MPiLoc: Self-Calibrating Multi-Floor Indoor Localization Exploiting Participatory Sensing

Chengwen Luo, Hande Hong, Mun Choon Chan, *Member, IEEE*, Jianqiang Li, Xinglin Zhang, *Member, IEEE*, and Zhong Ming

Abstract—While location is one of the most important context information in mobile and pervasive computing, large-scale deployment of indoor localization system remains elusive. In this work, we propose MPiLoc, a multi-floor indoor localization system that utilizes data contributed by smartphone users through participatory sensing for automatic floor plan and radio map construction. Our system does not require manual calibration, prior knowledge, or infrastructure support. The key novelty of MPiLoc is that it clusters and merges walking trajectories annotated with sensor and signal strengths to derive a map of walking paths annotated with radio signal strengths in multi-floor indoor environments. We evaluate MPiLoc over five different indoor areas. Evaluation shows that our system can derive indoor maps for various indoor environments in multi-floor settings and achieve an average localization error of 1.82 m.

Index Terms—Multi-floor indoor localization, participatory sensing, self-calibrating

1 Introduction

LOCATION is one of the most important types of context information in mobile and ubiquitous computing. Recently, indoor localization has been the focus of extensive research efforts [1], [2], [3], [4], [5], [6], [7], [8], due to both the need for indoor support of location-based services, and the unavailability of GPS in indoor environment. However, despite significant research progress, developing an indoor localization system that can be easily deployed on a large scale remains a challenge.

Two major obstacles hinder the large-scale deployment of such systems: (1) *Labor-intensive site surveys and system maintenance*: Many of these systems involve a dedicated offline calibration stage to build a radio map for the target location. The calibration requires the manual association of each location with its corresponding fingerprints, and needs to be repeated for any new location. Furthermore, the radio map needs to be periodically updated to reflect the environmental dynamics. These dedicated and time-consuming calibration and maintenance efforts thus make these systems less practical for large-scale deployment. (2) *Lack of accurate floor plans*: Recent research developments [2], [4] have shown that the calibration effort can be reduced with the prior knowledge of accurate floor plans of the places being measured. However, accurate floor plans are often not easily available.

- *C. Luo, J. Li, and Z. Ming are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. E-mail: {chengwen, lijq, mingz}@szu.edu.cn.*
- *H. Hong and M.C. Chan are with the School of Computing, National University of Singapore, Singapore 117417. E-mail: {honghand, chanmc}@comp.nus.edu.sg.*
- *X. Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: zhxlinse@gmail.com.*

Manuscript received 6 May 2016; revised 11 Apr. 2017; accepted 17 Apr. 2017. Date of publication 27 Apr. 2017; date of current version 1 Dec. 2017. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2017.2698453

In this work, we attempt to answer the following question: *can we design an indoor localization system that can automatic calibrates itself in unknown indoor environments?* Such a system should meet the following design goals. First, the system should not require specialized infrastructure support or prior knowledge of the environment such as floor plans and locations of wireless Access Points (APs). Second, there should not be a need for an expensive manual-calibration or site-survey stage. Third, the system should be able to automatically adapt to environmental changes and require minimal maintenance effort.

In this paper, we propose *MPiLoc*, a multi-floor-enabled indoor localization system that calibrates itself through participatory sensing. By utilizing the opportunistically collected data contributed by smartphone users, MPiLoc requires no prior knowledge about any building or any user intervention in both the calibration and maintenance stages. It adopts a novel trajectory matching and floor plan construction algorithm to automatically cluster, filter, and merge all user inputs to automatically construct indoor floor plans for multi-floor environments. Most importantly, radio maps required for localization are also automatically built and updated in this process. As a result, floor plans and radio maps that are essential to multi-floor indoor localization are automatically built and constructed. To achieve this, MPiLoc requires no special purpose hardware, the only assumption in MPiLoc is the availability of a WiFi infrastructure.

We implemented MPiLoc and evaluated it in five different buildings covering a total of about 5800 m^2. The implementation consists of a mobile client that track user's walking trajectories and upload to a server once sufficient data has been collected, and a server program that merges all user uploads to generate the indoor floor plan and radio map. The server also handles localization queries from user and returns the location based on the radio map constructed.

The rest of the paper is organized as follows. We present related works in Section 2 and in Section 3 the overview of MPiLoc. We then describe data collection method in Section 4.

Section 5 includes various trajectory clustering schemes (AP clustering, floor clustering and path segment clustering). We present trajectory matching in Section 6, floor plan and radio map construction in Section 7. Section 8 presents the energy management and Section 9 the evaluation results. We discuss applications enabled and system limitations in Section 10, and finally conclude in Section 11.

2 Related Work

2.1 Indoor Localization Systems

2.1.1 Infrastructure-Based

These systems rely on special-purpose infrastructures deployed to locate the target device. Cricket [9] uses radio and acoustic transmission and exploits Time Difference of Arrival (TDoA) in the signals. Recent developments explore multiple-input, multiple-output (MIMO) techniques using commodity APs and Angle of Arrival (AoA) to provide fine-grained localization [10]. While these techniques provide centimeter-level accuracy [9], [10], the need for special-purpose infrastructure and high deployment cost hinder their large-scale deployment.

2.1.2 Fingerprint-Based

A significant portion of research work in indoor localization exploits RF signal strengths and is known as the fingerprint-based approach. Most of these works use WiFi Receive Signal Strength (RSS) as the fingerprint [7], [14], [15], [16]. Recent work proposes other forms of fingerprints such as FM Radio [17] and physical layer information [18]. SurroundSense [19] generalizes the concept of fingerprint and explores ambient information such as noise, light color, etc. Fingerprint-based techniques suffer from high calibration cost since a labor-intensive site-survey process is typically required to construct the radio map for each indoor location. The static radio map is also vulnerable to environmental dynamics, resulting in high maintenance effort. MPiLoc aims to eliminate these overheads.

2.1.3 Propagation Model-Based

In trying to reduce the calibration effort, some researchers propose the signal propagation model-based technique to estimate the RSS value at a given location based on the theoretic model instead of manually tagging [20]. One popular model is log-distance path loss (LDPL) [20], which estimates RSS value based on the propagation distances. RADAR [14] provides a model-based approach to estimate the RSS value based on the AP locations and floor plans. EZ [20] further improves it and only needs to measure the signal strength at a few locations. Compared with the fingerprint-based techniques, model-based techniques typically reduce calibration effort at the cost of reduced accuracy. For most of these systems, AP locations or accurate floor plans need to be given.

2.1.4 SLAM-Based

Simultaneous Localization and Mapping (SLAM) techniques [21], [22] have been extensively studied by researchers recently [23], [24]. SLAM relies on landmarks detected by camera, laser or other ranging sensors and accurate controlled movement of mobile devices. Several work has been proposed to leverage the idea of SLAM by combing WiFi and IMU sensors on smartphones. Zee [4] exploits dead-reckoning and infers location according to the constraints imposed by floor plans. UnLoc [3] further exploits dead reckoning and learns indoor landmarks that exist in the environment to aid localization and requires at least one ground truth location of the landmark. MPiLoc is different in that knowledge of the indoor environment or sophisticated ranging sensors are not available in the system. In fact, MPiLoc requires no prior knowledge of the indoor environment or landmarks, and combines inertial sensing of smartphones to collect, cluster, match, and merge user contributed data through participatory sensing to generate and update floor plans and radio maps for multi-floor indoor localization.

2.2 Multi-Floor Localization

Recently, several research work has been proposed to tackle the problem of multi-floor localization [25], [26], [27], [28], [29]. [25] relies on delicately trained floor models basing on WiFi fingerprints to identify floor numbers. [26] combines pedestrian dead reckoning and particle filter to provide localization in multi-floor environment. [27] proposes a barometer-based calibration approach to provide elevation measurement on smartphones and achieves an average error of less than 5 meters. [28], [29] exploit barometer for floor detection and are closest to MPiLoc. However, they either require Bluetooth or synchronization in floor transition patterns (e.g., taking elevator at the same time) to detect encounter events for barometer calibration. MPiLoc, on the other hand, relaxes such assumptions and uses floor clustering algorithms to divide the trajectories into floor clusters. In addition, the floor clustering module seamlessly integrates with other modules in MPiLoc to extend the system from two-dimensional to three-dimensional.

2.3 Indoor Floor Plan Construction

Recently several research work has been focusing on automatic indoor floor plan construction and crowdsourcing map based localization [30]. CrowdInside [11] proposes an inertial sensing based system to automatically reconstruct the indoor floor plans with the assumption that the starting points of each trajectory in the indoor environment are known. Jigsaw [12] exploits both inertial sensing and vision-based approach to reconstruct the indoor floor plans. Similarly, IndoorCrowd2D [13] also exploits vision-based information for indoor scene reconstruction using smartphone-empowered crowdsourcing. Walkie-Markie [5] proposes an algorithm to map pathways using WiFi-Marks, which measures the tipping point in the signal strengths. MPiLoc is different from all these approaches in that it requires no prior knowledge of the environment (e.g., floor plans, starting points, landmarks, etc.), and automatically generates floor plans and radio maps for localization in multi-floor indoor environments. The signal correlation in MPiLoc extends the single tipping point matching in Walkie-Markie to whole spectrum matching, and extends single AP matching to multiple AP matching, which results in higher performance in floor plan construction as shown in Section 9. Table 1 summarizes the key differences between MPiLoc and these related work.

3 MPiLoc Overview

The MPiLoc architecture is shown in Fig. 1. MPiLoc exploits crowdsourcing to capture user walking trajectories using Inertial Measurement Unit (IMU) sensors equipped in the

TABLE 1
Comparison between MPiLoc and Recent Work on Indoor Floor Plan Construction

System	Floor Plan	Approach	Assumption
CrowdInside [11]	Single-Floor	IMU+GPS+Magnetic	Starting points of all trajectories can be inferred using GPS, anchor points can be frequently detected to reset errors.
Jigsaw [12]	Single-Floor	IMU+Vision	Participants take images indoor.
IndoorCrowd2D [13]	Single-Floor	IMU+Vision	Participants take images indoor.
Walkie-Markie [5]	Single-Floor	IMU+WiFi	WiFi-Marks can be detected robustly.
MPiLoc	Multi-Floor	IMU+WiFi+Barometer	No prior knowledge required and no explicit actions required from participants, exploits wireless spectrum similarity for automatic clustering, matching, and floor plan construction.

smartphones. To enable localization, it is required that one or more users carrying smartphones with the data collection application enabled to walk on various parts of the indoor area to be localized, and upload the annotated walking trajectories. An annotated walking trajectory consists of discrete walking steps, which further consists of IMU and barometer sensor readings, and the WiFi fingerprints associated with the steps. These user-contributed walking trajectories are used as inputs to construct or update the floor plan of the area covered by user movements.

The key challenge in MPiLoc is to combine these user-generated trajectories into multi-floor floor plans for localization. There are three main components involved. First, a clustering algorithm that uses AP signal strength and movement vectors is used to separate these walking trajectories into disjointed sets that cover different indoor floors and environments. The second component takes these disjointed segments and finds segments that cover the same walking paths based on the similarity of movement vectors and AP signals. Finally, in the third component, the system merges multiple trajectories to build floor plans. In the following sections, we present details of these three components.

4 Data Collection

4.1 Fingerprint Collection

MPiLoc opportunistically collects users' walking trajectories $T = \{\tau_i, i = 1, 2, \ldots, m\}$. Each walking trajectory τ_i is determined by two stationary points detected by the phone's accelerometer. $\tau_i = \{s_1, s_2, \ldots, s_n\}$, in which s_i is a discrete walking step detected. In MPiLoc, the Normalized Autocorrelation based Step Counting (NASC) approach used in Zee [4] is adopted for accurate step counting. The direction of the linear acceleration vector captured by the Android API [31] measures the walking direction in the phone's local x-y-z coordinate. However, since the phones are usually placed with arbitrary orientation, the linear acceleration vector needs to be multiplied by the inverse of the rotation matrices of the phone to obtain the phone acceleration in the world's East-North-Up (E-N-U) coordinate system [32]. The projected acceleration vector in the East-North plane represents the horizontal moving direction of the phone in the world coordinate, which directly reflects the heading angle of the user with respect to the earth North [32]. A Low-pass Filter (LPF) is then applied on the heading angles of consecutive steps for noise reduction. To cope with the stride length diversity, MPiLoc adopts the assumption in Zee [4] that 10 percent variations generally exist in normal human stride length, and uses a uniformly distributed random variable which added to the stride length to capture the estimation error. Besides stride length and heading direction, barometer readings and WiFi fingerprints are also collected between every two consecutive steps, and are automatically associated with each step. Therefore, each step $s_i = \{ID_i, x_i, y_i, f_i, B_i\}$ consists of five elements, global step identifier ID_i, horizontal displacement x_i, vertical displacement y_i, WiFi fingerprints f_i, and barometer reading B_i. 2D displacements x_i and y_i are calculated based on the heading angles and stride lengths. Fingerprints $f_i = \{r_1, r_2, \ldots, r_k\}$ represents the WiFi RSS measured at step i, where r_j is the received signal strength of the detected AP_j.

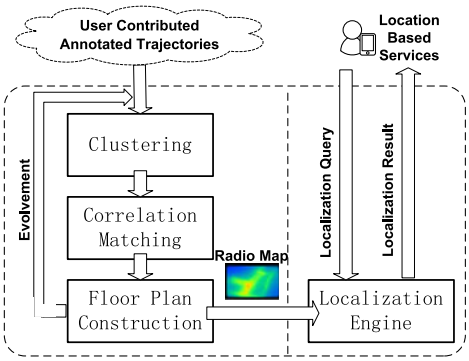


Fig. 1. Overview of MPiLoc.

4.2 Coping with the Noise of Dead Reckoning

One significant challenge with smartphone dead reckoning is the cumulated error over time. Dead reckoning can only be used to track the user for a short period of time, otherwise errors will need to be corrected frequently. Systems like Zee [4] or UnLoc [3] use the knowledge of indoor floor plans or landmarks to correct the errors. In MPiLoc, however, no prior knowledge about the indoor environment is assumed, making it especially challenging to align and merge different user contributed trajectories collected through smartphone dead reckoning. Although several work has been conducted to improve the accuracy of dead-reckoning with arbitrary phone placements [4], [33], the noise of step detection and heading angle detection still constitutes the major error source of dead reckoning based localization systems. For example, if the phone is strongly held in the hands, no steps can be detected since the accelerometer does not show periodic bumpy patterns.

To reduce the impact of dead reckoning errors, MPiLoc incorporates the following three different noise filtering and cancellation mechanisms. (1) MPiLoc is designed as a fully *opportunistic* system. That is, MPiLoc opportunistically detects "good" traces and filters out "bad" traces before

merging. To achieve this, stationary and fluctuation detection are used to filter out noisy traces. For example, when accelerometer does not show periodic bumpy patterns, no walking steps can be detected and user is detected to be stationary (Section 8.2.1). Therefore such stationary events have minimal impacts on MPiLoc since the traces are not used for merging. Similarly, MPiLoc detects heading angle noises (Section 8.2.2) and filters out the traces when the phone is placed at locations such as loose bags or pockets when the heading angle fluctuates during walking. As a result, only the walking trajectories with regular bumpy patterns and consistent heading angles are admitted to the system for merging. (2) In the merging process, the filtered traces are merged with each other through intra-trajectory merging and inter-trajectory merging, in which variances of heading angle estimations are canceled with each other (Figs. 10 and 11). (3) MPiLoc uses path correlation and shape correlation to match trajectories that have similar shapes and similar signal evolution patterns, as a result, those walking trajectories with abnormal orientations, shapes, and signal evolution patterns are not likely to be matched with the rest of trajectories, and will be discarded after the merging process. While more sophisticated error correction algorithms can be integrated with MPiLoc's inertial sensing module, the noise filtering mechanisms of MPiLoc significantly reduces the impact of dead reckoning error. We will provide the details in the rest of sections.

5 Trajectory Clustering

5.1 AP Clustering

As data collected from different users cover different parts of the indoor space, it is necessary to perform an initial level of data clustering to group the data into smaller, related groups. Given n trajectories inputs from all participating users, the AP clustering finds a clustering with l clusters $C = \{c_1, c_2, \ldots, c_l\}$, such that

$$\forall i \forall j \; APSet(c_i) \cap APSet(c_j) = \emptyset, \; 1 \leq i \neq j \leq l, \qquad (1)$$

in which $APSet(c_i)$ returns the set of all APs that appear in at least one of the fingerprints in the trajectories of cluster c_i. AP clustering therefore separates trajectories collected in different indoor environments that have different sets of APs into different clusters. Though AP clustering only provides building-level granularity, the light-weighted clustering is still an important technique to efficiently categorize the trajectory data once the system is deployed at scale.

5.2 Floor Clustering

5.2.1 Floor Transition Detection

To achieve floor-level clustering, barometer reading is annotated with each step. The absolute reading of the barometer can show variations across different devices and even due to different weather conditions [34]. As a result, in MPiLoc we use the relative reading instead of the absolute reading for accurate floor transition detection. Fig. 2 shows how the altitude reported by the barometer changes when the user takes stairs and an elevator. We can observe a marked change in height when the user is traveling up and down the stairs and elevators. We use this observation as the basis for accurate floor-transition detection in MPiLoc. To detect the floor transition, we maintain a sliding window of altitude values corresponding to steps taken by the user. For every new step taken

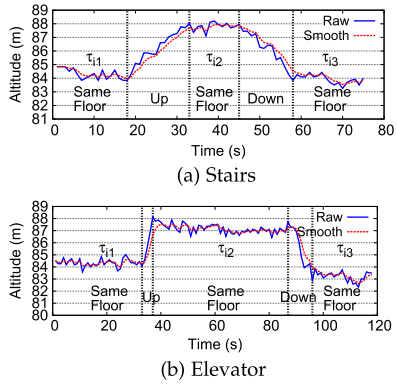


Fig. 2. Altitude changes during different floor transition events. Floor transition separates trajectory τ into different floor segments.

by the user, we sample the barometer height and advance the sliding window by 1 step. If the difference in height between the end and start of the sliding window exceeds a threshold, then we mark the event as a floor transition.

As illustrated in Fig. 2, the floor transition splits each trajectory τ_i into different floor segments $\{\tau_{i1}, \tau_{i2}, ..., \tau_{ik}\}$ if k-1 floor transitions are detected. To generate segments that cover only one single floor, we discard parts of the trajectories during which the sliding window reports floor transitions. Since the absolute altitude value reported by the barometer is not accurate, we do not know the exact floor from which the floor segments are collected. And only know that the consecutive floor segments are taken from two different floors. For example, if the floor transition detection algorithm reports that τ_{i1} has a mean altitude smaller than τ_{i2}, then a floor transition constraint $\tau_{i1} \rightarrow \tau_{i2}$ is detected, which indicates that τ_{i2} is collected from a higher floor than that of τ_{i1}. The floor transitions impose constraints on the floor-level clustering. We will show how we leverage this information to achieve floor-level clustering next.

5.2.2 Floor-Level Clustering

To cluster the collected trajectories into floor-based groups, we first need a similarity measurement for different trajectories. The similarity should be high for those collected from the same floor, and lower otherwise. Since the trajectories contributed by users are annotated with WiFi fingerprints during data collection, the floor-level similarity can be measured using the wireless signals. Different floors usually have different sets of WiFi access points. Even though there might be some overlapping in the AP sets, their signal strengths vary. For two trajectories $\tau_1 = \{s_1, s_2, \ldots, s_n\}$ and $\tau_2 = \{s_1, s_2, \ldots, s_m\}$, the floor similarity $Sim_f(\tau_1, \tau_2)$ is defined as

$$Sim_f(\tau_1, \tau_2) = \sum_{i=1}^{n} \sum_{j=1}^{m} Sim_s(s_i, s_j)/mn, \qquad (2)$$

where s_i and s_j are annotated steps in τ_1 and τ_2 respectively, and $Sim_s(s_i, s_j)$ is the step similarity measured by the Tanimoto Coefficient [35]

$$Sim_s(s_i, s_j) = \frac{f_i \cdot f_j}{\|f_i\|^2 + \|f_j\|^2 - f_i \cdot f_j}. \qquad (3)$$

Here f_i and f_j are fingerprints annotated to steps s_i and s_j respectively as described previously. The step similarity $Sim_s(s_i, s_j)$ ranges from 0 to 1. The final output of floor similarity Sim_f combining all step similarities becomes the

similarity metric between two trajectories and falls between 0 and 1 as well. If two trajectories have high floor similarity, they are more likely to be collected from the same floor.

To illustrate the floor-level clustering process, consider a sample AP Cluster $c = \{\tau_1, \tau_2, \ldots, \tau_{10}\}$ containing 10 trajectories. We do not know the exact floors from which they are collected and these trajectories in the same AP cluster might cover multiple floors. Based on the floor transition detection described in the previous section, we are able to detect those trajectories containing floor transitions. For example, if we have found a subset of five trajectories c'=$\{\tau_1, \tau_2, \ldots, \tau_5\}$ such that each trajectory in c' contains floor transition events, the floor transition detection will segment c' into another set $c'' = \{\tau_{11}, \tau_{12}, \tau_{21}, \tau_{22}, \ldots, \tau_{51}, \tau_{52}\}$ if each of the trajectory contains only one floor transition, such that each of the generated trajectory is collected from only one floor. Floor segmentation also generates a set of floor constraints $FC = \{\tau_{11} \rightarrow \tau_{12}, \tau_{21} \rightarrow \tau_{22}, \ldots, \tau_{51} \rightarrow \tau_{52}\}$ if each trajectory is going upstairs in this example. With the floor constrains we have, the goal of the floor-level clustering algorithm therefore is to group trajectories in c'' into different floor clusters, within which all trajectories are collected from the same floor.

Since the floor similarity between each pair of trajectories can be measured based on the wireless signal similarities using Equation (2), the clustering can be seen as a merging process to merge trajectories in c'' and generate disjointed floor clusters. Therefore the floor clustering can be modeled as the following optimization problem:

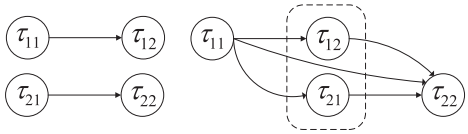
$$\begin{cases} maximize \ \sum_i \sum_j Sim_f(\tau_i, \tau_j) \\ s.t. \ Sim_f(\tau_i, \tau_j) > \sigma_0, \\ Floor \ constraint \ FC \end{cases} \quad (4)$$

where τ_i and τ_j are trajectories in c'' that are merged to the same floor cluster. The merging maximizes the sum of the floor similarities while ensuring that the floor constraint FC is not violated.

For each merged pair of trajectories, their floor similarity is ensured to be greater than the minimum similarity threshold σ_0. σ_0 can be learned from trajectories in c'' since each trajectory in c'' is collected from one single floor. To learn the average floor similarities for trajectories collected from the same floor, we split each trajectory in the c'' evenly and calculate the average inter-similarity between them using Equation (2). The minimum similarity σ_0 is taken to be the average floor similarity and we reject all those pairs with low similarities in the clustering.

The floor constraints FC represent the knowledge that certain pairs of trajectories belong to distinct floors. Due to the transitivity of the floor constraints, they need to be updated in the merging process once we merge two trajectories into the same floor cluster. Consider floor constraints $FC = \{\tau_{11} \rightarrow \tau_{12}, \tau_{21} \rightarrow \tau_{22}\}$. As illustrated by Fig. 3, if τ_{12} and τ_{21} are merged into the same floor cluster, the constraints need to be updated as $FC = \{\tau_{11} \rightarrow \tau_{12}, \tau_{21} \rightarrow \tau_{22}, \tau_{11} \rightarrow \tau_{21}, \tau_{11} \rightarrow \tau_{22}, \tau_{12} \rightarrow \tau_{22}\}$ due to their transitivity. The updating process must be performed whenever two trajectories are merged to the same floor.

The detailed floor clustering algorithm is described in Algorithm 1. For each AP cluster c, the floor clustering algorithm finds a set of floor clusters that cover different floors of the indoor environment covered by this AP cluster. The barometer-based floor transition detection first detects the floor transitions that are present in each walking trajectory and segments these trajectories to form c'', in which each trajectory only covers one particular floor. The segmentation also generates the initial set of floor constraints FC.



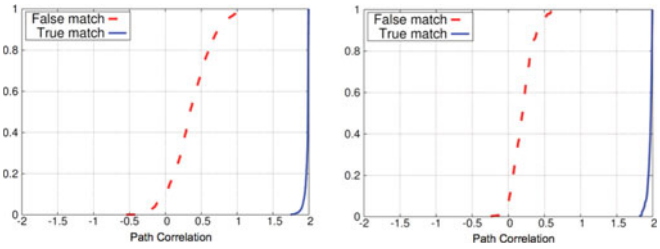
(a) Floor Constraint Before Merging τ_{12} and τ_{21} (b) Constraint Update After Merging τ_{12} and τ_{21}

Fig. 3. Floor constraint update in floor clustering.

Algorithm 1. Floor Clustering Algorithm

Input: AP cluster c
Output: Set of floor clusters $C_f = \{c_{f1}, c_{f2}, \ldots, c_{fk}\}$
Generate c'' with barometer-based floor transition detection and generate initial floor constraints FC;
Compute floor similarity $Sim_f(\tau_i, \tau_j)$ for each pair of trajectories τ_i and τ_j in c'' using Equation (2);
Sort pairs (τ_i, τ_j) in descending order based on $Sim_f(\tau_i, \tau_j)$;
for *each pair of* (τ_i, τ_j) **do**
  if $Sim_f(\tau_i, \tau_j) > \sigma_0$ **then**
    if $\tau_i \rightarrow \tau_j \notin FC$ && $\tau_j \rightarrow \tau_i \notin FC$ **then**
      if τ_i *or* τ_i *not in* C_f **then**
        Merge τ_i and τ_j to the same floor cluster in C_f;
        Update floor constraint FC;
      end
      else
        if *Clusters containing* τ_i *and* τ_j *can be merged based on* σ_0 *and* FC **then**
          Merge clusters containing τ_i and τ_j;
          Update floor constraint FC;
        end
      end
    end
  end
  else
    return C_f;
  end
end
return C_f;

To merge the trajectories in c'', each pair of trajectories is first sorted by their floor similarities in the descending order. Each time one pair of trajectories is picked from the top of the list. If their floor similarity is greater than σ_0 and they meet the floor constraints, they become candidates to be merged to the same floor cluster. If one of these two trajectories does not belong to any existing floor clusters, they are merged to the same floor cluster, and FC is also updated due to the transitivity of the floor constraints. However, if two trajectories already belong to different floor clusters, we need to ascertain whether these two clusters can be merged together. In MPiLoc, if the average floor similarity of these two clusters is greater than σ_0 and the merging will not cause any violation of the floor constraints, they are merged to the same floor cluster. Otherwise, we continue without updating the exiting floor clusters. The process is repeated until no such pair of trajectories can be found. The resultant clusters consist of disjointed groups of trajectories, and each cluster covers exactly one floor in this indoor environment.



(a) $3000m^2$ Office Floor (b) $120m^2$ Research Lab

Fig. 4. CDF of path correlation.

5.3 Path Segment Clustering

Within the same floor cluster, we further divide each single trajectory into disjointed path segments. While path segments can take any form in general, in this work, we consider only two kinds of path segments, namely: *turns* and *long straight lines*. Walking along a straight path and making corner turns are natural walking patterns in an indoor environment. A given trajectory $\tau = \{s_1, s_2, \ldots, s_n\}$ in each floor cluster c_f, can be broken into disjointed path segments (consisting of either turns and/or straight lines) $S = \{s_p, s_{p+1}, \ldots, s_q\}$ where $1 \leq p < q \leq n$. Each segment in the path segment clusters then becomes the basic unit for trajectory matching in the next stage.

6 Trajectory Matching

For each floor cluster, the goal of trajectory matching is to find those path segments that are collected from the same indoor space. For example, if two path segments are collected from the same corridor, they should be in the same floor cluster and should be marked as one pair of matched path segments. To achieve this goal, the path correlation and signal correlation are used in MPiLoc for matching.

6.1 Path Correlation

Like the clustering component, the trajectory matching algorithm follows a two-phase scheme. The first phase is based on a simple but effective idea. When people walk along the same segment (turns or straight lines), the evolutions of the two trajectories on a 2D plane should be highly correlated. The path correlation correction can be measured as

$$Corr_{path} = Corr_x(S_1, S_2) + Corr_y(S_1, S_2). \tag{5}$$

For two path segments from the same path segment cluster, $S_1 = \{s_1, s_2, \ldots, s_n\}$ and $S_2 = \{s'_1, s'_2, \ldots, s'_n\}$ with the same number of steps n, the Pearson correlation can be computed as

$$Corr_x(S_1, S_2) = \frac{E[(X_1 - \mu_{X_1})(X_2 - \mu_{X_2})]}{\sigma_{X_1}\sigma_{X_2}}, \tag{6}$$

where $X_1 = \{x_1, x_2, \ldots, x_n\}$ and $X_2 = \{x'_1, x'_2, \ldots, x'_n\}$ are sequence of horizontal displacements of steps of S_1 and S_2 respectively. Similarly, $Corr_y$ is the correlation of the vertical displacements of steps of S_1 and S_2. These displacements can be computed given the step distance and direction of movement. $Corr_{path}$ therefore measures the similarity between two walking paths in the 2D plane.

Fig. 4 shows the CDF of the path correlations for traces collected from both a large indoor floor level and a research lab. In both environments more than 90 percent of path correlations for correct matches (paths with same evolution trend in 2D plane) have value greater than 1.90 (maximum 2). The path correlation is much lower for incorrect matches, with 90 percent less than 0.75.

6.2 Signal Correlation

Path correlation alone is not sufficient for obtaining accurate matches. When path segments are collected from parallel corridors in the same building, these segments may have high path correlations. Another feature exploited in MPiLoc is changes in the RSS signal along the walking path. It is observed that there are specific patterns according to which RSS signal changes along the same path way. This change is due to the signal propagation and other environmental obstacles. The pattern in which RSS signal changes provide another useful hint to determine matching segments.

One question on using these signal measurements is the stability of their trends with respect to the changes in phone model and time. Fig. 6 shows the stability of WiFi signal trends on the same path across three different phone models (Samsung Galaxy S3, S4, and Galaxy Nexus). The trends are plotted with smoothed curves and are stable across different phone models for both APs. The variation is also relatively stable at different periods of the day. As shown in Fig. 7, the RSS trends collected for the same walking path in the morning (9 am), afternoon (1 pm) and in the night (10 pm) are also similar. Another observation is that the similarity for APs with higher RSS value tends to be higher than those with lower RSS values. As shown in Figs. 6 and 7, the trend detected for AP1 is more stable than that for AP2. With these observations, we use signal correlation as a metric to further measure the similarity between two path segments S_1 and S_2

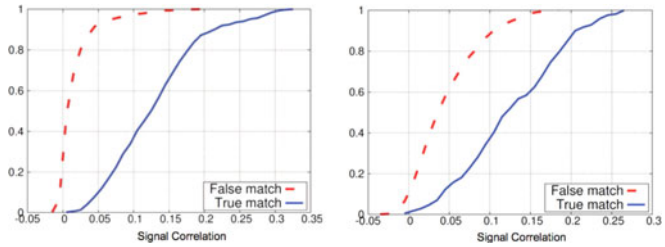
$$Corr_{signal} = \sum_i \omega_i \cdot Corr(R_1^i, R_2^i) \cdot I(R_1^i, R_2^i), \tag{7}$$

where $R_1^i = \{r_1, r_2, \ldots, r_n\}$ and $R_2^i = \{r'_1, r'_2, \ldots, r'_n\}$ are the sequences of RSS values of AP_i observed in S_1 and S_2 respectively. ω_i is the weight for AP_i and we set $\omega_i = \frac{2}{|\mu_{R_1^i} + \mu_{R_2^i}|}$. As signal strength values are given in negative terms (measured in dBm), APs with larger average RSS value will have more weight. $Corr(R_1^i, R_2^i)$ is the Pearson correlation of two RSS sequences for AP_i. $I(R_1^i, R_2^i)$ is an indicator function used to decide if an AP_i should be included in the computation

$$(R_1^i, R_2^i) = \begin{cases} 1, & |\mu_{R_1^i} - \mu_{R_2^i}| < \sigma_{RSS} \quad (8a) \\ 0, & otherwise, \quad (8b) \end{cases}$$

where σ_{RSS} is the maximum acceptable difference between the two mean RSS values of two path segments. The current value for σ_{RSS} is set to 5 dBm, which has been observed to work well for different environments. Similar to the path correlation computation, as movement can occur in both directions on the same path, we calculate the correlation for both the forward and reverse directions for each pair of segments and the maximum correlation is used. We exclude APs that appear only in one segment and not in the other, and also remove APs that appear in fewer than 10 steps in either of the two segments. In summary, for the signal correlation computation, we only consider APs that appear often enough in both segments and whose average signal strengths are similar.

In general, the $Corr_{signal}$ increases as two trajectory segments have more common APs and the trends of APs are



(a) $3000m^2$ Office Floor (b) $120m^2$ Research Lab

Fig. 5. CDF of signal correlation.

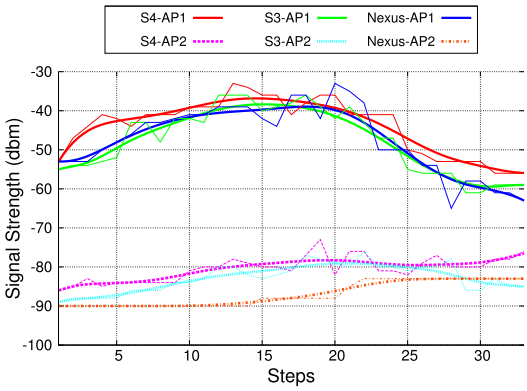


Fig. 6. Stability of signal trends (phone varying).

similar. Fig. 5 shows the signal correlation distribution for both the 3000 m^2 office floor and the 120 m^2 research lab. In both environments more than 42 percent of signal correlations for correct matches (same paths) have values greater than 0.15. The signal correlation is much lower for incorrect matches, with 98 percent less than 0.15.

6.3 Final Matching

MPiLoc leverages the discriminative power of both path and signal correlations in the final matching to achieve an accurate matching performance. For each pair of segments in the same path segment cluster, we use the turning points to align them and keep them to have the same number of steps by discarding extra steps that contained in the longer path segments. Fig. 8 shows the receiver operating characteristic (ROC) curve for both the large office floor and the small research lab. Both curves show high performances of matching, with large area under the curve. A good operating point can be chosen using the $y = x$ line. This operating point provides a guide for choosing the appropriate thresholds for the path and signal correlation values to be used for matching.

7 Floor Plan Construction

7.1 Algorithm

The trajectory matching discussed in the previous section generates matching path segments for each floor cluster. The output of the matching process $\mathcal{M} = \{(S_1, S_2), \ldots, (S_i, S_j)\}$ contains pairs of matching path segments and is used as input to the Algorithm 2. Algorithm 2 each time merges and generates floor plans for trajectories collected from the same floor, i.e., in the same floor cluster.

Initialization. In the initialization phase, MPiLoc first builds a displacement matrix M_d, which represents the final relative locations between each pair of steps in trajectories in the same floor cluster. That is, M_d represents the final floor plan. Given

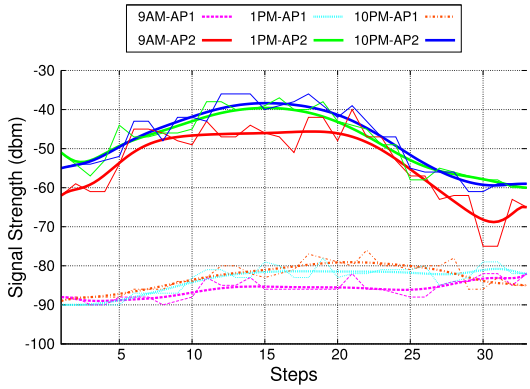


Fig. 7. Stability of signal trends (time varying).

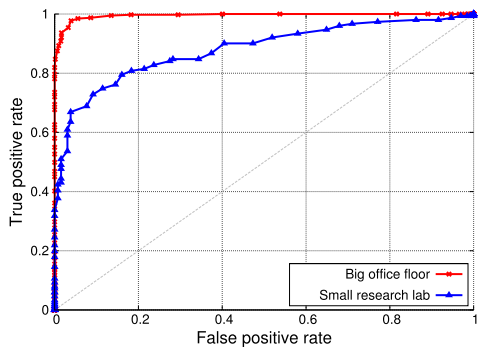


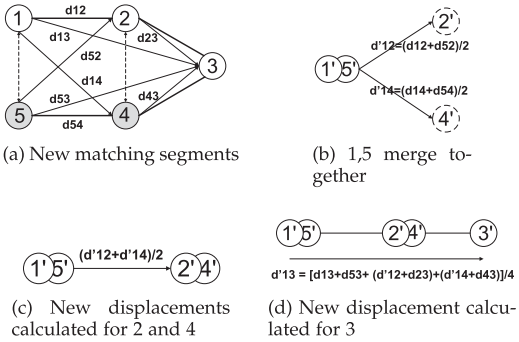
Fig. 8. ROC curve of final matching.

two steps with global ID i and j, the entry $M_d[i][j]$ determines the relative 2D displacement between the two steps in the floor plan, and are initially set to be their relative displacement in the trajectory. The displacement between two steps can only be measured if there are matching path segments that can relate them. The displacement is "undefined" if the steps are from two different trajectories with no matching path segments. S_{merge} stores the steps that are merged due to matching path segments and are initially set to be empty.

Algorithm 2. Floor Plan Construction Algorithm

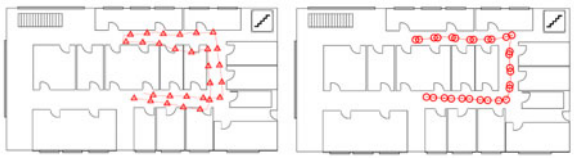
```
1:  Input: Matching output M, one floor cluster c_f
2:  Output: Updated displacement matrix M_d
3:  Initialized displacement matrix M_d;
4:  Set of merged steps S_merge is set to be empty;
5:  for each matching segment pair (S_i, S_j) in M do
6:    for each matching step pair (s_m, s_n) in (S_i, S_j) do
7:      Place s_m, s_n into the single location;
8:      New displacement of s_m and s_n are average displacements of s_m and s_n to all points in S_merge;
9:      S_merge = S_merge ⋃ s_n ⋃ s_m;
10:   end
11:   for each step p in c_f but not in S_merge do
12:     Displacement of p = average displacements of p to all points in S_merge;
13:   end
14:   Update displacement matrix M_d based on all new displacements calculated;
15: end
16: return M_d;
```

Iteration. In the iteration phase, each matching segment pair (S_i, S_j) is taken into account to update the displacement



(a) New matching segments (b) 1,5 merge together (c) New displacements calculated for 2 and 4 (d) New displacement calculated for 3

Fig. 9. Example of motion vector merging. d_{ij} denotes the current displacement and d'_{ij} denotes the new displacement.



(a) Before (b) After

Fig. 10. Intra trajectory merging and error correction.

matrix. Recall that matching segments have the same number of steps. For each pair of matching steps (s_m, s_n), since they are collected from the same indoor space, we set the starting position of these steps to be the same so that they start at the same location. We then compute the new displacements by finding the average displacements of these steps to those have already been merged. The detailed floor plan construction algorithm is shown in the Algorithm 2.

As an illustration, consider Fig. 9. The trajectory consists of five steps $\{1, 2, 3, 4, 5\}$. $S_1 = \{1, 2\}$ and $S_2 = \{5, 4\}$ are the only pair of matching segments in this example. The algorithm first computes the starting (relative) position of the first matching steps. Fig. 9a shows the original displacements of the points in the trajectory.

In Fig. 9b, the starting points of the first pair of matching steps $\{1, 5\}$ are considered to be at the same location (shown as $1'$ and $5'$ in the figure). In order to calculate the new displacements for the next pair of matching steps $\{2, 4\}$, which is again assumed to be collocated, the new displacements d'_{12} and d'_{14} are computed as $\frac{d_{12}+d_{52}+d_{14}+d_{54}}{4}$, as shown in Fig. 9c.

After the new displacements for all matching steps in this segment have been computed, the displacement of all the other steps are updated. As shown in Fig. 9d, the displacement d'_{13} is determined by averaging the displacements of all four matched steps.

Since the matching pair is either from the same trajectory or from different trajectories, the floor plan construction algorithm works for both intra-graph merging and inter-graphs merging. As shown in Fig. 10, the trajectory is refined internally and merged with itself using the algorithm. The error cumulated in dead-reckoning is corrected within the same trajectory. Fig. 11 shows the merging of different trajectories collected from the same floor. By merging trajectories, MPiLoc further cancels the dead-reckoning error through inter-trajectory merging. Note that since each step carries fingerprint data, it naturally can serve as the radio map for localization. Since the merging algorithm works for all geographically separated floor clusters, floor plans and radio maps are generated for all different indoor floors covered by the participating users.

(a) Before (b) After

Fig. 11. Inter trajectory merging and error correction.

The maps generated are relative maps, i.e., the locations in the map are not associated with the absolute location yet. To map the floor plan to the real locations in the indoor environment, MPiLoc only requires at least one point to be associated with a physical coordinate. This point becomes a global reference point set manually or detected by GPS opportunistically and after that all the locations of the rest points in the maps can be fixed.

7.2 Floor Plan Filtering

Filtering is required to remove the noisy samples and trajectories in the floor plan construction process. Trajectories that have no matching segments are first filtered out after the matching process. Therefore the outlier trajectories will not be reflected in the final results. To further smooth the constructed floor plans, we adopt a grid-based filtering scheme. The generated floor plans are divided into 1×1 m^2 grids. We observed that most grids that contain correct walking trajectories have more steps than the average number of steps over all grids in the floor plans generated by the trajectory merging algorithm. In the final floor plan constructed, all grids with number of steps less than the averaged will be removed. To smooth the floor plan constructed, morphological operators dilation and erosion [36] are used, and the extracted contour from the erosion result are used as the smoothed walking paths.

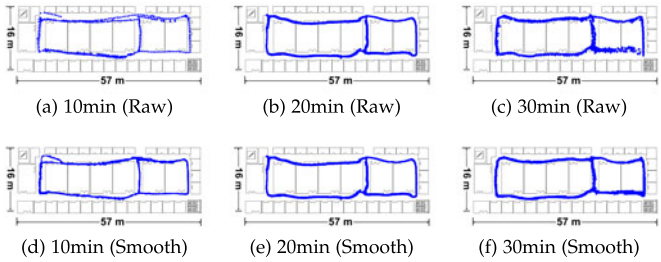
7.3 Floor Plan Evolution

To reflect the environmental changes and new user inputs, the floor plan generated needs to be periodically updated. One important feature of MPiLoc is the floor plans will keep evolving with continuous incoming of user inputs. The evolution is also fully automatic. In MPiLoc, the floor plan is updated every 10 minutes to handle the new user input. All new data will be clustered into the existing clusters or new clusters (e.g., new floors) may be generated. As shown in Fig. 12, the floor plan is updated every 10 minutes to generate an evolving indoor map. The radio maps are also updated during the same process to maintain an up-to-date localization database.

7.4 MPiLoc Localization

MPiLoc adopts a fingerprint-based approach for indoor localization. The radio maps are automatically built and updated by merging user contributed walking data. In this way, MPiLoc is able to handle localization queries and return the user location using the radio map and input fingerprints. Previous works such as RADAR [14] utilize the fingerprint database by using the nearest neighbors from the query point to the reference points in the database as the similarity metric.

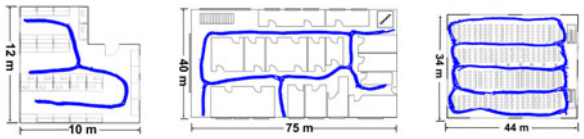
Such an approach works relatively well for indoor areas with sparse AP deployments (In RADAR, only three APs are presented). However, during our data collection we observe that many indoor environments have very dense



(a) 10min (Raw) (b) 20min (Raw) (c) 30min (Raw)

(d) 10min (Smooth) (e) 20min (Smooth) (f) 30min (Smooth)

Fig. 12. Floor plan evolution in floor plan construction.



(a) Research Lab (b) Office Floor (c) Library

Fig. 13. Floor plan construction for various indoor environments.

AP deployments (sometimes more than 100 on one floor). Nearest neighbor matching works poorly in the dense AP environment. This is because at each location smartphones can observe a long list of remote APs with RSS ranging from -80 dbm to -90 dbm. The RSS fluctuations of large numbers of these remote APs overwhelm the small set of nearby APs in calculating the similarity. However, nearby APs are more important in deciding the current location of the user since high RSS values only cover a small area for each AP. Based on this observation, MPiLoc uses the simple but more effective *weighted maximum similarity* as the metric

$$WMS = \sum_{i=1}^{n} \omega_i' \cdot \frac{1}{max\{|r_i - r_i'|, 1\}}, \quad (9)$$

where n is the total number of APs, $\omega_i' = 1/|\mu_i|$ is the weight of the ith AP and is inverse to the absolute of its mean value. Therefore, nearby APs with higher average RSS values will have higher weights. r_i is the input RSS of AP_i and r_i' is from the radio map. WMS will have a higher value if the input point and reference have more common APs and the RSS differences for nearby APs are smaller. The location will be determined by the maximum WMS matching in the radio map.

8 Energy Management

8.1 WiFi Scanning Modes

8.1.1 Collection Mode

During data collection, it is important to increase the collected fingerprint density when users are walking indoors. To increase the fingerprint sampling rate, in MPiLoc we only scan Channels 1 (2,412 MHz), 6 (2,437 MHz) and, 11 (2,462 MHz) during data collection. These channels do not overlap with the commonly deployed 802.11 b/g/n [37] network. The *iw* wireless package for Android is used [38]. As shown in Fig. 14, these three channels cover most of the deployed APs in the environment we measured. In our scan, we also include one channel (5,240 MHz) from the less commonly deployed 802.11a network. By reducing the number of channels scanned and improving the efficiency of the code, we significantly increase the sampling rate. On the average, around three radio fingerprints can be collected every second, compared with using the Android

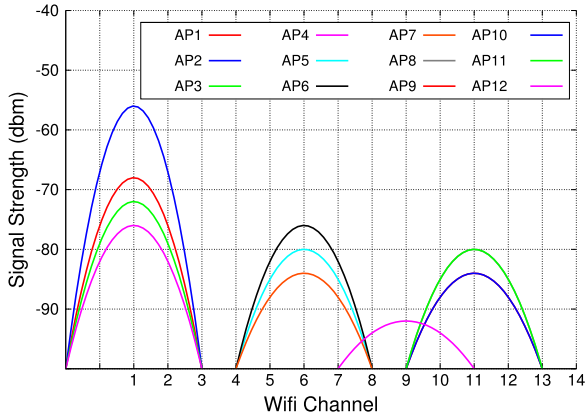


Fig. 14. MPiLoc increases scanning speed by scanning sub-channels.

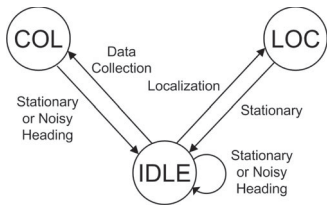


Fig. 15. Sensor-triggered scanning for energy management.

WifiManager which can only collect one sample every three two to three seconds. The average number of fingerprints per step is computed by combining all fingerprints collected between two consecutive steps. However, the aggressive sampling also increases the energy consumption, and should to be performed only when necessary. We will discuss the sensor-triggered WiFi scanning scheme in Section 8.2.

8.1.2 Localization Mode

During online localization, the system becomes less sensitive to the WiFi sampling speed, and a two-to-three second WiFi refreshing rate is normally sufficient for most applications to achieve the 'real-time' localization. As a result, it is no longer necessary to sacrifice energy to WiFi sampling speed, and so we use the normal Android WiFiManager scanning for online localization.

8.2 Sensor-Triggered WiFi Scanning

To further reduce the power consumption of WiFi scanning, we exploit smartphone sensors to differentiate between different system states to switch the scanning mode dynamically. As shown in Fig. 15. MPiLoc runs in three scanning states: COL, LOC, and IDLE. In the COL state, MPiLoc performs data collection and uses the fast scanning described in Section 8.1.1 to collect fingerprints as fast as possible. In the LOC state, MPiLoc performs localizations tasks and uses the normal Android WiFiManager scanning to reduce the sampling cost. In the IDLE state, MPiLoc only samples the low-cost IMU sensors and stops all WiFi scanning to save energy.

8.2.1 Stationary Detection

During opportunistic data collection, since there is no control on their walking patterns, participants may stop occasionally and WiFi scanning will obtain duplicated fingerprints for the same location. Similarly, during localization it becomes unnecessary to refresh the locations when

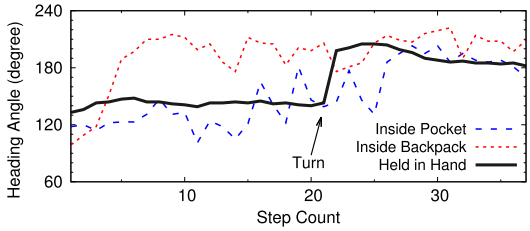


Fig. 16. Heading noise detection in MPiLoc.

users are staying at the same locations. To save power, it is important to reduce the WiFi sampling rate or stop WiFi scanning to avoid collecting redundant fingerprints for the same location. In MPiLoc, since the system detects walking steps, the user is deemed stationary if the step counter is not updated for a given amount of time. In MPiLoc, this period is set as 10 seconds. Users are determined to be stationary if no steps are detected after timer expires.

8.2.2 Heading Noise Detection

Heading angle estimation using IMU sensors can be noisy [39] and the noise of heading angles calculated using smartphone IMU sensors constitutes the major error source of the system. In addition, users might put their phones in different places during data collection, for example holding the phones in their hands, or putting them in pockets or backpacks. Although the trajectory merging provides error correction for dead-reckoning as described in Section 7, it is important to filter out noisy compass readings before uploading them for merging. As shown in Fig. 16, putting phones inside loose pockets or backpacks introduces more heading angle fluctuations than when users are holding the smartphones in their hands during data collection. Detecting such noisy traces not only avoids adding additional noise to the trajectory merging process, but also provides important hints to the smartphones to switch to a low-power state to save energy.

In MPiLoc, we opportunistically capture traces with consistent heading angles and discard the rest with noisy heading angles. We measure the smoothness of the heading angles using the Hodrick-Prescott filter[40] to detect the level of fluctuation of the heading angles when walking

$$Smoothness = \sum_{i=3}^{n} (\alpha_i - 2\alpha_{i-1} + \alpha_{i-2})^2, \tag{10}$$

where α_i is the heading angle sampled at the ith step. We maintain n as 10 steps and report heading noise when it exceeds an empirical threshold. The heading noise detection also triggers the smartphone to switch from the COL state to the IDLE state to save power.

8.2.3 Triggered Scanning

As shown in Fig. 15, during data collection the smartphone will transit from the IDLE state to the COL state when the user is walking and the compass readings are not fluctuating, and will switch back to the IDLE state either when the user is detected to be stationary, or when noisy heading angles are detected. Similarly, during localization the phone will switch to the LOC state from the IDLE state when the users is walking normally and switch back to the IDLE state when the user stops walking. The detailed energy consumption of different states and the final triggered scanning scheme is evaluated in Section 9.

9 Evaluation

9.1 Implementation

MPiLoc has both client and server components. The client performs two functions: *data collection* and *issues localization query*. For data collection, the client runs an Android smartphone service in the background to opportunistically collect walking trajectories and radio fingerprints. For localization, the client issues queries to the server to localize the phone. The server collects user uploaded trajectory and fingerprint data. It uses the data collected to construct and update the floor plans periodically for all indoor environments it has data for. For each localization query, the server first determines the correct radio map to use based on the AP clustering result. The weighted maximum similarity match is then used to find the best matching location of the phone.

9.2 Data

The experiment data is collected from five different areas which cover about 5800 m^2 areas in total. The sizes of these five different floors range from 120 m^2 to 3000 m^2. Three different phone models are used: Google Galaxy Nexus, Samsung S3 and Samsung S4. All phones run the Android OS. An average of 37 APs are detected in each of the 5 areas. In total, 700 user trajectories from 20 participating students are recorded, which contain about 100,000 steps, with each step associated with direction as well as WiFi fingerprints. We do not have special requirements on phone placement during data collection. The participants may hold the phone in their hands, put inside pockets or backpacks. The heading noise detection discussed in Section 8.2.2 is used to filter out walking trajectories with fluctuating heading angles. To obtain the ground truth for evaluation, we ask them to periodically tap their locations on the map so that the corresponding steps' ground truth locations in the indoor environment are known.

9.3 Performance

9.3.1 Evaluation Metrics

We evaluate the overall performance of MPiLoc by evaluating the quality of the floor plan constructed and the localization accuracy. Two major metrics are used in the measurement of the floor plan construction and localization:

Step Mapping Error (SME). The floor plan constructed maps steps of walking trajectories into the real floor plan. The step mapping error measures how accurately the trajectories fit the real floor plan. Since fingerprints are associated with each step, a lower step mapping error results in higher fingerprint mapping accuracy, which directly affects the localization accuracy. The SME is defined as

$$SME = \|L(s) - L(s')\|, \tag{11}$$

where $L(s)$, $L(s')$ are the mapped location of the step and the ground truth location of the step respectively. A smaller SME reflects better matching of the constructed floor plan to the real one. To establish the ground truth, the locations where each step is taken in the reference floor plan are manually tagged. Since each step has a globally unique identifier, the location of one particular step in the constructed floor plan can be obtained by querying the ID, and SMEs are measured by calculating the differences between the estimated step locations and their respective ground truth locations.

TABLE 2
Performance of Barometer-Based Floor Transition Detection

	Morning	Afternoon	Evening
Precision	100%	100%	96%
Recall	97.5%	98%	98%

TABLE 3
Floor Clustering Performance

Precision	Recall	Accuracy
95.2%	88.9%	97.1%

Localization Error (LLE). LLE measures how well the location given by the localization server matches the ground truth location of the phone

$$LLE = \|L(p) - L(p')\|, \tag{12}$$

where $L(p)$ is the estimated location and $L(p')$ is the real location of the phone. The smaller the euclidean distance, the better the localization quality.

9.3.2 Trajectory Clustering

The clustering algorithms in MPiLoc group user-contributed data into smaller groups for higher efficiency in the later stage of the floor plan construction process. Since the major uncertainty in the whole clustering process lies in the floor clustering process, we focus on the evaluation of floor clustering here.

Table 2 shows the evaluation results for the barometer-based floor transition detection. The ground truth is input by the user whenever a floor transition happens when taking stairs or elevators. The collected time is also recorded down due to barometer readings might vary over time. We group our data into different time periods. We can see that even the barometer is sampled at a low sampling rate (1 Hz), the floor transition can be accurately detected in all datasets. Since we use the relative altitude value instead of the absolute value for floor transition detection in MPiLoc, the accuracy remains high in all scenarios regardless of the fact that they are collected in different time periods. Floor transition detection using relative barometer reading has above 96 percent precision, and above 97 percent recall. The relative altitude based floor transition detection in MPiLoc makes it possible for robust detection from large quantities of input data that are collected from different users on different days.

To evaluate the floor clustering performance, we evaluate the quality of all generated floor clusters. If two trajectories clustered to the same floor cluster are actually from the same floor, this results in a true positive (TP), otherwise it is a false positive (FP). If the clustering algorithm group two trajectories from the same floor into different floor clusters, and it is a false negative (FN), similarly for true negative (TN). In this way, we have $precison = TP/(TP + FP)$, $recall = TP/(TP + FN)$ and $accuracy = (TP + TN)/\ (TP + FP + TN + FN)$. As shown in Table 3, the floor clustering algorithm using floor similarity and floor constraints can efficiently cluster trajectories into floor-based groups. The floor-clustering accuracy achieves an average precision of 95.2 percent, recall of 88.9 percent and final accuracy of 97.1 percent.

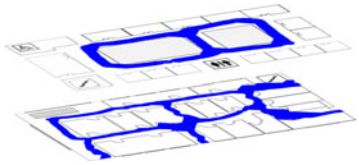


Fig. 17. Multi-floor floor plan construction.

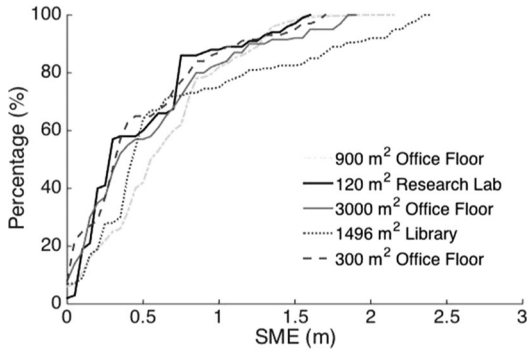


Fig. 18. CDF of SME in floor plan construction.

9.3.3 Floor Plan Construction

Since each floor cluster contains trajectories from a single floor, the floor plan construction algorithms can be applied to each individual cluster to generate a floor plan for that floor. By looking at the relative floor constraints among all floor clusters, the relationships between each pair of generated floor plans can be obtained, resulting in a multi-floor floor plan as shown in Fig. 17.

Since the input trajectories for merging contain stride length information, the constructed floor plans are in the unit of meters. We scale the constructed floor plan to the same scale of the ground truth indoor floor map and overlay them assuming one GPS location is available. The sample overlaid graphs are shown in Figs. 12 and 13. To measure SME, each step is assigned a global ID such that the final location in the constructed floor plan can be found. We mark the localizations on the real floor map for steps that have ground truth locations tapped by participating users and measure the SME by measuring the distances between the steps in the constructed floor plan and the ones on the real floor map. Fig. 18 shows the CDF of SME for all five different indoor environments. As shown in the figure, in all environments MPiLoc achieves consistent step mapping performance, with an average SME of 0.76 m for all five indoor floors. The results reflect that the constructed maps of walking paths fit the ground truth indoor floor plans well, and the floor plan construction algorithm can be applied in various indoor environments.

Fig. 19 shows the CDF of average SME of MPiLoc and Walkie-Markie in floor plan construction. Walkie-Markie relies on WiFi-Marks for floor plan construction. One key difference between WiFi-Mark and the signal correlation is that signal correlation extends the single tipping point matching in WiFi-Mark to whole spectrum matching in MPiLoc. In practice, most APs detected in the indoor space have medium RSSI values ranging from -85 dbm to -65 dbm. Within this range RSSI changes in a more random manner when users are walking indoor, and peaks are harder to be detected to form WiFi-Marks. For those with stronger RSSI values (e.g., $>$-65 dbm), WiFi-Marks can be detected but due to the fluctuation of RSSI, the locations in

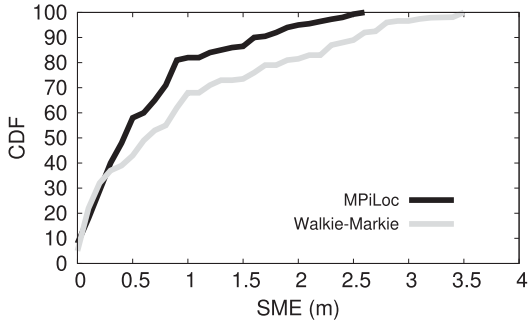


Fig. 19. Floor plan construction comparison between MPiLoc and Walkie-Markie.

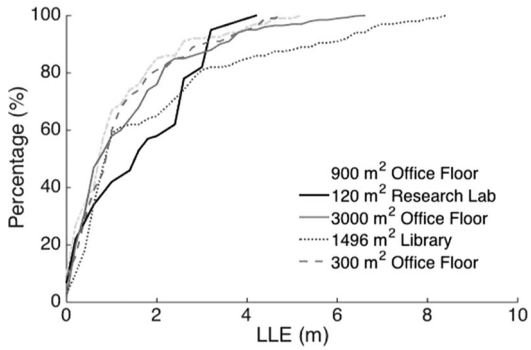


Fig. 20. CDF of LLE in final localization.

TABLE 4
Comparison of Related Localization Systems

System	Average LLE	Effort
RADAR [14]	2∼ 5 m	Site survey
Horus [41]	∼ 1 m	Site survey
Zee [4]	1∼ 3 m	Floor plan
UnLoc [3]	1∼ 2 m	Door localization
LiFS [2]	3∼ 7 m	Floor plan, time-consuming
MPiLoc	1∼ 3 m	Self-calibrating

which the peaks are detected can be significantly varied, resulting in the larger errors in the floor plan construction. As shown in Fig. 19, due to more reliable matching using signal correlation, MPiLoc achieves significant performance gain with about 32 percent less SME on average in floor plan construction.

9.3.4 Localization

Localization performance is evaluated in all five indoor environments to measure the accuracy of the constructed radio maps. As shown in Fig. 20, in all five different indoor environments, MPiLoc achieves high localization accuracy with an average LLE of 1.82 m, and 80 percent of localization errors are less than 3 meters. Table 4 provides a brief summary and qualitative comparison between MPiLoc and other localization systems. As the evaluations are performed in different settings, the localization errors listed (obtained from the respective papers) can only provide a high level guide to the relative performances of the various systems. Even though MPiLoc does not require manual calibration and landmarks, it can achieve localization accuracy that is comparable with that of the other localization schemes.

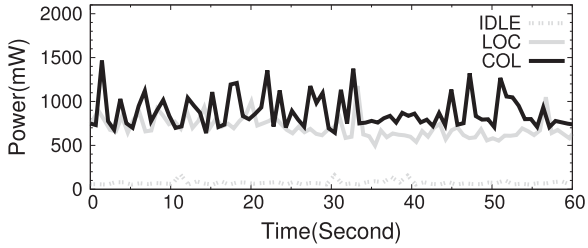


Fig. 21. Power profile of MPiLoc in different states.

TABLE 5
Power Consumption Measurement

	IDLE	LOC	COL	Tri-Scan
Power	74.8 mW	714.7 mW	852.2 mW	462 mW

9.3.5 Power Consumption

To evaluate the energy consumption, we use Monsoon Power monitor to profile the power cost of MPiLoc in three states. The one-minute snapshots for different states are shown in Fig. 21. We keep the display off for accurate measurement for all the three states. Three different phone models are used for power measurement: Google Galaxy Nexus, Samsung S3 and S4. As shown in Table 5, the average power consumptions of three WiFi scanning modes are 74.8, 714.7, 852.2 mW respectively. As shown in Fig. 21, the COL state is the most power hungry and incurs an additional 137.5 mW on top of the normal WiFi scanning used in LOC state. Running MPiLoc in both the COL state and the LOC state incurs roughly additional power consumption of 700 mW more than in the IDLE state when only IMU sensors are sampled, which indicates that we should switch to the IDLE state whenever possible. We simulate the state transitions of the sensor-trigger scanning scheme by looking at the step patterns and the heading angles in the uploaded walking trajectories and measure the final power consumption based on the percentage of time staying in different states. As shown in Table 5, the sensor-triggered scanning reduces the average power consumption to 462 mW, which corresponds to a battery lifetime of approximately 20 hours.

10 Discussions and Limitations

10.1 Error Correction of Smartphone Dead Reckoning

Despite significant amount of research efforts and progress in the literature, efficient and automatic error correction for smartphone dead reckoning remains a challenging open problem. More sophisticated error correction techniques are the key to the performance improvement of participatory sensing systems that rely on accurate inertial sensing. Future development of highly efficient error correction mechanisms for smartphone dead reckoning can be seamlessly integrated with the MPiLoc's inertial sensing module to further improve its overall performance, and we leave it as an important future work to explore.

10.2 Fingerprint Database Update

To reflect the environmental dynamics, the fingerprint database needs to be periodically updated. In MPiLoc, we

periodically update the fingerprint database by merging the newly collected walking trajectories and by running the matching and merging algorithms over them. After the merging process, the fingerprint database is updated by simply appending the new fingerprints into the database and removing those older than certain period (e.g., one month). While we find that this simple update strategy is able to keep the system working, more sophisticated updating strategies are possible (e.g., aging mechanisms) to improve the system update performance.

10.3 Complicated Floor Plans

MPiLoc focuses on commonly seen office-like environments where people work and live daily. To construct indoor floor plans, MPiLoc extracts turn segments and line segments for matching. Extending to more complicated layouts containing curve shapes and large open space requires extracting additional curve segments. Although in practice walking paths inside buildings are often separated by walls or other obstacles, in open spaces where people may not walk along distinct walkways, path correlation and signal correlation may fail to differentiate intersecting or parallel aisles that are not separated by sufficiently large distances. To extend the system capability to cover large open space, it is possible to integrate MPiLoc with fixed-starting-point dead reckoning approach such as [11] to extend the coverage of the constructed floor plans. Also, in some specially designed indoor environments with big open areas, such as shopping malls with big open space in the middle, the wireless signal exhibit more similarities in each floor, and the confusion rate might increase as well. It is possible to further improve the floor clustering performance in such environments through techniques such as sophisticated barometer calibration [27] and include absolute barometer readings in the floor clustering process. These are important open problems and we leave them as future directions to explore.

11 Conclusion

In this paper we propose and evaluate MPiLoc, an indoor localization scheme that takes user walking trajectories as input and automatically builds and updates the indoor floor plan in multi-floor indoor environment. By incorporating radio fingerprints, the indoor radio map is also automatically managed by MPiLoc. MPiLoc requires no human intervention, works in multi-floor indoor environment, and can achieve high localization accuracy with an average error of 1.82 meters. As MPiLoc only requires minimal user effort for calibration and maintenance, it has the potential for large-scale deployment in practice.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 61602319, No. 61572330, No. 61300174), Natural Science Foundation of SZU (Grant No. 2016048), and the Technology Planning Project (Grant No. 2014B010118005) from Guangdong Province, China. Jianqiang Li is the corresponding author. A preliminary version of this work appeares in ACM/IEEE IPSN 2014 [1].

References

[1] C. Luo, H. Hong, and M. C. Chan, "Piloc: A self-calibrating participatory indoor localization system," in *Proc. 13th Int. Conf. Inf. Process. Sensor Netw.*, 2014, pp. 143–153.
[2] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 269–280.
[3] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, 2012, pp. 197–210.
[4] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 293–304.
[5] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-Markie: indoor pathway mapping made easy," in *Proc. 10th USENIX Conf. Netw. Syst. Des. Implementation*, 2013, pp. 85–98.
[6] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-Navi: Self-deployable indoor navigation system," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 471–482.
[7] C. Luo, H. Hong, L. Cheng, M. C. Chan, J. Li, and Z. Ming, "Accuracy-aware wireless indoor localization: Feasibility and applications," *J. Netw. Comput. Appl.*, vol. 62, pp. 128–136, Feb. 2016.
[8] C. Luo, L. Cheng, M. C. Chan, Y. Gu, J. Li, and Z. Ming, "Pallas: Self-bootstrapping fine-grained passive indoor localization using WiFi monitors," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 466–481, Feb. 2017.
[9] N. B. Priyantha, "The cricket indoor location system," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, USA, 2005.
[10] J. Xiong and K. Jamieson, "ArrayTrack: A fine-grained indoor location system," in *Proc. 10th USENIX Symp. Netw. Sys. Des. Implementation*, 2013, pp. 71–84.
[11] M. Alzantot and M. Youssef, "CrowdInside: Automatic construction of indoor floorplans," in *Proc. 20th Int. Conf. Advances Geographic Inf. Syst.*, 2012, pp. 99–108.
[12] R. Gao, et al., "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 249–260.
[13] S. Chen, M. Li, K. Ren, X. Fu, and C. Qiao, "Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 59–71.
[14] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM*, 2000, pp. 775–784.
[15] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial internet: A survey on the enabling technologies, applications, and challenges," *IEEE Commun. Surveys Tuts.*, 2017.
[16] B. Wang, S. Zhou, W. Liu, and Y. Mo, "Indoor localization based on curve fitting and location search using received signal strength," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 572–582, Jan. 2015.
[17] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "FM-based indoor localization," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, 2012, pp. 169–182.
[18] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the Mona Lisa: Spot localization using PHY layer information," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, 2012, pp. 183–196.
[19] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: Mobile phone localization via ambience fingerprinting," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 261–272.
[20] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 173–184.
[21] B. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, vol. 7, no. 1, pp. 2480–2485.
[22] P. Mirowski, T. K. Ho, S. Yi, and M. MacDonald , "SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, 2013, pp. 1–10.
[23] Q. Chang, S. Van de Velde, W. Wang, Q. Li, H. Hou, and S. Heidi, "Wi-Fi fingerprint positioning updated by pedestrian dead reckoning for mobile phone indoor localization," in *Proc. China Satellite Navigat. Conf.*, 2015, pp. 729–739.

[24] Z. Chen, H. Zou, H. Jiang, Q. Zhu, Y. C. Soh, and L. Xie, "Fusion of WiFi, smartphone sensors and landmarks using the Kalman filter for indoor localization," *Sensors*, vol. 15, no. 1, pp. 715–732, 2015.
[25] L. Sun, Z. Zheng, T. He, and F. Li, "Multifloor Wi-Fi localization system with floor identification," *Int. J. Distrib. Sensor Netw.*, vol. 2015, 2015, Art. no. 82.
[26] V. Radu and M. K. Marina, "HiMLoc: Indoor smartphone localization via activity aware pedestrian dead reckoning with selective crowdsourced WiFi fingerprinting," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, 2013, pp. 1–10.
[27] G. Liu, et al., "Beyond horizontal location context: Measuring elevation using smartphone's barometer," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 459–468.
[28] H. Ye, T. Gu, X. Tao, and J. Lu, "B-Loc: Scalable floor localization using barometer on smartphone," in *Proc. IEEE 11th Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2014, pp. 127–135.
[29] H. Ye, T. Gu, X.-P. Tao, and J. Lv, "Infrastructure-free floor localization through crowdsourcing," *J. Comput. Sci. Technol.*, vol. 30, no. 6, pp. 1249–1273, 2015.
[30] J. Dong, Y. Xiao, Z. Ou, Y. Cui, and A. Yla-Jaaski , "Indoor tracking using crowdsourced maps," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, 2016, pp. 1–6.
[31] Android motion sensors. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion.html
[32] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2011, pp. 222–230.
[33] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proc. ACM Conf. Pervasive Ubiquitous Comput.*, 2012, pp. 421–430.
[34] K. Sankaran, M. Zhu, X. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, "Using mobile phone barometer for low-power transportation context detection," in *Proc. 12th ACM Conf. Embedded Netw. Sensor Syst.*, 2014, pp. 191–205.
[35] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proc. ACM Conf. Pervasive Ubiquitous Comput.*, 2012, pp. 481–490.
[36] Erosion and dilatation. [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html
[37] J. Niu, B. Lu, L. Cheng, Y. Gu, and L. Shu, "ZiLoc: Energy efficient WiFi fingerprint-based localization with low-power radio," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2013, pp. 4558–4563.
[38] The IW wireless package. [Online]. Available: https://wireless.wiki.kernel.org/en/users/Documentation/iw
[39] N. Roy, H. Wang, and R. Roy Choudhury , "I am a smartphone and i can tell my user's walking direction," in *Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2014, pp. 329–342.
[40] R. J. Hodrick and E. C. Prescott, "Postwar us business cycles: An empirical investigation," *J. Money Credit Banking*, vol. 29, no. 1, pp. 1–16, 1997.
[41] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. 3rd Int. Conf. Mobile Syst. Appl. Services*, 2005, pp. 205–218.

Chengwen Luo received the PhD degree from the School of Computing, National University of Singapore (NUS), Singapore. He was a post-doctoral researcher in computer science and engineering at the University of New South Wales (UNSW), Australia. He is currently an assistant professor in the College of Computer Science and Software Engineering, Shenzhen University, China. He has authored or co-authored research papers in top venues of mobile computing and wireless sensor network, such as ACM SenSys and ACM/IEEE IPSN. His research interests include mobile and pervasive computing, indoor localization, wireless sensor network, and security aspects of Internet of Things.

Hande Hong received the BS degree from the Department of Computer Science, Zhejiang University, Hangzhou, China, in 2012. He is working toward the PhD degree in the School of Computing, National University of Singapore, Singapore. His research interests include mobile computing and context-aware systems.

Mun Choon Chan received the BS degree in computer and electrical engineering from Purdue University, in 1990, and the MSEE and PhD degrees both from Columbia University, in 1993 and 1997, respectively. He is currently an associate professor in the Department of Computer Science, National University of Singapore (NUS). From 1997 to 2003, he was a member of technical staff in the Networking Research Laboratory, Bell Labs, Lucent Technologies. He joined NUS in 2004. He has served as a program committee and editorial board member for numerous international conferences and journals such as the *IEEE Transactions on Mobile Computing*. He is a member of the IEEE.

Jianqiang Li received the BS and PhD degrees from the South China University of Technology, in 2003 and 2008, respectively. He is an associate professor in the College of Computer and Software Engineering, Shenzhen University. He led a project of the National Natural Science Foundation, and a project of the Natural Science Foundation of Guangdong Province, China. His major research interests include hybrid systems, Internet of Things, and embedded systems.

Xinglin Zhang received the BE degree from the School of Software, Sun Yat-Sen University, in 2010 and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2014. He is currently with the South China University of Technology. His research interests include wireless ad-hoc/sensor networks, mobile computing, and crowdsensing. He is a member of the IEEE and the ACM.

Zhong Ming is a professor in the College of Computer and Software Engineering, Shenzhen University. He led three projects of the National Natural Science Foundation, and two projects of the Natural Science Foundation of Guangdong Province, China. His major research interests include Internet of Things and cloud computing. He is a senior member of the Chinese Computer Federation (CCF).