distinct force changes at the inside metatarsals when people are turning.

Fig. 8. Probability distribution of direction change estimation error.

Fig. 9. Average absolute direction change error when the sensor number changes. The corresponding positions of sensors are also shown.

Fig. 10. Probability distribution of stride length estimation error.

Fig. 11. Average absolute stride length error when the sensor number changes. The corresponding positions of sensors are also shown.

8.2 Performance of Stride Length Estimator

To collect the training data for the stride length estimator, the volunteers are required to walk along a straight path of 40 m. It is hard to control the stride length during walking. Thus, we require each volunteer to repeat this process 5 times, and each time walk at a different average speed, which ranges in 1-3 m/s at intervals of 0.5 m/s. We get the average stride length by dividing the distance by the number of steps, and the labeled results are used as the training data.

During the test, each volunteer walks along the same path 5 times. Each time the speed is not tightly controlled, and the average speed varies around 1-3 m/s. We get the ground truth of each footstep's stride length by cameras and tape measures. The probability distribution of the errors are shown in Fig. 10. The average signed error is -1.22 cm, and the average absolute error is 6.25 cm. Since the average stride length is 72.60 cm, the error is 8.61%, which is sufficient for the purpose of walking path tracking because the cumulative error can be reduced by the particle filter.

We also evaluate the performance of the stride length estimator when the number of available sensors are changed. For a given number of sensors, we select the sensors that can achieve the lowest error. The results are given in Fig. 11. It is shown that, when only 3 sensors can be deployed, the average absolute error is still no larger

Fig. 12. CDF of absolute direction change estimation error with different loads.

Fig. 13. CDF of absolute stride length estimation error with different loads.

than 6.41 cm. Moreover, it can be seen that the sensor at the heel is the most important one for the stride length estimation.

8.3 Impact of Carrying Bags

As mentioned before, carrying handbags or backpacks changes the force distribution on insoles. In ShoesLoc we design the weight normalization method to reduce this impact. To evaluate the robustness of ShoesLoc in the face of changing body barycenter, we require each volunteer to carry a bag of 2 kg or 5 kg. Four of the volunteers carry the bag with two shoulders and the others carry the bag by hands. The experiments in Section 8.1 and 8.2 are repeated, and the cumulative distribution functions (CDFs) of the absolute errors are shown in Fig. 12 and 13.

Fig. 12 and 13 show that, by applying weight normalization, the error of the direction change and stride length estimation can be reduced. The average angle errors without the normalization are 28.04° and 29.55° , while after the normalization the errors are 17.39° and 19.07° , respectively. The average errors of stride length estimation are also improved from 7.12 cm and 9.38 cm to 6.37 cm and 6.32 cm, respectively. Therefore, with the weight normalization method, we can significantly reduce the impact of carrying bags.

8.4 Performance of ShoesLoc

To evaluate the overall performance of ShoesLoc, we require the volunteers to walk in two indoor spaces, as shown in Fig. 14. During the test, each of the 7 volunteers walks randomly in each test region for 30 minutes. The possible walking paths are shown in Fig. 14. During the walking, the volunteers are asked to carry their backpacks and handbags (1-5 kg). The way that they carry the bags is not controlled, e.g., the backpack can be carried by either one or two shoulders. During each test, the volunteers are allowed to take breaks in place. The break time is not included in the 30 minutes. The stoppages can be easily detected since the force signals stop showing periodic changes. The tracking process is resumed when a break ends. We use cameras to record the real walking trajectories, and use time stamps to match them with the localization results. The trained models in Section 8.1 and 8.2 are used in this experiment.

Before ShoesLoc can track the location of the user, the particles must converge (Step 5 of Algorithm 1). It is meaningless to evaluate the tracking accuracy before the convergence, because the user's location has not yet been uniquely determined. Moreover, it is hard to control the time length of the convergence phase because it depends on the complexity of the walking path and the floor structure, as well as the walking speed of the user. Therefore, we divide the working process before and after the convergence into two phases, i.e., the convergence phase and the tracking phase, and mainly focus on the tracking phase. During each walk, once the particles have converged and entered the tracking phase, we continue recording 500 steps made by the volunteer. After that we

Fig. 14. The floor maps of the test regions. The dashed lines show the path used in Section 8.5.

restart ShoesLoc and the convergence phase begins again. Thus, each walk of 30 minutes is divided into several tests, and each test has two separate phases. During the experiment in Fig. 14(a), 40 tests are generated, i.e., each volunteer achieves 5.7 convergences on average. For the test in Fig. 14(b), 38 tests are generated.

Fig. 15. Location error during two walks.

As examples, the location errors during two walks are shown in Fig. 15. We also evaluate the performance of the particle filter used in [34] where the direction correction is not applied. Fig. 15(a) shows that, ShoesLoc keeps converging while the person is walking, and the convergence phase ends around the 110th step. After that, the system enters the tracking phase and still maintains high accuracy. If tracing back is applied, the accuracy during the convergence phase can be significantly improved. On the other hand, if direction correction is not applied, the particles once tend to converge before the 95th step. However, after that they keep diffusing during the whole walk, and thus the user cannot be localized. The reason is that, without the direction correction algorithm, it is easy for the particles on valid routes to go cross the walls because of the cumulative direction errors. Therefore, the direction correction algorithm significantly improves the performance of the particle filter, which is also confirmed by the other example shown in Fig. 15(b).

The average step number that a volunteer takes before convergence is 69 in Fig. 14(a) and 85 in Fig. 14(b). The main reason why more steps are needed to converge in Fig. 14(b) is that, its floor map contains more symmetric corridors, which makes the convergence a little harder.

We also evaluate the location error during the tracking phase of all the tests. The results are shown in Fig. 16. The average errors for the two floor maps are 0.88 m and 1.30 m, respectively. The case without direction

Fig. 16. CDF of location error in two floor maps.

Fig. 17. CDF of location error when the walking speed changes.

correction is excluded because the particles cannot converge. Moreover, if tracing back is applied, the average errors for the whole walking paths are 0.86 m and 1.05 m, respectively. Thus, the average location error of ShoesLoc is lower than that of the Wi-Fi RSS-based methods (2-3 m) [48], and is comparable with that of the inertial sensor-enhancing approaches (1.5-2 m) [47]. Meanwhile, ShoesLoc requires no site survey and has better robustness to magnetic interference.

To evaluate the realtime performance of ShoesLoc, we also record the computation time. When the particle number is 60,000, the average computation time for each step is 0.35s. If direction correction is not applied, the average computation time is 0.73s. This is because the high dropping rate of the particles on valid paths increases the time cost of generating new valid particles. In some extreme cases, more than 90% of the particles could be dropped at a step, which leads to a long computation time. Given that the average time length of people's each step is around 0.70-0.81s [10], ShoesLoc is able to provide realtime localization and tracking. A shorter computation time can be achieved, if the particle number is further reduced. In Section 9, we will show that ShoesLoc is also efficient when working on a mobile device.

8.5 Impact of Walking Speed

In Section 8.4, we have no control on the walking speed of users. In this subsection, we evaluate the impact of the walking speed on the performance of ShoesLoc. Since it is hard to measure and control the actual instantaneous speed of users, we use the average speed instead. The average speed used for our tests ranges from 1 m/s to 3 m/s at intervals of 0.5 m/s. As shown in Fig. 14(a), each volunteer is required to go along a given path 5 times. For each time, a different average speed is followed. During each walking, an experimenter walks after the volunteer and uses a timer to help the volunteer reach the destination at the given average speed. We then calculate the location error in the tracking phase. The results are shown in Fig. 17. When the average speed changes from 1 m/s to 3 m/s, the average errors of all the volunteers' walks are 1.09 m, 0.83 m, 0.81 m, 0.98 m and 0.85 m, respectively. Therefore, the change of the walking speed does not seriously impact the performance of ShoesLoc. This is because the features used for direction change estimation are not highly impacted by the walking speed, and the design of the stride length estimator naturally handles the speed changes well.

9 EVALUATION CONSIDERING FREQUENT TURNS, PAUSES AND SHOE TYPE CHANGES

In Section 8, some walking paths selected by the volunteers go through long straight hallways. In our daily life, it is also likely that users' walking paths include more short straight lines with more turns, e.g., walking from a desk to a printer or to a meeting room. Moreover, users would also make short pauses while walking, e.g., looking at a poster, or greeting someone. On the other hand, it is possible that the type of the shoes that the user

Fig. 18. Test region in Section 9.

Fig. 19. CDF of location error when users make frequent turns. The impacts of frequent pauses and different shoe types are also shown in the figure.

wears is different during the training and the daily usage. In this section, we evaluate the impacts of frequent turns, pauses, and shoe type changes.

In the indoor space shown in Fig. 18, we conduct a similar experiment as in Section 8.4. We recruit another group of volunteers (6 males and 5 females) with different ages (21-53), heights (1.62-1.85 m) and weights (57-104 kg). The same training format is followed, and all the volunteers wear hard bottom shoes during the training. To test ShoesLoc's performance on mobile devices, we implement our components on an iPad (2017) with iOS 11.2.2 operating system. To adapt to the computing power of the device, we adjust the density of initial particle locations, and the particle number is 45,000.

We design four test cases for each volunteer. In *Case 1*, the volunteer walks randomly for 10 minutes. The only rule is that a left or right turn must be made at each intersection. To make the comparison among different test cases meaningful, we record the walking path, and require the volunteer to walk along the same path in the following test cases. In *Case 2*, the volunteer is required to make a 2-second pause every 10 seconds (controlled by an interval timer), and continue to walk along the original direction after the pause. In both Case 1 and Case 2, all the volunteers wear hard bottom shoes. In *Case 3*, the volunteer follows the same path, but wears sneakers with air cushions instead and makes no stop. In *Case 4*, the volunteer wears sneakers and repeats Case 2.

The test results without traceback are shown in Fig. 19. The average errors of the four test cases are 1.03 m, 0.99 m, 1.15 m and 1.27 m, respectively. The average step numbers before convergence are 90, 95, 92 and 97, respectively.

The comparison between Case 1 and the experiment in Section 8.4 shows that, frequently turning does not impact the average error. On the other hand, while the floor plan in Fig. 18 has a similar scale to those in Fig. 14, the step number before convergence increases. This is because the floor plan contains multiple corridors in similar shapes, and more short straight lines make it harder to uniquely determine the user's location. However, ShoesLoc can still converge and achieve a good accuracy.

The comparison between Case 1 and Case 2 as well as Case 3 and Case 4 shows that there is no obvious impact caused by frequent pauses. Making a stop and restarting walking would slightly increase the direction and stride length estimation errors for the footsteps before and after the pause. However, the particle filter adds random variables (δ_n and λ_n) to the estimated values, which reduces the impact of the estimation errors for these footsteps. The EMA smoothing in the stride length estimator further mitigates this impact.

The results of Case 3 and Case 4 show a slight decrease in accuracy. This is because air cushions attenuate the maximum impact force on insoles when the shoes are touching the ground [1], which also changes the force

distribution on the sensors. However, the weight normalization method normalizes the measured data and keep the distribution of the force peak amplitudes consistent with that in the training data. Therefore, ShoesLoc still maintains good performance even if the shoe type in the tests is different from that in the training.

During the tests, the average computation time for each step is 0.43s, which is still efficient for realtime tracking. We also analyze the device's energy log with the Instruments application of Xcode, and use the *energy usage level* as the metric [17]. The level ranges from 0 to 20, which indicates how much energy an application is using at the given time. The sampling rate is 1 Hz, and a lower average level means that the application consumes fewer energy. The average energy usage level of ShoesLoc during the tests is 10/20. As a benchmark, our test shows that the average level of Apple Maps is 8/20. Therefore, the energy consumption of ShoesLoc is acceptable since users typically do not keep using the tracking service all the time.

10 DISCUSSION

10.1 Accelerating Convergence

Before ShoesLoc can locate the user, the user needs to keep walking until the particles converge. The speed of the convergence depends on the user's walking speed, the complexity of the walking path, as well as the structure of the barriers on the map. To speed up the convergence, similar to the idea in [41], we can further make use of the special facilities on the map to reduce the possible region of the user's initial location. For instance, the user must go through entrances to enter buildings. Also, by using the force sensors in the shoes, it is easy to detect the actions of going up/down stairs or using elevators. By recognizing the start point of the user's walking path, the convergence process can be significantly sped up. In our paper, to guarantee the practicality of ShoesLoc, we only focus on the general case that the initial location is unknown. The walking path tracking with known initial locations is a special case of the challenge solved by ShoesLoc.

10.2 Hallways in Irregular Shapes

Although most hallways are straight, some of them would still have complex shapes. For instance, in some shopping malls the hallways are curving or in the shape of arcs. However, usually the radius of the arcs are large enough that they can be considered as straight hallways by ShoesLoc. Specifically, if the direction difference between the start and the end of a curving hallway is smaller than 30° , we treat it as a straight hallway. If the direction changes a lot, we can crop the hallway into shorter hallways and treat each of them as a straight hallway. For more complex irregular hallways, we can simply consider them as non-straight hallways, which can still be handled by ShoesLoc.

10.3 Security Issue

As what we have mentioned, there have been commercial force sensors or force mapping systems for shoes [4, 20, 31, 35]. Some of these products allow users to upload their data to the cloud for analysis or statistics. The design of ShoesLoc implies that, if the data are leaked, it is possible for unauthorized third-parties to recognize the daily walking paths of the users. Although the training data would be unavailable, it is still possible to learn the walking pattern of the users by recognizing some common actions. For example, it is easy to detect the action of going up/down stairs, during which a user needs to make several left or right turns at each floor. Then the attacker can crop the training data for the direction change estimation. We will continue to explore the privacy leakage through the force data from smart shoes in our future work.

10.4 General Model for Walking Path Tracking

Currently, because of the diversity of walking habits, ShoesLoc requires each user to provide training data before use. Moreover, our weight normalization method assumes that the weight of the user is consistent during the

training data collection. If we train a general model by simply mixing the training data from different people, the signal measured during the user's walking will be incorrectly normalized to fit the data patterns of the heaviest or lightest training data providers. Some other factors such as the measurement differences caused by the device diversity would also lead to a low accuracy of the general model. However, there are still potential solutions. For instance, based on a large amount of data, it is possible to train the general model with deep neural networks. We leave it to our future work to design a system that works without requiring training data from each user.

10.5 Obtaining and Preprocessing Floor Maps

In our paper, we assume that the indoor floor maps are available to users' devices. In practice, there are multiple possible sources of floor maps. For instance, Google Maps [12] provides the floor plans of a large number of buildings, which can be fetched by users' devices. Moreover, since the rooms and hallways on the maps are labeled with different colors, the walking regions and barriers can be programmatically recognized with image processing techniques. For the buildings where digital floor plans are not available yet, there have been approaches that reconstruct indoor maps based on crowdsourcing techniques [9, 11]. In our implementation, after we have obtained the floor plan image, we preprocess the floor map by dividing it into a matrix of small grids with the resolution of $0.2 \text{ m} \times 0.2 \text{ m}$. The grids occupied by walls and barriers are programmatically recognized and labeled in the matrix according to their colors in the image. In this way, we can locate possible walking regions and detect if a walking trajectory goes through walls or barriers.

11 CONCLUSION

In this paper, we present ShoesLoc, a novel indoor walking path tracking method that is based on in-shoe force sensors. We design algorithms to estimate the walking direction change and the stride length based on the force signals, and apply the particle filter to determine the walking path. We further propose the direction correction algorithm and the weight normalization method to improve the particle filter's performance and handle the impact of handbags and backpacks. The experimental results show that ShoesLoc achieves the accuracy of 0.9-1.3 m, which is sufficient for walking path tracking.

ACKNOWLEDGMENTS

This research was funded by the National Science Foundation under award number CNS-1330491. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

REFERENCES

- [1] Arnel Aguinaldo and Andrew Mahar. 2003. Impact loading in running shoes with cushioning column systems. *Journal of applied biomechanics* 19, 4 (2003), 353–360.
- [2] Nasim Alikhani, Shima Amirinloo, Vahideh Moghtadaiee, and Seyed Ali Ghorashi. 2017. Fast fingerprinting based indoor localization by Wi-Fi signals. In *2017 7th International Conference on Computer and Knowledge Engineering (ICCCKE)*. IEEE, 241–246.
- [3] Marco Altini, Davide Brunelli, Elisabetta Farella, and Luca Benini. 2010. Bluetooth indoor localization with multiple neural networks. In *IEEE 5th International Symposium on Wireless Pervasive Computing 2010*. IEEE, 295–300.
- [4] Altra. 2018. Altra IQ powered by iFit. Retrieved March 22, 2018 from <https://www.altrarunning.com/iq>
- [5] Paramvir Bahl, Venkata N Padmanabhan, et al. 2000. RADAR: An in-building RF-based user location and tracking system. In *IEEE infocom*, Vol. 2. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 775–784.
- [6] John Burnett and Yaping Du. 2001. Low frequency magnetic interference in high-rise buildings. In *Seventh International IBPSA Conference: Building Simulation*, Vol. 7. 327–333.
- [7] Quin Cai and Jake K Aggarwal. 1998. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, 356–362.
- [8] Sudarshan S Chawathe. 2009. Low-latency indoor localization using bluetooth beacons. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 1–7.

- [9] Si Chen, Muyuan Li, Kui Ren, and Chunming Qiao. 2015. Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos. In *2015 IEEE 35th International conference on distributed computing systems*. IEEE, 1–10.
- [10] Claire T Farley and Octavio Gonzalez. 1996. Leg stiffness and stride frequency in human running. *Journal of biomechanics* 29, 2 (1996), 181–186.
- [11] Ruipeng Gao, Mingmin Zhao, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Tao Wang, and Xiaoming Li. 2014. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 249–260.
- [12] Google. 2018. Google Maps. Retrieved September 1, 2018 from <https://www.google.com/maps/about/partners/indoormap/>
- [13] Mahanth Gowda, Justin Manweiler, Ashutosh Dhekne, Romit Roy Choudhury, and Justin D Weisz. 2016. Tracking drone orientation with multiple GPS receivers. In *Proceedings of the 22nd annual international conference on mobile computing and networking*. ACM, 280–293.
- [14] İsmail Güvenc. 2003. *Enhancements to RSS based indoor tracking systems using Kalman filters*. Ph.D. Dissertation. University of New Mexico.
- [15] Suining He and S-H Gary Chan. 2016. Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 466–490.
- [16] Harlan Hile and Gaetano Borriello. 2008. Positioning and orientation in indoor environments using camera phones. *IEEE Computer Graphics and Applications* 28, 4 (2008), 32–39.
- [17] Apple Inc. 2018. Energy Efficiency Guide for iOS Apps. Retrieved September 26, 2018 from <https://developer.apple.com/library/archive/documentation/Performance/Conceptual/EnergyGuide-iOS/MonitorEnergyWithInstruments.html>
- [18] Christian S Jensen, Hua Lu, and Bin Yang. 2009. Graph model based indoor tracking. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 122–131.
- [19] Guang-yao Jin, Xiao-yi Lu, and Myong-Soon Park. 2006. An indoor localization mechanism using active RFID tag. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, Vol. 1. IEEE, 4–pp.
- [20] Kickstarter. 2018. Digitsole Smartshoe The World's First Intelligent Sneaker. Retrieved March 22, 2018 from <https://www.kickstarter.com/projects/141658446/digitsole-smartshoe-the-worlds-first-intelligent-s>
- [21] Sivanand Krishnan, Pankaj Sharma, Zhang Guoping, and Ong Hwee Woon. 2007. A UWB based localization system for indoor robot navigation. In *2007 IEEE International Conference on Ultra-Wideband*. IEEE, 77–82.
- [22] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. 2000. Multi-camera multi-person tracking for easy living. In *Proceedings Third IEEE International Workshop on Visual Surveillance*. IEEE, 3–10.
- [23] KyungMoon Lee, YangHwan Kim, JaeMu Yun, and JangMyung Lee. 2005. Magnetic-interference-free dual-electric compass. *Sensors and Actuators A: Physical* 120, 2 (2005), 441–450.
- [24] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. 2012. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*. ACM, 421–430.
- [25] Chang Liu, Lei Xie, Chuyu Wang, Jie Wu, and Sanglu Lu. 2015. FootStep-tracker: an anchor-free indoor localization system via sensing foot steps. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ACM, 285–288.
- [26] Chang Liu, Lei Xie, Chuyu Wang, Jie Wu, and Sanglu Lu. 2016. Track Your Foot Step: Anchor-free Indoor Localization based on Sensing Users' Foot Steps. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 201–209.
- [27] DAVID MARCO, GEORGE EISINGER, and DAVID L HAYES. 1992. Testing of work environments for electromagnetic interference. *Pacing and Clinical Electrophysiology* 15, 11 (1992), 2016–2022.
- [28] Benoit Mariani, Mayté Castro Jiménez, François JG Vingerhoets, and Kamiar Aminian. 2013. On-shoe wearable sensors for gait and turning assessment of patients with Parkinson's disease. *IEEE transactions on biomedical engineering* 60, 1 (2013), 155–158.
- [29] Rainer Mautz and Sebastian Tilch. 2011. Survey of optical indoor positioning systems. In *2011 international conference on indoor positioning and indoor navigation*. IEEE, 1–7.
- [30] Paul Meissner, Erik Leitingner, and Klaus Witrisal. 2014. UWB for robust indoor tracking: Weighting of multipath components for efficient estimation. *IEEE Wireless Communications Letters* 3, 5 (2014), 501–504.
- [31] Nikeplus. 2018. Set Up, Calibrate and Troubleshoot Nike+ Sensors. Retrieved March 22, 2018 from http://support-en-us.nikeplus.com/app/answers/detail/a_id/35503/p/3169,3171
- [32] Lauro Ojeda and Johann Borenstein. 2000. Experimental results with the KVH C-100 fluxgate compass in mobile robots. In *Proceedings of the IASTED International Conference on Robotics and Applications*. 1–7.
- [33] Anindya S Paul and Eric A Wan. 2009. RSSI-based indoor localization and tracking using sigma-point Kalman smoothers. *IEEE Journal of Selected Topics in Signal Processing* 3, 5 (2009), 860–873.
- [34] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. 2012. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 293–304.
- [35] ReTiSense. 2018. The new Stridalyzer INSIGHT Clinical Diagnostic Solution. Retrieved March 8, 2018 from <http://www.retisense.com/>

- [36] Martin Scherhäufl, Reimar Pfeil, Markus Pichler, and Achim Berger. 2012. A novel ultrasonic indoor localization system with simultaneous estimation of position and velocity. In *2012 IEEE Topical Conference on Wireless Sensors and Sensor Networks*. IEEE, 21–24.
- [37] Vladimir Sović, Akshay Athalye, Miodrag Bolić, and Petar M Djurić. 2011. Particle filtering for indoor RFID tag tracking. In *2011 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 193–196.
- [38] D Spackman, D Kacprzak, and JK Sykulski. 2007. Magnetic interference in multi-pickup monorail inductively coupled power transfer systems. *Journal of the Japan Society of Applied Electromagnetics and Mechanics* 15, 3 (2007), 238–241.
- [39] Enno-Edzard Steen, Marco Eichelberg, Wolfgang Nebel, and Andreas Hein. 2012. A novel indoor localization approach using dynamic changes in ultrasonic echoes. In *Ambient Assisted Living*. Springer, 123–133.
- [40] Young Soo Suh and Sangkyung Park. 2009. Pedestrian inertial navigation with gait phase detection assisted zero velocity updating. In *2009 4th International Conference on Autonomous Robots and Agents*. IEEE, 336–341.
- [41] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. 2012. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 197–210.
- [42] Martin Werner, Moritz Kessel, and Chadly Marouane. 2011. Indoor positioning using smartphone camera. In *2011 International Conference on Indoor Positioning and Indoor Navigation*. IEEE, 1–6.
- [43] Oliver Woodman and Robert Harle. 2008. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 114–123.
- [44] Henk Wymeersch, Stefano Marano, Wesley M Gifford, and Moe Z Win. 2012. A machine learning approach to ranging error mitigation for UWB localization. *IEEE transactions on communications* 60, 6 (2012), 1719–1728.
- [45] Yujia Xiang, Jasbir S Arora, and Karim Abdel-Malek. 2011. Optimization-based prediction of asymmetric human gait. *Journal of Biomechanics* 44, 4 (2011), 683–693.
- [46] Hongwei Xie, Tao Gu, Xianping Tao, Haibo Ye, and Jian Lu. 2016. A reliability-augmented particle filter for magnetic fingerprinting based indoor localization on smartphone. *IEEE Transactions on Mobile Computing* 15, 8 (2016), 1877–1892.
- [47] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. 2015. Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys (Csur)* 47, 3 (2015), 54.
- [48] Ali Yassin, Youssef Nasser, Mariette Awad, Ahmed Al-Dubai, Ran Liu, Chau Yuen, Ronald Raulefs, and Elias Aboutanios. 2016. Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys & Tutorials* 19, 2 (2016), 1327–1346.
- [49] Jaegool Yim, Seunghwan Jeong, Kiyoun Gwon, and Jaehun Joo. 2010. Improvement of Kalman filters for WLAN based indoor tracking. *Expert Systems with Applications* 37, 1 (2010), 426–433.
- [50] Junhui Zhao and Yongcai Wang. 2008. Autonomous ultrasonic indoor tracking system. In *2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*. IEEE, 532–539.
- [51] Yang Zhao, Kaihua Liu, Yongtao Ma, Zheng Gao, Yangguang Zang, and Jianfu Teng. 2017. Similarity analysis-based indoor localization algorithm with backscatter information of passive UHF RFID tags. *IEEE Sensors Journal* 17, 1 (2017), 185–193.
- [52] Han Zou, Zhenghua Chen, Hao Jiang, Lihua Xie, and Costas Spanos. 2017. Accurate indoor localization and tracking using mobile phone inertial sensors, WiFi and iBeacon. In *2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*. IEEE, 1–4.

Received May 2018; revised November 2018; accepted January 2019