A Self-Evolving WiFi-based Indoor Navigation System Using Smartphones

Zhenyong Zhang, Shibo He, *Member*, *IEEE*, Yuanchao Shu, *Senior Member*, *IEEE*, Zhiguo Shi, *Senior Member*, *IEEE*

Abstract—Given a wide spectrum of demands for indoor location-based service, great research effort has been devoted to developing indoor navigation systems. Nevertheless, due to high engineering complexity and expensive infrastructure and labor cost, scalable indoor navigation is still an unsolved problem. In this paper, we present SWiN, a Self-evolving WiFi-based Indoor Navigation system. SWiN provides plug-and-play and light-weight indoor navigation in a sharing manner. To alleviate the impact of the environmental change and device diversity, SWiN extracts both the static and dynamic properties of WiFi signals including scanned AP list, variations of signal strength and AP's relative strength order. SWiN exploits the leader-follower structure, navigating following users by tracking their motion patterns to provide real-time navigation guidance. In specific, during navigation, SWiN utilizes a light-weight synchronization algorithm to synchronize multi-dimensional WiFi measurements between leader and follower traces. Furthermore, a trace updating mechanism is developed to guarantee the long-term utility of SWiN by extracting useful information in followers' traces. Consolidating these techniques, we implement SWiN on commodity smartphones, and evaluate its performance in a five-story office building and a newly opened two-story shopping mall with test areas over $8000m^2$ and $6000m^2$, respectively. Our experimental results show that 95% of the tracking offsets during navigation are less than $2m$ and $3.2m$ in these two environments.

Index Terms—Indoor navigation, self-evolving, WiFi-based, leader-follower, long-term utility, smartphone.

1 Introduction

Numerous studies have been recently devoted to realizing the mobile-device-based (MDB) indoor navigation/localization, using various technologies such as WiFi [1–3], Bluetooth [4], Sound [5, 6], Ultra-wide band [7] and etc. Conventional approaches with high-end devices, density deployed beacons and floor plans could make the accurate and real-time indoor navigation/localization a reality, however, they are not always available and affordable. For example, the mainstream navigation service providers such as Google Maps and Apple Maps currently only promote indoor positioning in limited areas such as large malls, stations and airports. To address this issue, an alternative way is to utilize the ubiquitous indoor signals such as geo-magnetic field [8, 9], light sight [10] and RF signal [11, 12], enabling indoor localization depending on a location-representative fingerprinting map with the association of a digital map. Nevertheless, this kind of map-based systems have two limitations. First, the map construction process is labour-intensive and needs professional help. Second, due to the difficulty of calibrating the map, the localization accuracy cannot be guaranteed. Thus, these indoor localization systems may not be able to satisfy users' demands.

- *Z. Zhang and S. He (corresponding author) are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou, China. E-mail: zhangzhenyong@zju.edu.cn, s18he@zju.edu.cn*
- *Y. Shu is with Microsoft Research Redmond, Washington, USA. E-mail: yuanchao.shu@microsoft.com*
- *Z. Shi is with the College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China. E-mail: shizg@zju.edu.cn*

Manuscript received January 7, 2018; revised August 19, 2018; revised April 17, 2019; accepted April 28, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61672458 and Zhejiang Natural Science Foundation under grant No. LR16F020001.

Although extensive efforts have been paid on feasible indoor navigation by reducing the map construction and calibration cost [13–16], the resulting solutions induce additional concerns for indoor navigation. For example, crowdsourcing-based systems need a sustainable incentive mechanism [17–20], and model-based systems need a precise building structure information [21]. Moreover, indoor localization system has to work with path-planning algorithms to enable navigation [22]. Therefore, an ideal indoor navigation system should avoid complicate map creation, expert support and tedious manual for users.

Recently, there are increasing studies on easy-to-deploy navigation systems, which do not depend on the pre-deployed comprehensive map or path-planning algorithms. In these literatures, the well-known leader-follower structure is commonly adopted [23–26]. Translating this structure to the navigation case, a previous visitor acts as the "leader" sharing his/her trace during an indoor trip with latecomers. Then, the latecomers use this trace as a reference and track it to the desired destination as the leader did before. Given an example, the conference organizer can construct reference traces to different meeting rooms for attendees, such that, attendees are able to find their ways to their interest rooms by following these traces. In fact, this mechanism is widely applied in many social apps such as WeChat, Skype and Facebook, where a user can share his locations to his friends. Due to the self-deployable properties, these systems provide a more promising way to meet the ever-growing demands for indoor navigation.

In this paper, we design a real-time indoor navigation system called SWiN based on the leader-follower structure, which provides plug-and-play, light-weight and user-friendly navigation without a comprehensive map and/or

localization-assisted path planing. SWiN is inspired by two characteristics of the old Chinese saying "*Many hands make light work* " : benefit and contribution. Specifically, the previous travelers (leaders) contribute his "footmarks" as a reference trace for the followers, who benefit from this provided convenience for easier way-finding. Meanwhile, the followers can also contribute to improve this service by renewing the "footmarks" for visitors after them. SWiN leverages the ubiquitous WiFi signal, which is extensively distributed with densely deployed APs in indoor environments. A leader records the sensory data and WiFi signals, using his smartphone during the trip from one position to another position. The position-specific features extracted from the WiFi signal (acting as "footmarks") are combined with the leader's motion events (e.g., steps, turns and going upstairs/downstairs) to construct the reference trace. As the following visitor begins the trace, his start point is locked on firstly. After that, SWiN navigates him to the destination in accordance with motion hints shown on the smartphone screen. Furthermore, the useful information in the followers' traces are abstracted to maintain the long-term utility of SWiN.

Based on the above paradigm, the realization of SWiN entails particular challenges. (1) SWiN is installed on the smartphone and expected to guide the follower with timely and accurate motion hints from the start point to the final destination. With interferences from multi-path effect, device-diversity and users' different walking patterns, it is possible that untimely and wrong instructions are displayed, leading to a failed guidance. Therefore, how to accurately compare and synchronize the follower's trace with the reference trace is the most important and challenging issue. (2) Incorrect and untimely instructions or intended actions (e.g., taking a deliberate turn) can deviate the follower from the correct path. Due to the limited information provided by a single reference trace in the leader-follower structure, SWiN must detect this deviation timely and navigate the follower back to the right location. (3) As some traces may be visited frequently by users, an user-friendly navigation system should remain effective for a long period of time. To avoid rebuilding the reference trace repeatedly, SWiN has to exploit the information in the followers' traces to automatically update it. Since there is no direct control of follower's behaviors, utilizing traces with low qualities may seriously affect the updating promotions.

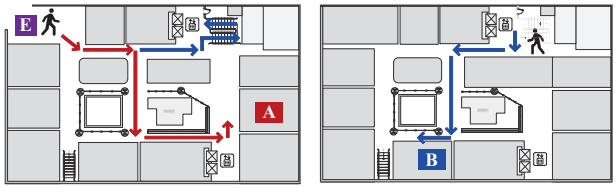
In this study, we devise efficient solutions to address these issues. First we propose a new quantification metric to measure the dissimilarity between two WiFi signal samples. To alleviate the impact of the multi-path effect and device diversity, both the static features (e.g., the sensed AP set and APs' relative strength orders) and dynamic features (e.g., the gradient information) are extracted from the WiFi signals. These features are overall considered to form a composite metric for calculating the dissimilarity at each point as the follower tracks the reference trace. We observe that the quantification result has nice properties for precise trace synchronization. During the navigation phase, we propose a self-calibration strategy to guarantee the accuracy of the synchronization result. Specifically, the local optimums of the online signal match algorithm are stochastically calibrated by a parallel global optimum algorithm. Moreover, the abnormal change of the dissimilarities along with the synchronization results is analysed for deviation detection. We use a statistical method to detect the beginning of the deviation for navigating the follower back inversely, using his own signal recordings. Furthermore, we develop a trace updating mechanism by jointly considering the merging and replacing process, making use of useful information in the followers' traces. Unlike traditional methods, it works in a bootstrapping manner. The merging process and replacing process operate alternately to avoid environmental changes and accumulated merging errors.

Synthesizing the above techniques, we implement SWiN on the Android platform. The system performances are evaluated by conducting extensive experiments in multiple real-world scenarios such as the office building and shopping mall. More than 30 volunteers are involved and 10 miles traces are collected. The whole experiments take more than 6 months. In experiments, SWiN shows promising results with accurate tracking and timely instructions (95% of offsets are within 2m in the office building and less than 10% failure rate), proper deviation warning (within 5s and less than 10% false positive rate) and long-term utility (more than a week).

The main contributions of this paper are summarized as follows:

1) Our system design is centered on WiFi signals. Both the static and dynamic properties are extracted from WiFi signals to form a composite metric, which is able to precisely quantify the dissimilarity between WiFi signal samples.
2) We devise a new step-constrained hybrid synchronization algorithm considering the real-time, accuracy and adaptivity attributes for users' walking progress estimation. An updating mechanism is designed to guarantee the long-term usability of the reference trace, enhancing the system's utility and extending the applicable cases.
3) Finally, we implement SWiN in multiple commercial smartphones and evaluate its performances in different real-world scenarios such as the office building and large shopping mall. The experimental results demonstrate that SWiN can provide delightful and satisfactory navigation service.

The rest of this paper is organized as follows. In Section 2, we give an application example and present the system architecture. The detailed design of SWiN is provided in Section 3. We present our mechanism for updating the reference trace in Section 4, and evaluation results in Section 5. Section 6 reviews related works, and Section 7 concludes the paper and presents some discussions.



(a) (b)

Fig. 1. An application example: (a) The first floor; (b) The second floor.

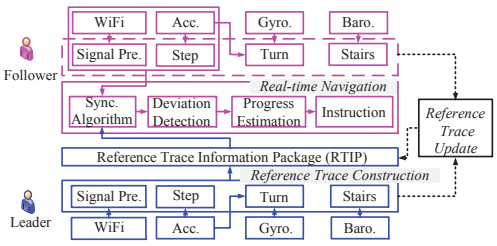


Fig. 2. System architecture of SWiN.

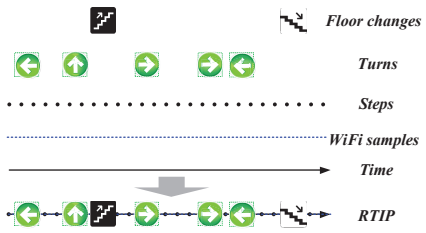


Fig. 3. The generation of RTIP.

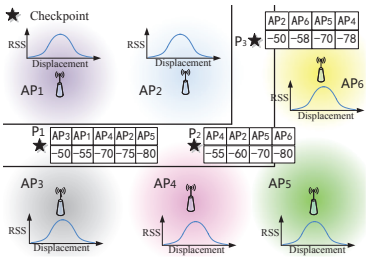


Fig. 4. An illustration of the position-specific property of WiFi signal (RSS unit: dBm).

2 SWiN OVERVIEW

2.1 An application example

Considering the scenario in which Alice and her friend Bob visit different shops in an unfamiliar shopping mall (Fig. 1). Alice's destination is shop A on the first floor, whereas Bob's is shop B on the second floor. The shopping mall is too large for new customers to find a given place easily and quickly. To help their businesses, the owners of shop A and shop B have built the reference traces from the mall's entrance E to their respective shops. They shared these traces to a trace center. Previously, Alice and Bob have installed SWiN on their smartphones. Now they can download the traces from the center, and follow the motion hints promoted by SWiN to shop A and shop B, respectively. After they have visited their shops, they can change their roles from the follower to the leader. They can share their collected traces to the trace center and contribute to update the reference traces from E to A and B. Moreover, Alice can build a new trace from shop A to shop C, which has not been built before. Then, Alice can share this trace to Bob directly for him to find her.

2.2 System architecture

SWiN mainly composes of three parts: the reference trace construction, the real-time navigation and the update of the reference trace. Fig. 2 shows the overall design structure.

Reference trace construction. The construction process of the reference trace starts right after the leader turns on SWiN. It samples gyroscope, accelerometer, and barometer readings as well as WiFi signals along the trace during the leader's trip. When the leader stops at the destination, SWiN launches a series of signal processing methods to detect the leader's motion events such as turns, steps and going upstairs/downstairs. Together with the preprocessed results of WiFi signals, a reference trace information package (RTIP) is generated. In RTIP, all motion events are indexed with the time-stamped WiFi signals, as shown in Fig. 3. The leader can share RTIP directly to the follower or store it in the trace center (e.g., a remote server).

Real-time navigation. SWiN navigates the follower to the same destination as the leader does by presenting timely motion hints, in accordance with the synchronization results indicated by RTIP. Along with the follower's tracking, WiFi signals are preprocessed and steps are measured based on the incoming WiFi and accelerometer recordings, respectively. With the previously loaded RTIP, the navigation phase runs a signal synchronization algorithm to estimate the portion the follower has walked. The deviation detector following the synchronization process is used to monitor whether or not the follower walks along the correct path. SWiN warns the user and navigates him back when it happens.

Reference trace updating. SWiN updates some traces that are frequently visited by users in the trace center to maintain their effectiveness for a relatively long time. Actually, there are two cases should be considered. If the user directly shares the trace to the latecomers, we do not need to update it because the RTIP is used only once. While if the user shares the trace to the trace center, which manages all traces and provides them to users in need, then we can update it by extracting the useful information in the followers' traces. All users can contribute to this renovation process.

3 SYSTEM DESIGN

The following paragraphs provide details about the design of SWiN. To provide satisfied indoor navigation service, SWiN estimates the follower's mirror position in accordance with the reference trace. To some extend, SWiN is designed as an up-to-date system for commercial and social scenarios, which inevitably brings in a few challenges. First, both the leader and follower only walk along the pathway once and pass every physical position in a very short time. Thus, the collected WiFi signal may not be that representative. Second, with the limited information provided by this single reference trace, the user, if he/she has deviated from the correct path, has to be navigated back. SWiN should enable this capability for this annoying case. Third, due to users' different walking speeds and the device diversity, SWiN should be designed to be adaptable for these interferences.

3.1 WiFi signal preprocessing

During the walk, WiFi antenna in the smartphone scans the surroundings at a fixed sampling rate. We record the sensed APs' information including MAC address, RSS value and timestamp. To improve the position-specific attribute of WiFi signals, we extract both the static and dynamic properties of them.

3.1.1 Static property

Even though APs are densely distributed in the indoor environments, we can not assure that every AP can be sensed by the smartphone. In fact, a smartphone can only capture a subset of visible APs at a given position. This is correlated with the AP's response rate. It has been proved that the response rate is proportional to the RSS value [42]. In SWiN, to increase the reliability of the recorded WiFi

signal, a high response rate are required by setting a RSS threshold more than -80 dBm. Unfortunately, even some APs with high RSS values may not stably exist, for example, hotspots created by users. We filter out those APs which are only sampled for a few times less than a threshold N_t, given by

$$N_t = \frac{T \times S_c}{S \times T_w}, \tag{1}$$

where T and S are respectively the time and steps taken by the user to finish a given path, T_w is the sampling time of the WiFi signals, S_c is the minimum steps that the AP should be recorded. That is, if the AP is recorded less than S_c steps, we will discard it. On the basis of steps that each AP is sampled, we only filter out low-quality APs that are recorded for a few times such as hotspots which may exist temporarily and has a relatively short broadcast range.

Particularly, we observe that the APs' strength order along a path is position-specific. Due to the propagation property of the WiFi signal, along the path, the signal sequence of a given AP shows an obvious peak (Fig. 4). Since APs are placed at various locations in the indoor space, at different positions, the strength orders of the recorded APs are different. For example, the sensed AP sets at checkpoints P_1, P_2 and P_3 have different strength orders. Therefore, for every recorded WiFi signal sample, we sort the sensed APs according to their RSS values, i.e

$$\bar{w}_i = sort(w_i), \tag{2}$$

in which, $w_i = (t_i, mac_i^o, rss_i^o)$ is the ith WiFi signal sample after filtering out some APs using equation (1). $\bar{w}_i = (t_i, mac_i, rss_i)$ is the ranking result, where t_i is the timestamp. $mac_i := \{ad_{i1}, ad_{i2}, \ldots, ad_{ip}\}$ is the MAC set, where ad_{ij} denotes the MAC address of the jth AP in the ith WiFi signal sample. $rss_i := \{r_{i1}, r_{i2}, \ldots, r_{ip}\}$ is the set of RSS values, where r_{ij} denotes the RSS value of the jth AP in the ith WiFi signal sample. p is the number of APs in the ith WiFi signal sample.

3.1.2 Dynamic property

Since the biased RSS measurements across devices along with transmission power control techniques of WiFi APs undermine the fidelity of the fingerprint-based localization/navigations systems, we leverage the more robust and stable gradient information of RSS values. Actually, this gradient information reflects the transition (dynamic) difference between WiFi signal samples.

Gradient code. Let $\bar{W} := \{\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_n\}$ be the WiFi signal sequence after being processed by the methods proposed in Section 3.1.1. Suppose $\bar{w}_i$ and $\bar{w}_k$ ($i < k \leq p$) are two WiFi signal samples. The difference of the MAC sets between these two samples is computed by

$$E = mac_i \cap mac_k, C = mac_k - I, \tag{3}$$

where E is an intersection set, C is a complementary set, which means that every element in C belongs to mac_k but dose not belong to mac_i. We denote $\mathcal{E}$ and $\mathcal{C}$ as the sets containing the indices of the MAC addresses in E and C, respectively.

Then, we compute the RSS difference by

$$dist_{kj} = \begin{cases} r_{kj} - r_{ij}, j \in \mathcal{E} \\ 100, \qquad j \in \mathcal{C}\ , \end{cases} \tag{4}$$

where r_{kj} and r_{ij} are respectively the RSS values of the jth AP in the kth WiFi signal sample and the ith WiFi signal sample. In order to alleviate the impact of the environmental changes and device diversity, we map $dist_{kj}$ to four values, given by

$$c_{kj} = \begin{cases} 0, & abs(dist_{kj}) \leq \theta, \\ 1, & dist_{kj} > \theta, \\ -1, & dist_{kj} < -\theta, \\ 2, & dist_{kj} = 100. \end{cases} \tag{5}$$

where θ is 6 dBm, which is determined according to our experimental results. Inspired by the concept of "code alphabet", we term $\{-1, 0, 1, 2\}$ as the gradient codes. Finally, we transfer the original WiFi signal sequence to a coded WiFi signal sequence (CWSS): $\tilde{W} := \{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_n\}$, in which, $\tilde{w}_i = (t_i, mac_i, cod_i)$. $\tilde{W}$ is formed by letting $k - i = 1$, thus, $cod_k = \{c_{k1}, c_{k2}, \ldots\}$.

3.2 Quantification of the dissimilarity

Two recent leader-follower navigation systems [23] and [25] also adopted the WiFi signal. Zheng et al. [23] compared the absolute RSS values between two WiFi signal samples to update the weights of particles for the signal synchronization algorithm. Yin et al. [25] extracted the radio and visual features of the sequential WiFi fingerprints. Even though the later made use of image-related features of WiFi fingerprints for signal match, it still depends on the absolute RSS values of APs. While it has been proved that the absolute RSS values of APs are not stable indoors [27]. In SWiN, we do not use the absolute RSS values. We compare the dissimilarity between two WiFi signal samples by jointly considering three metrics, the gradient code distance, MAC set distance and strength order distance, to form a composite metric.

Essentially, only utilizing the difference of the gradient codes cannot accurately quantify the dissimilarity between two WiFi signal samples, as WiFi signal samples recorded at different positions may have similar gradient code sets. Using this single metric may lower the spatial discrimination of WiFi signals to a certain extend. Therefore, we consider two other features of WiFi signals. One is the common APs shared by two WiFi signal samples. Intuitively, if two WiFi signal samples share more common APs, the corresponding physical positions are closer. Nevertheless, the number of common APs cannot exactly represent the difference of two positions. For example, in Fig. 4, P_2 and P_3 have the same AP set $\{AP_2, AP_4, AP_5, AP_6\}$. But, on the other hand, we observe that the APs' strength orders are quiet different. Therefore, we also consider the difference of the APs' strength orders between two WiFi signal samples.

Gradient code distance. The gradient code distance between two WiFi signal samples is used to measure the transition difference between the corresponding physical positions, given by

$$\begin{aligned} d_1 &= \sum_{j \in \mathcal{E}_i} |cod_{ij} - cod_{i'j}| + \sum_{k \in \mathcal{C}_i} cod_{ik}, \\ E_i &= mac_i \cap mac_{i'}, C_i = mac_i - mac_i \cap mac_{i'} \end{aligned} \tag{6}$$

where cod_i and $cod_{i'}$ are respective gradient code sets in the ith WiFi signal sample and the i'th WiFi signal sample, mac_i and $mac_{i'}$ are the MAC sets, j in the index of an AP in $\mathcal{E}_i$, k in the index of an AP in $\mathcal{C}_i$.

MAC set distance. The MAC set distance denotes the difference of the MAC sets between two different WiFi signal samples, which is calculated using the Jaccard similarity, given by

$$d_2 = \frac{mac_i \bigcap mac_{i'}}{mac_i \bigcup mac_{i'}}. \tag{7}$$

Strength order distance. Even though we can not always obtain the same strength order of APs at a fixed point due to device diversity and/or multi-path effect, the relative orders of them are stable. In SWiN, we calculate the strength order distance between two WiFi signal samples by using the longest common sequence (LCS) solver [28], given by

$$d_3 = Lcs(mac_i, mac_{i'}), \tag{8}$$

where Lcs is the algorithm used to solve the longest common sequence problem.

We calculate the dissimilarity value between two WiFi signal samples using the weighted sum of the above distances. They are normalized before being added together. Since d_1 is used to measure the transition (dynamic) difference between two WiFi signal samples, if d_1 is larger, these two WiFi signal samples are more dissimilar. Therefore, we normalize d_1 using the function $g(d_1) = \frac{e^{-\frac{1}{d_1}}-1}{e-1}$. We can see that the value of $g(d_1)$ is proportional to d_1 and $0 < g(d_1) < 1$. As for d_2, it is used to measure the similarity of the MAC sets between two WiFi signal samples. If d_2 is larger, these two samples are more similar. Therefore, we normalize d_2 using the function $h(d_2) = \frac{e^{\frac{d_2}{\lambda_1}}-e^{\frac{1}{\lambda_1}}}{1-e^{\frac{1}{\lambda_1}}} (-1 \leq \lambda_1 \leq 1)$. If d_2 is smaller, $h(d_2)$ is larger, which indicates that two WiFi signal samples are more dissimilar. d_3 is used to measure the similarity of the APs' strength order between two WiFi signal samples. If d_3 is larger, it also indicates that these two samples are more similar. Therefore, we can use the same method as $h(d_2)$ to normalize d_3. Since $0 \leq d_2, d_3 \leq 1$, we have $0 \leq h(d_2), h(d_3) \leq 1$. Above all, the dissimilarity between two WiFi signal samples is given by

$$d = \alpha g(d_1) + \beta h(d_2) + \gamma h(d_3)$$

$$g(d_1) = \frac{e^{-\frac{1}{d_1}}-1}{e-1}, h(d_2) = \frac{e^{\frac{d_2}{\lambda_1}}-e^{\frac{1}{\lambda_1}}}{1-e^{\frac{1}{\lambda_1}}}, h(d_3) = \frac{e^{\frac{d_3}{\lambda_2}}-e^{\frac{1}{\lambda_2}}}{1-e^{\frac{1}{\lambda_2}}},$$

where $\alpha, \beta, \gamma > 0$, are the corresponding weights, which should satisfy $\alpha+\beta+\gamma = 1$, λ_1 ($-1 \leq \lambda_1 \leq 1$) and λ_2 ($-1 \leq \lambda_2 \leq 1$) are two tunable parameters, d is the dissimilarity value. We give some intuitive analysis for the determination of the tunable parameters λ_1 and λ_2. As for the function $h(x) = \frac{e^{\frac{x}{\lambda}}-e^{\frac{1}{\lambda}}}{1-e^{\frac{1}{\lambda}}}$, if $-1 \leq \lambda \leq 0$ and is smaller, the function $h(x)$ is more concex; if $0 \leq \lambda \leq 1$ and is smaller, the function $h(x)$ is more concave. λ can be determined according to the density of APs deployed indoors. For example, if the deployed APs are denser, it is easier to record the same APs at two positions, although they are relatively far from each other. Thus, if the deployed APs are denser, under the same x, $h(x)$ should be larger. As for the weights α, β and γ, we determine them through extensive experiments.

3.3 Online Navigation

The rationale of the navigation phase is to estimate the user's walking progress in accordance with the reference trace. Typically, it must have the following attributes: accurate, real-time and adaptable. First, accuracy is the basic requirement for a navigation system. Second, we cannot assume that the follower walks slowly enough to interact with the system or wait for the system's response. The forward instructions should be shown to the user at right locations timely. Third, SWiN must handle some uncertainties such as device diversity, users' walking patterns and environmental changes.

To achieve these attributes, we devise a step-constrained hybrid synchronization (ScHS) algorithm based on the Dynamic Time Warping algorithm (DTW) [29]. The main idea is to use a step-constrained online DTW (ODTW) algorithm to provide real-time signal synchronization results, meanwhile, a modified standard DTW (MDTW) algorithm executes occasionally to calibrate the alignment drifts for the synchronization results. The ODTW algorithm matches the follower's current signal recordings with RTIP forward. While the MDTW algorithm is executed backward to match the follower's current signal sequence with RTIP.

The basic ODTW algorithm aligns the follower's incremental CWSS $U := \{\tilde{w}_1^u, \tilde{w}_2^u, \cdots, \tilde{w}_m^u\}$ with a reference CWSS $V := \{\tilde{w}_1^v, \tilde{w}_2^v, \cdots, \tilde{w}_n^v\}$, where only the first m samples of U are known at a certain point. The goal is, for each $a = 1, 2, \cdots, m$, to find the corresponding index I_a^v in V so that the subsequence $\{\tilde{w}_1^u, \tilde{w}_2^u, \cdots, \tilde{w}_a^u\}$ is aligned to $\{\tilde{w}_1^v, \tilde{w}_2^v, \cdots, \tilde{w}_{I_a^v}^v\}$. The alignment path $Path$ is a sequence of tuple (a, I_a) obtained according to the cost matrix, whose column and row denote the indices of the WiFi signal samples in the signal sequence U and V, respectively. We use a step-constraint condition to prevent the match result of the ODTW algorithm from steep jumps or getting stuck in a local minimum. This can efficiently handle the users' walking speeds during navigation. Specifically, for a time duration $\tau = t_a^u - t_{a'}^u$ and a threshold S_τ (in SWiN, we set $\tau = 1s$ and $S_\tau = 3$ since we observe that the pace frequency of human ranges 1Hz-3Hz), the steps taken by the leader and follower are $S_{aa'}^u$ and $S_{I_{aa'}}^v$, respectively. Then, if $S_{aa'}^u - S_{I_{aa'}}^v \geq S_\tau$ (e.g., the follower speeds up), we increase the row of the cost matrix to find an optimal match in the reference trace. That is, we catch up with the follower's walking speed for signal synchronization. If $S_{I_{aa'}}^v - S_{aa'}^u \geq S_\tau$ (e.g., the follower speeds down or stop), we increase the column of the cost matrix to find an optimal match in the reference trace. That is, we slow down the forward speed of the match point in the reference trace for signal synchronization. Otherwise, for each incoming WiFi signal sample $\tilde{w}_a^u$, the current $n_t \times m_t$ cost matrix D_t is computed based on the past $n_{t-1} \times m_{t-1}$ cost matrix D_{t-1}, where $n_t, n_{t-1} \leq n$ and $m_t, m_{t-1} \leq m$. $Path$ forwards by computing a new raw, a new column, or both, relying on in which the current minimum match cost is found on D_{t-1}'s frontier: a column, a raw, or the diagonal corner, respectively.

With the ODTW algorithm, we can realize real-time navigation and make the system adapt to users' different walking speeds. Nevertheless, we have two concerns about

the alignment results. The first concern is the suboptimal match result of the ODTW algorithm. Since only the local information is used for calculating the cost matrix, the alignment result output from the ODTW algorithm is known to have alignment drifts [30]. Thus, it needs a comprehensive study where the drifts occur and how to alleviate them. The second concern is where exactly the follower starts. Generally, it seems that the start point of the follower's sequence U is the same as the reference sequence V. But we can not assure that, since the follower might not start absolutely at the same physical point as the leader does. This mismatch from the beginning may affect later alignments.

To overcome these concerns, we propose several countermeasures in SWiN. First, we use CWSSs as inputs to the synchronization algorithm and calculate the cost matrix using the dissimilarity quantification method proposed in Section 3.2. Second, we specify a local weighted recursion cell in OTDW and adaptively change the weights to calibrate the alignment drifts using MDTW. Third, we adopt an average KNN method to lock on the follower's start point.

Update the weights of the cost cell with MDTW. The main part of ODTW algorithm is the cost cell calculated for each alignment. It can be formulated as:

$$D(i,j) = min \begin{cases} \delta_1 * d(i,j) + D(i-1,j) \\ \delta_2 * d(i,j) + D(i-1,j-1) \\ \delta_3 * d(i,j) + D(i,j-1) \end{cases} \quad (9)$$

where $D(i,j)$ is the value in position (i,j) of the cost matrix D, $d(i,j)$ is the current dissimilarity value, δ_1, δ_2 and δ_3 are three local weights. We develop the MDTW algorithm to detect alignment drifts and adaptively change the local weights δ_1, δ_2 and δ_3 to calibrate them. It needs to note that MDTW and ODTW run in two parallel threads, and they can exchange information.

The MDTW thread contains two steps. First, after a match result (a, I_a^v) is obtained in the ODTW thread, we run a standard DTW algorithm in parallel using the current sequence $U = \{\tilde{w}_1^u, \tilde{w}_2^u, \cdots, \tilde{w}_a^u\}$ and the reference sequence $V_s = \{\tilde{w}_1^v, \tilde{w}_2^v, \cdots, \tilde{w}_{I_a}^v\}$. This process outputs an alignment path $Path_m = \{(1, I_1^{mv}), (2, I_2^{mv}), \cdots, (a, I_a^{mv})\}$. Since the execution speed of this thread is much slower than the ODTW thread, a new online match result has been generated when we get $Path_m$. To catch up with the follower's current signal readings, we run an ODTW algorithm serially in the MDTW thread starting from the point $(t, I_t^{mv})(t = a - 10)$ in $Path_m$ to compute a new alignment path $Path_{me}$. Because all the follower's past signals are used, the alignment path $Path_{me}$ is more accurate than the online alignment result. Suppose the current length of the follower's trace is l. The match result of the follower's lth WiFi signal sample obtained from the MDTW thread is (l, I_l^{ev}). And the match result obtained from the ODTW thread is (l, I_l^v). Then, if $I_l^{ev} \neq I_l^v$, there are match drifts in the ODTW thread. Initially, we set the local weights as $\delta_1 = \delta_3 < \delta_2$ for avoiding vertical or horizontal segments in the warping path (i.e., the alignment path). We calibrate the alignment drifts by changing the local weights according to: 1) if $I_l^{ev} > I_l^v$, we set $\delta_1 > \delta_2 > \delta_3$, that is, we speed up the forward speed of the match point in the reference trace for signal synchronization; 2) if $I_l^{ev} < I_l^v$, we set $\delta_1 < \delta_2 < \delta_3$, that is, we slow down the forward speed of the match point in the reference trace for signal synchronization.

The follower's start point detection. Identifying the follower's start point is critical for the synchronization process. In SWiN, we adopt an average KNN method to lock on the follower's start point. Specifically, we use the WiFi signals collected by the follower during the first $3s$. The top K nearest points are selected as candidates according to dissimilarity values, which are calculated with the MAC set distance and strength order distance (i.e., $\alpha = 0$, see Section 3.2). Finally, the follower's start point can be determined by $Ini = Rnd(\frac{1}{K}\sum_{i=1}^{K} P_i)$, where Ini is the detected start point, P_i denotes the candidate point, $Rnd(\cdot)$ is the rounding operation of a number.

3.4 Turn detection

Due to the ferromagnetic interference, we cannot use the compass to detect turns in indoor environments. In SWiN, turns are detected by fusing measurements both from the accelerometer and gyroscope. Since the user may place his/her smartphones arbitrarily on body, we need to determine the smartphone's attitude first. To avoid the Gimbal Lock problem [31] that occurs when rotating the three-dimensional smartphone, we adopt the quaternion method [32] to estimate the smartphone's attitude. After that, we integrate the gyroscope's z-axis readings to compute the turn angle. Besides, we adopt the complementary filter [33] to guarantee the detection accuracy. We have conducted experiments to show the effectiveness of our method. We placed the smartphones (Samsung Galaxy S5) at different positions on the volunteer's body (Fig. 5) and collected the readings of gyroscope and accelerometer when the volunteer turned 90^o. We compared the turn angles calculated using our method and directly integrating the z-axis readings of gyroscope without any preprocessing. The results were plotted in Fig. 5. We can see that the turn angles calculated using our method are around 90^o when the phone is placed horizontally, vertically and in the hand. Even though it is 74^o when the phone is placed in the pocket, it is enough for us to detect a turn. While if we directly integrating the z-axis, the results are satisfied when the smartphone is placed horizontally and vertically, but the method fails to detect the turn when the phone is placed in the hand and the pocket.

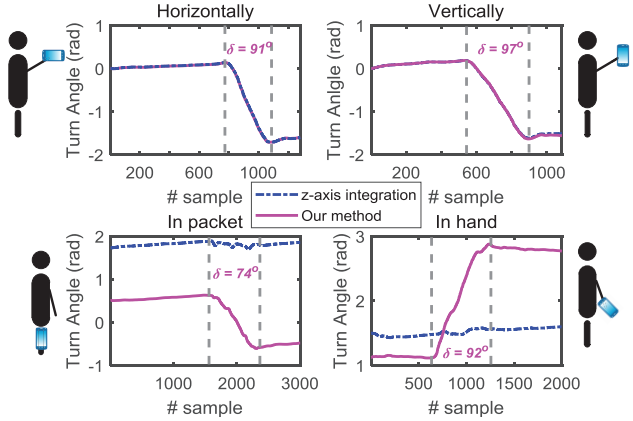


Fig. 5. The detection of turn angles with our method and integrating of gyroscope's z-axis readings without any preprocessing.

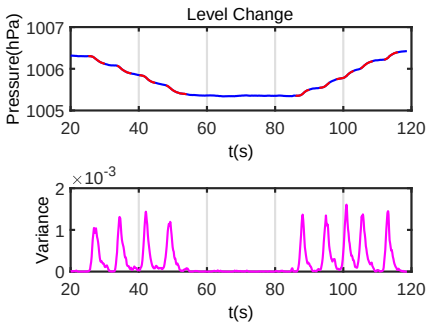


Fig. 6. Barometer readings when going upstairs/downstairs (above) and the corresponding variance (below).

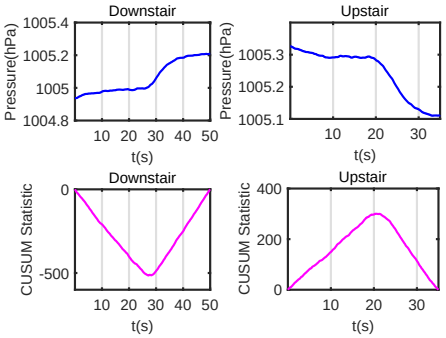


Fig. 7. Level-change detection using the CUSUM method.

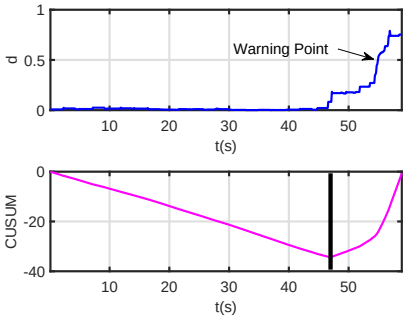


Fig. 8. Deviation detection when the user deviates the reference trace.

3.5 Level-change detection

Newly released smartphones are commonly equipped with barometer, which is a sensor can be used to detect level changes. But the noisy measurement and the absence of ground truth limit its application to determine the exact level in indoors. Nevertheless, we observe that the pressure value (i.e., the barometer's reading) drops down/rises up quickly when we go upstairs/downstairs. Therefore, we exploit the variation trend of the barometer readings to detect level changes. References [24] and [25] also adopted the barometer for detecting levels, but their detection methods depend on a heuristic threshold, which is not adapt to buildings with different floor heights. By using a single sensor such as accelerometer [34] or barometer [35], the authors realized a feasible, scalable and high-accuracy floor localization system when the floor height is unknown. But these map-based methods cannot be directly used in our paper since SWiN does not depend on map and localization.

In SWiN, we introduce the CUSUM method to detect level changes. CUSUM is a statistical analysis method used for monitoring abnormal changes when the data sequence is monotonic increasing or decreasing [36], given by

$$X_i = X_{i-1} + |b_i - \bar{b}|, X_0 = 0, \tag{10}$$

in which b_i is the pressure value, $\bar{b}$ is the average of all pressure values in the signal sequence, X_i is the current sum. The abnormal change point of the original sequence can be determined according to the extreme point of X. To detect the level change, first of all, we use a sliding window to calculate the variance of the barometer readings. As shown in Fig. 6, we can see that its value reaches a peak during the level-change period. According to the peak range of the variance, we segment the pressure sequence into multiple subsequences. For each subsequence, we use CUSUM to detect where the stairs begin. As shown in Fig. 7, the extreme points of the CUSUM statistic correspond to the start points of the level changes. We observe that the above method can also be used when the user takes escalators. When taking elevators, the user must determine which floor he/she needs to go first. To solve this issue, we ask the leader to provide the floor number of the destination.

3.6 Deviation detection

Apart from detecting turns and level changes, we also need to detect the followers' deviations from the correct path for some incidents such as being attracted by some interesting things. To handle this problem, we introduce a deviation detector following the synchronization process. Specifically, we first use a threshold-based method to automatically detect and warn the follower's deviation. Second, SWiN finds out the start point in the timestamp where the follower goes off the correct path. Based on this result, third, we separate out the subsequence of the follower's trace from the deviation point to the warning point and reverse it. Since the reverse navigation works the same with the forward process, this subsequence can be used as a new reference trace for navigating the follower back.

In SWiN, the follower's deviation is detected by tracking the dissimilarity value calculated with the ScHS algorithm. The insight is simple: the farther the offset from the correct path, the recorded WiFi signals are more different. To this end, SWiN keeps monitoring the dissimilarity value computed in the ScHS algorithm during the follower's trip. When the dissimilarity value is larger than a threshold (i.e., 0.5), the follower is warned about the deviation. Further, we detect the deviation point using the CUSUM method. Taking Fig. 8 as an example, in our experiment, the follower deviated the target trace at 48s during the trip. We can see that the dissimilarity value increases significantly after that. The follower was altered at the warning point (where the arrow points to in the figure). In the figure below, the CUSUM statistic is calculated using the sequence of dissimilarity values before this warning point. According to the dark vertical line, the deviation point was successfully detected at around 48s. With this result, we separated the follower's signal sequence upon this point and generated a reference trace by reversing the subsequence. Then, the follower tracked this new reference trace back to the correct path.

4 Reference Trace Update

For some "hot" traces which are frequently visited by users, SWiN should maintain their usability even when the indoor environments have been changed. Previously, we assume that the leader constructs the reference trace just once, which is acceptable when the reference trace is temporarily used. But it cannot guarantee the long-term usability of some frequently visited traces. In SWiN, we handle this issue by using an updating mechanism for refreshing the reference trace timely.

First of all, in order to quantify the environmental changes in a given trace, we introduce a concept named "trace diversity". It is actually used to measure the diversity of APs deployed around a given trace, given by

$$Diver = e^{-\sum_{i=1}^{n} p_i ln(p_i)} \tag{11}$$

where n is the total number of the sensed APs along a given path, p_i is the proportion of the ith AP among all APs, $Diver$ denotes the trace diversity. We did experiments to show that we can use "$Diver$" to quantify the environmental changes. Fig. 9 shows the variation of $Diver$ in a pathway for more than 4 months. We calculated $Diver$ at different times in a day and averaged their results. For the first 40 days, we observed the values of $Diver$ varied due to the interferences from weathers and human activities. At the 40th day, we deliberately switched off some APs along the trace. We can see the values of $Diver$ decreased dramatically. This indicates that the sensed WiFi signals can be changed a lot due to environmental changes.

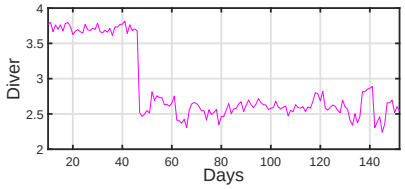


Fig. 9. The variation of the trace diversity for nearly 4 months.

Based on the above observations, we find that we must update the reference trace since SWiN may fail to work at some time due to environmental changes. We take advantage of the useful information contained in the followers' traces. In fact, the new information in the followers' traces can reflect the environmental changes to a certain extent, which can refresh the original reference trace. Technically, we update the reference trace by jointly considering the merging and replacing process. These two processes complement to each other. That is, the merging process fuses the past traces with the new trace, while the replacing process retrieves the whole reference trace due to the environmental changes and accumulated merging errors.

Note that updating the reference trace depends on the followers' inconsistent-quality traces, we encounter a set of challenges when considering extracting useful information from them. (1) Devise diversity: different WiFi chipsets are sensitive to different WiFi APs and channels. Thus, even at the same location, the recorded WiFi signals can be different across heterogeneous devices. (2) Users' casualty behaviors: as most followers are unfamiliar with the indoor environments, they may not walk as steadily as the leader does. Thus, we cannot guarantee the reliability of the provided traces.

4.1 The merging process

We merge the reference trace built by the leader with the followers' traces off-the-shelf in the trace center. As the leader and follower might carry different types of smartphones, we cannot directly average their absolute RSS values. In SWiN, we use the following normalization method to alleviate the impact of the inconsistent WiFi recordings. That is,

$$\widetilde{rss}_{ij} = \frac{rss_{ij} - rss_i^{min}}{rss_i^{max} - rss_i^{min}}, \tag{12}$$

where rss_{ij} is the RSS value of the ith AP in the jth WiFi signal sample, rss_i^{min} and rss_i^{max} are respective the minimum and maximum values of this AP along a given trace, $\widetilde{rss}_{ij}$ is the normalized result. Before normalization, we adopt an offline standard DTW to match the leader's trace with the follower's trace. According to the alignment path, for each matched pair, we average the normalized results under the constraint that the dissimilarity value between these two WiFi signal samples is less than a certain threshold. Additional APs in the follower's WiFi signal sample are added to the matched sample in the reference trace.

We conducted experiments to illustrate the necessity for normalization. As shown in Fig. 10, the RSS values of an AP along a trace were recorded by four different smartphones (vivo-X6D, Xiaomi HM 1SW, HUAWEI VNS-AL00 and Samsung Galaxy S5). We can see that the absolute RSS values have the same trend but with different magnitudes, while the normalized results track the trend and almost have the same magnitude. This indicates that the normalization can eliminate the impact of device diversity. Since the normalization process does not affect the variation trend of RSS values, the synchronization algorithm ScHS used in Section 3.3 works normally without any modification.

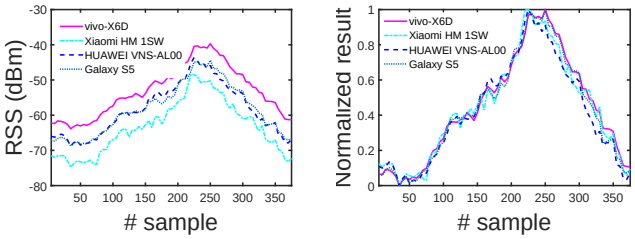


Fig. 10. An illustrative example for the normalization method.

4.2 The replacing process

Even though the merging process can alleviate the impact of the environmental changes, the merging errors will accumulate with time due to mismatches. This makes it a necessity to replace the reference trace when it becomes unsatisfied for navigation. Heuristically, we propose an event-triggered method to decide whether the reference trace should be replaced or not.

There are two events that can activate the replacing process: after a time period T_u and regional environmental changes. There is no priority between these two events. T_u is determined based on the observation of the navigation failure rate $FR = \frac{\#\ failures}{\#\ experiments}$, where $\#failures$ and $\#experiments$ respectively denote the total number of navigation failures and experiments. Navigation failure means SWiN fails to guide the follower to the correct destination. The explicit value of T_u will be evaluated in Section 5. As for the environmental change, the worst case is that the change is suddenly and in a large-scale, thus, nobody can be successfully navigated to the destination. In other words, the reference trace cannot be used anymore. In that case,

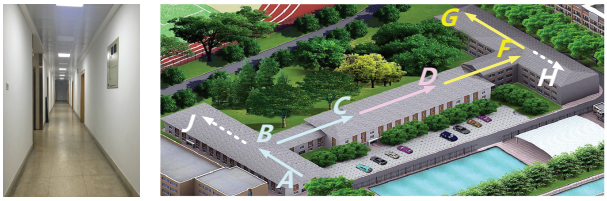


Fig. 11. The office building.

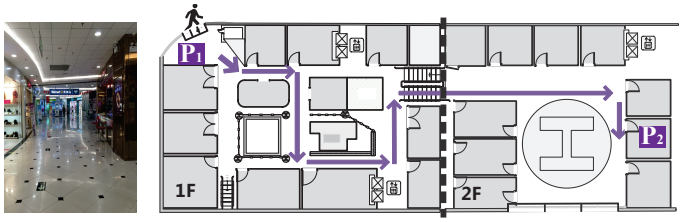


Fig. 12. The shopping mall.

the only way to retrieve the reference trace is to rebuild it again. Here we consider a weaker case that the value of trace diversity is beyond a threshold compared with that of the original reference trace. When it happens, we replace the reference trace with a new one selected in the latest followers' traces.

Note that even though the replacing process is activated, it does not mean that we will replace the reference trace with an arbitrary trace selected from the followers' traces. We determine whether the trace is qualified or not by using an verification method based on the stability of the users' steps. Specifically, we calculate the variance of the time taken by the follower's each step in a sliding time window first. We denote it as V_1, which is a sequence. Then, we compute V_1's variance v_2. Actually, v_2 can indicate the stability of the follower's walk along the trace. We observe that when the follower walks steady, v_2 is relatively small. If we assume that the leader walks steady. Then, if the value of v_2 is lower than a certain threshold comparing to that of the leader, we choose it as a new reference trace.

5 EVALUATION

In this section, we test SWiN in real-world scenarios to help better understanding of its effectiveness and limitations.

5.1 Implementation

We build a prototype of SWiN on the Android platform (version 4.4.2) and use the Lenovo T440p as a remote server (trace center). Two operational activities run separately as leader activity and follower activity in the smartphone. Users can play different roles by switching between these two activities. Navigation instructions are presented in the middle of the smartphone screen. Instructions are updated when the follower moves forward and encounters some specific places. In our experiments, all leaders collected the reference traces by holding the smartphone stably and walking steadily. It is reasonable since the leader is usually the person who tries to provide navigation service for visitors. Actually, our system is robust with respect to the leaders' walking habits and device placements. As for followers, we asked them to track the traces with their normal walking patterns. But we did not have any requirement on their behaviours such as where they should put their smartphones and which side of the passage they must follow. The recordings of IMU sensors and barometer during the trip were used to detect motion events such as steps, turns and level changes when the leader activity was activated. We only used the accelerometer readings and WiFi signal recordings when the follower activity was activated, since we only needed them to execute the signal synchronization algorithm. We collected the sensory data from the gyroscope and barometer from the followers' smartphones for the test of the reference trace update function.

5.2 Evaluation

We conducted experiments in both an $8000m^2$ five-story office building and a $6000m^2$ two-story shopping mall (Fig. 11 and Fig. 12), which were abundant with WiFi APs. More than 30 volunteers were involved in the experiments that lasted for more than 6 months. Over 15 different traces were built with a total length more than 10 miles. For each trace, it was a trip taking at least 2 minutes containing turns and stairs, which was complicated enough on which intensive efforts should be paid to find the destination without the help of a navigation system. Followers tracked the reference trace according to the navigation instructions promoted by SWiN. To remind the follower of the next motion behaviour, the instructions were displayed on the smartphone screen in advance.

5.2.1 *Determination of the weights* (α, β, γ)

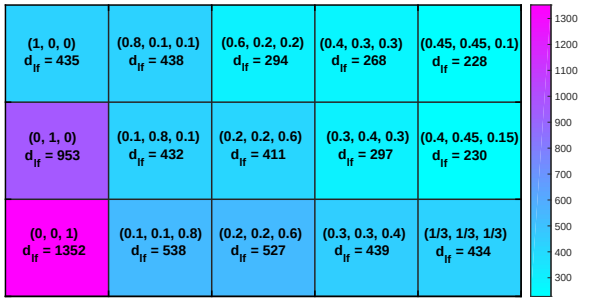


Fig. 13. The alignment results of ScHC algorithm with α, β, γ as the maximum weight, respectively.

First of all, we need to determine the values of (α, β, γ) for calculating the dissimilarity between two WiFi signal samples. Clearly, the values of λ_1, λ_2 and the values of α, β, γ are interdependent. We fixed $\lambda_1 = \lambda_2 = 0.2$ and strive to determine α, β, γ in SWiN. In the experiment, we constructed a reference trace in the office building with the volunteer walked steadily and held the smartphone (Samsung Galaxy S5) stably. After a while, another volunteer tracked this trace with a steady pace using the same smartphone. We collected these two WiFi signal sequences and preprocessed them offline. Then, we ran the proposed ScHS algorithm in the simulator implemented in PC (Lenovo T440p), which output an alignment result of these two traces. We updated the values of α, β and γ based on the alignment results. We used the L_1 norm between the index vector $I^u = \{1, 2, 3, \cdots\}$ of the followers sequence and

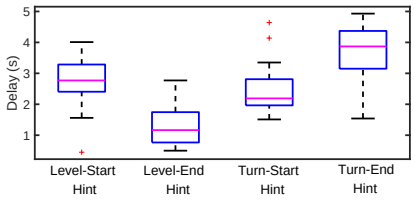


Fig. 14. Warning delays of the level-change and turning events.

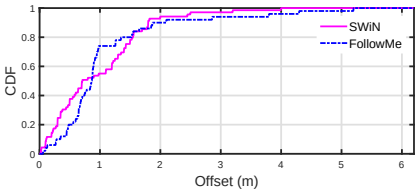


Fig. 15. Tracking offsets in the office building.

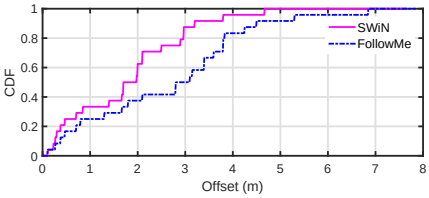


Fig. 16. Tracking offsets in the shopping mall.

the alignment result $Path = \{I_1^v, I_2^v, \cdots\}$ as the metric (u and v denote these two signal sequences, respectively). It is defined as the alignment error $d_{lf} = |Path - I^u|_1$, where $|\cdot|_1$ is the L_1 norm. Intuitively, the value of L_1 norm is relatively small if the weights α, β and γ are properly chosen. As shown in Fig. 13, we respectively set α, β and γ as the maximum weight. We can see that the values of d_{lf} are all larger than 400 when γ is set as the maximum weight, which indicates that γ should be smaller than α and β. In the other two cases when α and β are the maximum weights, we can see the alignment results are better if the difference between these two variables are smaller. Moreover, we observe that the alignment result is the best when we set $(\alpha, \beta, \gamma) = (0.45, 0.45, 0.1)$. Through more extensive experiments, we find that even though we determine the weights based on some specific scenarios, they work efficiently in other indoor environments.

5.2.2 Motion hints

We tested the delay of the level-change and turning tips by measuring the difference between the warned time and the occurred time of the true event. Level changes and turns were detected and indexed with the leader's WiFi signal sequence from A to G in the office building (Fig. 11). The reference trace included 6 specific points. A, B and C were on the first floor. A was the start point. B was a right turn corner. From C to D was a section of stairs. D, F and G were on the third floor. F was a left turn corner. G was the destination.

A shadow person (observer) walking with the follower used a stopwatch to record the warning time of the turns and stairs, the disappearance time of the warning instructions and the follower's arrival time at corners and stairs. We calculated the following delays: 1) from the warning time of stairs to the arrival time at the first stair; 2) from the arrival time of the ending stair to the disappearance time of the stair warning; 3) from the warning time of the turn to the arrival time at the corner; 4) from the complete time of the turn to the disappearance time of the turn warning. From Fig. 14, we can see that SWiN warned the user about stairs 2.75s in average before the true beginning of stairs and canceled this warning 1.2s in average after the ending of stairs. As for turns, SWiN warned the user 2.12s in average before the corner and canceled this warning 3.83s in average after the corner. We observe that the turn duration is relatively longer. This is because the turn event does not happen exactly at the corner, but before and after it to form a palpable turn angle. We also launched a survey to investigate the users' experiences, 85% of users thought that SWiN provided timely navigation instructions. Besides, they suggested that it would be better if more details were informed during navigation, for example, the number of steps needed from the current position to the next stairs or corners.

5.2.3 User navigation

Follower tracking. First of all, the start point of the follower should be locked on for the accurate navigation service. To test this functional module of SWiN, we marked several checkpoints in the reference trace from A to G. As the leader met these checkpoints, he pressed the tick button to record these events. We let 20 volunteers start at these check points. If the volunteer was locked on these checkpoints correctly, SWiN would show a right check icon on the screen. Results shown that **80%** of the followers' start points are correctly identified. With this detection result, next, SWiN estimated the follower's walking progress and promoted timely navigation instructions. We evaluated the tracking performance of SWiN by comparing it with FollowMe [24]. We conducted experiments in two different scenarios. In the office building, we set A to G as the reference trace (Fig. 11). While in the shopping mall, we set P_1 to P_2 as the reference trace (Fig. 12). Along the target trace, we marked multiple checkpoints in advance. When the leaders passed by these checkpoints, they ticked these encounters in the timestamps. These events would be shown later along with the navigation process if the follower also passed by those checkpoints. As the follower moved forward, a shadow person (observer) would measure the offsets from the marked checkpoints to the poistions where the follower was warned.

We used the latest leader-follower structure navigation system FollowMe (magnetic field-based) for comparison. The tracking offsets are shown in Fig. 15 and Fig. 16. We can see that 95% of the offsets for SWiN and 90% of the offsets for FollowMe are less than 2m in the office building. Relatively, SWiN shows a better performance. Especially in the shopping mall, SWiN has 95% of the offsets less than 3.2m, while it is less than 4.48m for FollowMe with the same percentage. We also observe that the navigation performance in the office building outperforms that in the shopping mall using both SWiN and FollowMe. We think this is because the reference trace in the office building was narrower, and there were customers moving around when we did experiments in the shopping mall. Moreover, we conducted experiments to prove the robustness of SWiN with respect to the leaders' walking behaviours. We found that 90% of the tracking offsets are less than 2.0m in cases when the leader temporary stopped or stepped down, wagged from side to side along the trace and placed the smartphone in his/her pocket. According to our survey, most users were satisfied with the offset less than 4m.

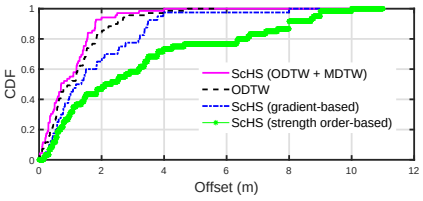


Fig. 17. Tracking offsets using different synchronization algorithms in the office building.

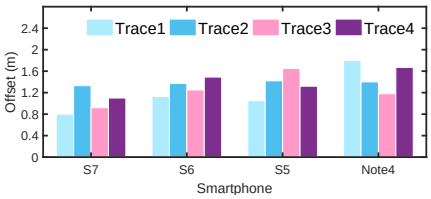


Fig. 18. Tracking offsets using different smartphones in the office building.

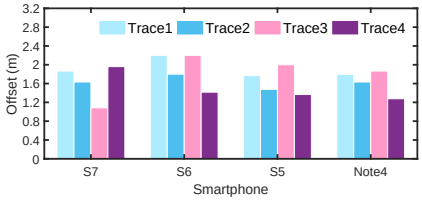


Fig. 19. Tracking offsets using different smartphones in the shopping mall.

Comparison tests. Next, we analysed the tracking performance of SWiN with different synchronization algorithms. Four synchronization algorithms were tested. They were: ScHS algorithm, ODTW algorithm, gradient-based ScHS algorithm and strength order based ScHS algorithm, in which the gradient-based ScHS algorithm uses the gradient code distance and the MAC set distance (i.e., $\gamma = 0$), while the strength order based ScHS algorithm only uses the strength order distance (i.e., $\alpha = \beta = 0$). For each algorithm, we let 10 volunteers track the reference trace A to G for more than 10 times. After they have finished, we recorded their offsets and averaged the failure rates (defined in Section 4.2). We showed the experimental results in Fig. 17 and Table 1. We can see that ScHS algorithm used in SWiN has better performance than the other synchronization algorithms. And the last two algorithms have higher failure rates, which are more than 20 %. We think this is because only a single reference trace is provided for navigation, which limits the discrimination power of positions for the gradient-based and strength order based ScHS algorithm, thus, increasing the tracking offsets and failure rates.

TABLE 1
Failure rates using different synchronization algorithms

ScHS in SWiN	ODTW	gradient-based	strength order based
7.3%	15.2%	22.1%	38%

Heterogeneous devices. We also validated the tracking performance of SWiN using different devices. In this experiment, we used four types of smartphones: Samsung Galaxy S7, Samsung Galaxy S6, Samsung Galaxy S5, and Samsung Note 4. To evaluate the impact of device diversity, we took the following experimental scheme. Firstly, four volunteers collected four different reference traces carrying these smartphones both in the office building and shopping mall. Then, for each reference trace, an volunteers used four different smartphones to track it respectively. We plotted the average offsets in Fig. 18 and Fig. 19. We can see that different smartphones may result in different offsets, but most of them are less than 1.6m (in average) in the office building and 2m (in average) in the shopping mall. The variation of the tracking offset across devices is because the execution times of the MDTW thread are different with different devices, resulting in different calibration times for the synchronization drifts caused by the ODTW thread. Moreover, the extensive experiments show that SWiN is also adaptable to smartphones produced by different manufactures other than Samsung.

5.2.4 Deviation detection

SWiN should handle the follower's deviation from the target trace and alert him/her timely. In the deviation detection experiment, we let 4 volunteers track the reference trace A to G in the office building. To evaluate the deviation-detection performance of SWiN, volunteers intentionally deviated at some pre-set points. For example, the first volunteer did not take a right turn at point B but went straight to J; the second volunteer missed the stair warning at point C; the third volunteer climbed to the fourth floor but not the third floor at D; the fourth volunteer took a wrong turn at point F to H. Every volunteer repeated the deviation event more than 10 times for each pre-set deviation detection threshold δ. We changed δ from 0.1 to 0.9. If $1 - d$ (d is the dissimilarity value calculated with the ScHS algorithm) was less than δ, SWiN warned the follower about the deviation. In experiments, volunteers ticked the timestamp when they exactly deviated the corrected path. We recorded the delay from the tick point to the warning point. In fact, it is possible that the deviation detector is incorrect, mistaking the normal matches as deviations. It depends on the value of δ.

We plotted the warning delay and the false positive rate in Fig. 20. It shows that if δ is smaller, the false positive rate is smaller as well, but the delay for detecting the deviation is longer. For example, if δ is 0.15, the warning time is 15s and the false positive rate is 0.01. But if δ is 0.85, the alert time is 3.8s and the false positive rate is nearly 100%. In SWiN, we set δ to be 0.5, because we observe that about 100% of the dissimilarity value in a satisfied alignment is less than 0.5 (i.e., $1 - d$ is bigger than 0.5).

5.2.5 Reference trace update

For traces which are visited frequently by users, we update them offline in the trace center to guarantee their long-term usability. The updating mechanism is introduced in Section 4. To validate its effectiveness, the trace center stored all users' traces from A to G in the office building. We did this experiment for nearly a month. Everyday a random number of volunteers tracked the target trace at different times and reported their offsets and failure rates to us. We calculated the average values of the offsets and failure rates everyday.

We did three comparative experiments in the same scenario. They were: 1) the reference trace was not updated; 2) the reference trace was updated only using the merging process; 3) the reference trace was updated only using the replacing process. For the first case, we plotted the tracking offsets and failure rates in Fig. 21 and Fig. 22, respectively. We can see that without updating the reference trace, the offset increases with time. After a week, it reaches more than 4m. The failure rate increases to be more than 40% as well. This indicates a terrible performance degradation after a week if we do not update the reference trace.

In the second case, we merged all the followers' traces with the reference trace. And in the third case, we asked

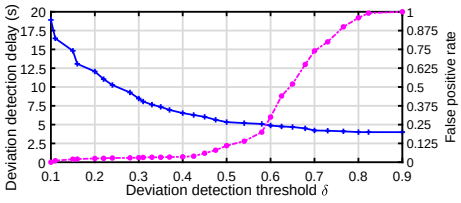


Fig. 20. The relationship of δ with the deviation-detection delay and the false positive rate.

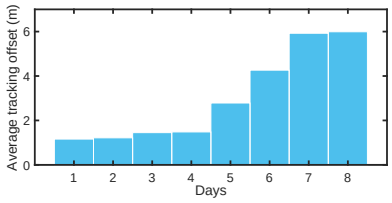


Fig. 21. The variation of the tracking offset with time.

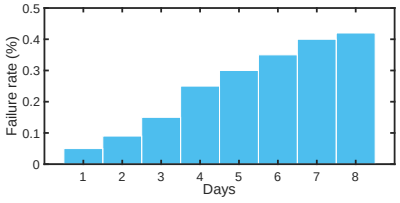


Fig. 22. The variation of the failure rate with time.

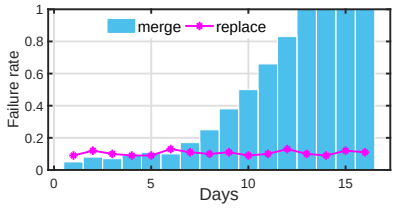


Fig. 23. The failure rates with the merging and replacing process, respectively.

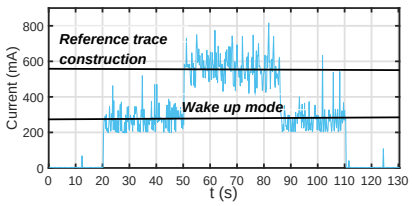


Fig. 24. Energy consumption for the construction of RTIP.

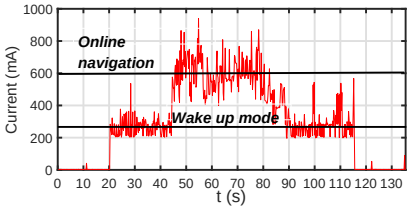


Fig. 25. Energy consumption for the real-time navigation.

4 volunteers tracked the reference trace walking steadily everyday, whose traces would be candidates for a new reference trace. Fig. 23 shows the results of these two cases. We can see that, when updating the reference trace only with the merging process, the failure rate of SWiN increases as well. But it is much better than that in the first case. The reason for the performance degradation is that the fusion errors will accumulate with time due to mismatches. After 10 days, the failure rate is more than 50%, which indicates that the reference trace can not be used anymore. Therefore, we set the time period T_u (see Section 4.2) equal to 10 to replace the reference trace in SWiN. As for the third case, the failure rate remains around 10%, which is more stable than the other two cases. However, in practice, it is not easy to find a stable trace from the followers' traces everyday. The latecomers may provide low quality traces since most of them visit the building for the first time.

5.2.6 *Energy consumption of SWiN*

Due to the limited battery energy and mobile property of the smartphone, the energy consumption of SWiN is one of the critical metrics that the user might worry about. We evaluated the energy consumption of SWiN using the Monsoon power monitor, which is a tool widely used to measure the power consumption of the mobile devices [37]. During the experiment, we turned off all background applications, but turned on the WiFi module and sensors such as gyroscope, accelerometer and barometer to collect the necessary signals.

Fig. 24 and Fig. 25 shows the current measurements when running SWiN on Samsung Galaxy S5. As the power consumption shown in Fig. 24, from 0 $\sim$ 20s the smartphone was in sleep mode. The phone was woke up at around 20s. We activated the leader activity of SWiN at 50s and built the reference trace for about 35s. Similarly, Fig. 25 shows the power consumption of the follower activity. The navigation phase was running from 45s to 90s. From 45s to around 55s, SWiN tried to lock on the start point of the follower.

We compared the energy consumption of SWiN with FollowMe and Travi-Navi [23] (image based). To avoid the impact of different implementation devices of SWiN (Samsung Galaxy S5), FollowMe (Samsung Galaxy S4) and Travi-Navi (Samsung Galaxy S2), we only focus on the incremental energy consumption when these systems start to run. As shown in Fig. 24 and Fig. 25, the average runtime currents of SWiN are 293.4mA and 334.8mA in the RTIP construction process and the navigation phase, respectively. According to the data from [24][23], we listed the energy consumption of these three systems in Table 2. We can see that the energy consumption of SWiN is lower than Travi-Navi but higher than FollowMe. This is because SWiN does not use the energy-hungry sensor such as camera but needs to turn on the WiFi module. And instead of using the computation-intensive particle filtering algorithm in Travi-Navi, SWiN uses the light-weight synchronization algorithm modified from DTW algorithm. But this two-thread ScHS algorithm consumes more energy than the signal synchronization algorithm in FollowMe. Even though SWiN is more energy-efficient, the navigation accuracy is higher in Travi-Navi (90% of the offsets are less than 1.6m). Moreover, although the runtime current of the start point detection process is much higher (i.e., 453.4mA), we can offload it to the trace center for further reducing energy consumption.

TABLE 2
Energy consumption of indoor navigation systems

System	Trace construction (Current)	Navigation (Current)
Travi-Navi	433.4mA	349.5mA
SWiN	293.4mA	334.8mA
FollowMe	224.6mA	303.4mA

6 RELATED WORK

In the recent decade, indoor navigation/localization has been extensively studied in various areas. Some researchers hold that indoor navigation is a kind of location based service (LBS). RSSI fingerprinting-based localization approaches are adopted by a large body of indoor localization systems. The main idea is that the signal signature at every location can be used to build a fingerprinting map. Gloc [27] presents a robust indoor localization system which builds a gradient map based on the fingerprinting map. Map information in this paper has high dimensions by using the up, down, right

and left relative RSSI values. Generally, fingerprinting-based localization systems always need to build a comprehensive map, which costs considerable manual efforts. We design SWiN as a plug-and-play system, it does not need a map but only a single reference trace.

There are also indoor navigation systems similarly designed with our system. In Escort [38], dead-reckoning techniques and crowd encounters' information are used to provide optimal paths for users. However, it requires predeployed audio beacons, and the navigation results calculated in the server side bring too much delay, both of which impairs the usability of the system. Travi-Navi [23] is a vision-guided system, which can help users to easily bootstrap and deploy indoor navigation services without building entire localization system. It uses the Leader-Follower structure, and employs particle filter algorithm to merge WiFi signals, geo-magnetic signals and inertial signals to provide accurate navigation. However, it is computation extensive and restricts the user to hold the smartphone vertically during walk.

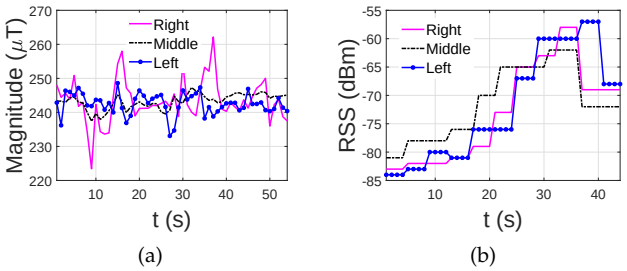


Fig. 26. The variation of signals in different sections along the same passage. (a) Geo-magnetic signal; (b) WiFi signal.

FollowMe [24] performs a lightweight synchronization algorithm to realize the last-mile navigation to compensate for the outdoor GPS navigation. FollowMe leverages the ubiquitous geo-magnetic signal. The geo-magnetic field is affected by building structures due to the presence of ferromagnetic materials. These structures may act as landmarks for object localization and tracking [24, 39–41]. As shown in Fig. 26(a), we recorded the geo-magnetic signal along the same 2m width passage in the right, middle and left section, respectively. It indicates that the signal's magnitude profiles vary differently even they are collected along the same passage. This may affect the navigation performance of the leader-follower navigation system. Suppose that a leader collects a sequence of geo-magnetic signals in the middle section along the passage. While the follower tracks the trace along the left (or right) section. The follower's trace may not match with the leader's trace due to the different signal profiles in different sections. This "vertical difference" will increase the alignment drifts, and thus degrade the navigation performance. On the contrary, the corresponding WiFi signal profiles (especially the trends) (Fig. 26(b)) are almost the same in these three sections. That is, if the leader collects the WiFi signals in the middle section, the follower's trace is consistent with the leader's trace regardless of which section he/she walks along. Hence, we exploit the variation trend of WiFi signals for navigation in SWiN.

Both SWiN and ppNav [25] make use of the WiFi signal. ppNav borrows the idea from the image processing technique. It projects the sequential WiFi fingerprints to a diagrammed form and extracts both radio and visual features of the diagram to track relative locations of the followers. However, it does not consider about the performance degradation of the reference trace with time. SWiN proposes a updating mechanism to maintain the long-term utility of this navigation system.

7 CONCLUSION AND DISCUSSION

In this paper, we presented SWiN, a real-time navigation system, which enables light-weight, plug-and-play and user-friendly navigation off-the-grid without considering a comprehensive map or a localization-assisted path planing. We leveraged the ubiquitous WiFi signal and extracted its static and dynamic properties to form a composite metric quantifying the dissimilarity between WiFi signal samples. To provide real-time, adaptive and accurate indoor navigation, we proposed a step-constrained hybrid synchronization (ScHS) algorithm based on the leader-follower structure. Along with the navigation process, motion hints like turns, level changes and deviation warnings are detected using our dedicated methods. Furthermore, we developed an updating mechanism to guarantee the long-term usability of the frequently visited traces. Finally, we implemented SWiN on commercial smartphones, and evaluated it in a five-story office building and a two-story shopping mall. Experimental results demonstrated that SWiN can provide delightful and satisfactory navigation service.

In the following, we discuss some limitations of SWiN and possible countermeasures. First, even though the navigation activity runs in the user end, the leaders' privacy may be exposed when they share their information to the trace center. We think the anonymity technique can be used for this concern. Second, currently, SWiN cannot re-route or path planning for followers if they deviate from the correct path or just want to go another way. This problem may be solved by building a "graph" for the building using the point-to-point reference traces based on the crowdsourcing technology. Third, the life time (about 11 days) of the reference trace is short now. We can consider to build a large data set which can efficiently provide personalized reference traces for specific users. Fourth, SWiN is not flexible enough to create new reference traces using existed traces. In fact, through cutting and/or stitching the traces, we can create more reference traces for the navigation service [44] if we have collected enough traces, which definitely improves the usability of our system.

REFERENCES

[1] W. Li and D. Wei and H. Yuan and G. Ouyang, "A novel method of WiFi fingerprint positioning using spatial multi-points matching," *in Proc. IEEE Int. Conf. Indoor Positioning Indoor Navigation*, 2016, pp. 1-8.

[2] C. Luo and L. Cheng and M. C. Chan and Y. Gu and 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.

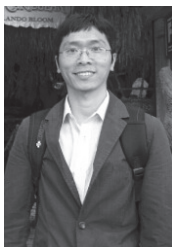
[3] D. Vasisht and S. Kumar and D. Katabi, "Decimeter-Level Localization with a Single WiFi Access Point," *in Proc. 13th USENIX Symp. Netw. Syst. Design Implement.*, 2016, pp. 165-178.

[4] F. S. Danis and A. T. Cemgil, "Model-Based Localization and Tracking Using Bluetooth Low-Energy Beacons," *Sensors*, vol. 17, no. 11, pp. 2484, Nov. 2017.

[5] W. Huang and Y. Xiong and X. Y. Li and H. Lin and X. Mao and P. Yang and Y. Liu, "Shake and walk: Acoustic direction finding and fine-grained indoor localization using smartphones," *in Proc. 33rd IEEE Int. Conf. Comput. Commun.*, 2014, pp. 370-378.

[6] K. Liu and X. Liu and X. Li, "Guoguo: Enabling fine-grained smartphone localization via acoustic anchors," *IEEE Trans. Mobile Comput.*, vol. 15, no. 5, pp. 1144-1156, May 2016.
[7] Y. Xu and Y. S. Shamaliy and Y. Li and X. Chen, "UWB-Based Indoor Human Localization With Time-Delayed Data Using EFIR Filtering," *IEEE Access*, vol. 5, no. 5, pp. 16676-16683, May 2017.
[8] H. Xie and T. Gu and X. Tao and H. Ye and J. Lu, "A reliability-augmented particle filter for magnetic fingerprinting based indoor localization on smartphone," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1877-1892, Aug. 2016.
[9] V. Pasku and A. De Angelis and G. De Angelis and D. D. Arumugam and M. Dionigi and P. Carbone and D. S. Ricketts, "Magnetic Field Based Positioning Systems," *IEEE Commun. Surve. Tutor.*, vol. 5, no. 8, pp. 16676-16683, Aug. 2017.
[10] S. Liu and T. He, "Smartlight: Light-weight 3d indoor localization using a single led lamp," *in Proc. 15th ACM Conf. Embed. Netw. Sensor Syst.*, 2017, pp. 1-14.
[11] S. Yoon and K. Lee and Y. Yun and I. Rhee, "Acmi: Fm-based indoor localization via autonomous fingerprinting," *IEEE Trans. Mobile Comput.*, vol. 15, no. 6, pp. 1318-1332, June 2016.
[12] C. Xu and B. Firner and Y. Zhang and R. E. Howard, "The case for efficient and robust rf-based device-free localization," *IEEE Trans. Mobile Comput.*, vol. 15, no. 9, pp. 2362-2375, Sep. 2016.
[13] A. Rai and K. K. Chintalapudi and V. N. Padmanabhan and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," *in Proc. 18th ACM int. conf. Mobile computing netw.*, 2012, pp. 293-304.
[14] C. Zhang and K. P. Subbu and J. Luo and J. Wu, "GROPING: Geomagnetism and crowdsensing powered indoor navigation," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 387-400, Feb. 2015.
[15] C. Zhou and Y. Gu and S. He and Z. Shi, "A Robust and Efficient Algorithm for Coprime Array Adaptive Beamforming," *IEEE Trans. Vehicular Tech.*, vol. 67, no. 2, pp. 1099-1112, Feb. 2018.
[16] J. Chen and K. Hu and Q. Wang and Y. Sun and Z. Shi and S. He, "Narrowband Internet of Things: Implementations and Applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2309-2314, Dec. 2017.
[17] X. Zhang and Z. Yang and W. Sun and Y. Liu and S. Tang and K. Xing and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Commun. Surve. Tutor.*, vol. 18, no. 1, pp. 54-67, Jan. 2016.
[18] S. H. Jung and B. C. Moon and D. Han, "Unsupervised learning for crowdsourced indoor localization in wireless networks," *IEEE Trans. Mobile Computing*, vol. 15, no. 11, pp. 2892-2906, Nov. 2016.
[19] S. Yang and P. Dessai and M. Verma and M. Geria, "FreeLoc: Calibration-free crowdsourced indoor localization," *in Proc. 32nd IEEE Int. Conf. Comput. Commun.*, 2013, pp. 2481-2489.
[20] G. Yang and S. He and Z. Shi and J. Chen, "Promoting Cooperation by the Social Incentive Mechanism in Mobile Crowdsensing," *IEEE Commu. Magazine*, vol. 55, no. 3, pp. 86-92, March 2017.
[21] L. Li and G. Shen and C. Zhao and T. Moscibroda and J. H. Lin and F. Zhao, "Experiencing and handling the diversity in data density and environmental locality in an indoor positioning service," *in Proc. 20th ACM int. conf. Mobile computing netw.*, 2014, pp. 459-470.
[22] Y. Oh and W. L. Kao and B. C. Min, "Indoor Navigation Aid System Using No Positioning Technique for Visually Impaired People," *in Proc. the 19th Int. Conf. Human.-Comput. Interact.*, 2017, pp. 390-397.
[23] Y. Zheng and G. Shen and L. Li and C. Zhao and M. Li and F. Zhao and M. Li, "Travi-navi: Self-deployable indoor navigation system," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2655-2669, May 2017.
[24] Y. Shu and K. G. Shin and T. He and J. Chen, "Last-mile navigation using smartphones," *in Proc. 21th ACM int. conf. Mobile computing netw.*, 2015, pp. 512-524.
[25] Z. Yin and C. Wu and Z. Yang and Y. Liu, "Peer-to-Peer indoor navigation using smartphones," *IEEE J. Selec. Areas Commun.*, vol. 35, no. 5, pp. 1141-1153, May 2017.
[26] Q. Roy and S. T. Perrault and S. Zhao and R. C. Davis and A. Pattena Vaniyar and V. Vechev and A. Misra, "Follow-My-Lead: Intuitive Indoor Path Creation and Navigation Using Interactive Videos," *in Proc. CHI Conf. Human Factors Comput. Syst.*, 2017, pp. 5703-5715.
[27] Y. Shu and Y. Huang and J. Zhang and P. Coue and P. Cheng and J. Chen and K. G. Shin, "Gradient-based fingerprinting for indoor localization and tracking," *IEEE Trans. Indust. Electronics* , vol 63, no. 4, pp. 2424-2433, Apirl 2016.
[28] M. Paterson and V. Dancik, "Longest common subsequences," *Springer Math. Foundat. Comput. Sci.*, vol. 841, pp. 127-142, Aug. 1994.
[29] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," *in Proc. SIGKDD Conf. Knowledge Disc. Data Mining Workshop*, 1994, pp. 359-370.
[30] G. Burloiu, "An online audio alignment tool for live musical performance," *in Proc. 11th IEEE Int. Symp. Electro. Telecommun.*, 2014, pp. 1-4.
[31] X. Zhu and W. S. Yoo, "Suggested new element reference frame for dynamic analysis of marine cables," *Springer Nonlinear Dynamics*, vol. 87, no. 1, pp. 489-501, Jan. 2017.
[32] K. Djamel and M. Abdellah and A. Benallegue, "Attitude Optimal Backstepping Controller Based Quaternion for a UAV," *Hindawi Math. Problems Engineering*, vol. 2016, no. 8573235, pp. 1-11, Feb. 2016.
[33] M. Euston and P. Coote and R. Mahony and J. Kim and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," *in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 340-345.
[34] H. Ye and T. Gu and X. Zhu and J. Xu and X. Tao and J. Lu and N. Jin, "FTrack: Infrastructure-free floor localization via mobile phone sensing," *in Proc. IEEE Int. Conf. Pervasive Comput. Commu. (PerCom)*, 2012, pp. 1-9.
[35] H. Ye and T. Gu and X. Tao and J. Lu, "Scalable floor localization using barometer on smartphone," *Wireless Commu. Mobile Comput.*, vol. 16, no. 1, pp. 2557-2571, Nov. 2016.
[36] Taylor W., "A pattern test for distinguishing between autoregressive and mean-shift data," *Taylor Enterprises, Libertyville, Illinois. Web: http://www. variation.com/cpa/tech/changepoint.html*, 2000, pp. 1-14.
[37] R. E. Schwartz., "How to Power Profile A Mobile App Using Hardware," *https://mostly-tech.com/tag/monsoon-power-monitor/*, 2015.
[38] I. Constandache and X. Bao and M. Azizyan and R. R. Choudhury, "Did you see bob?: human localization using mobile phones," *in Proc. 16th ACM int. conf. Mobile computing netw.*, 2010, pp. 149-160.
[39] B. Gozick and K. P. Subbu and R. Dantu and T. Maeshirp, "Magnetic maps for indoor navigation," *IEEE/ACM Trans. Instrument. Measur.*, vol. 60, no. 12, pp. 3883-3891, Dec. 2011.
[40] W. Storms and J. Shockley and J. Raquet, "Magnetic field navigation in an indoor environment," *in Proc. IEEE Ubiquitous Posit. Indoor Navi. Loc. Servi.*, 2010, pp. 1-10.
[41] G. Shen and Z. Chen and P. Zhang and Moscibroda P. and Y. Zhang, "Walkie-markie: indoor pathway mapping made easy," *in Proc. 10th USENIX conf. Netw. Syst. Design Implement.*, 2013, pp. 85-98.
[42] Y. Zhuang and Z. Syed and Y. Li and N. El-Sheimy, "Evaluation of two WiFi positioning systems based on autonomous crowdsourcing of handheld devices for indoor navigation," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1982-1995, Aug. 2016.
[43] Z. Shi and Z. Zhang and Y. Shu and P. Cheng and J. Chen, "Demo abstract: Indoor navigation leveraging gradient wifi signals," *in Proc. 15th ACM Conf. Embed. Netw. Sensor Syst.*, 2017, pp. 1-2.
[44] H. Wen and Y. Shen and S. Papaioannou and W. Churchill and N. Trigoni and P. Newman, "Opportunistic Radio Assisted Navigation for Autonomous Ground Vehicles," *in Proc. IEEE int. conf. Distributed Comput. Sensor Syst. (DCOSS)*, 2015, pp. 21-30.

Zhenyong Zhang is a fourth year Ph.D. candidate in the school of control science and engineering, Zhejiang University, Hangzhou, China. He received his bachelor degree in control science and engineering from Central South University, Changsha, China, in 2015. His research interests include mobile computing and cyber-physical system security.



Shibo He (M13) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2012. He was an Associate Research Scientist from March 2014 to May 2014, and a postdoctoral scholar from May 2012 to February 2014, with Arizona State University, Tempe, AZ, USA. He is currently a Professor at Zhejiang University. His research interests include wireless sensor networks, crowdsensing, and big data analysis. He is a recipient of the IEEE Asia-Pacic Outstanding Researcher Award in 2015.

Yuanchao Shu (M15-SM19) is currently a Researcher with Mobility and Networking Research Group at Microsoft Research. His research interests lie broadly in mobile and wireless systems, networked control and optimization, and mobile security and privacy. His previous research results have been published at top-tier venues including MobiCom, JSAC, TMC, USENIX Security etc. He won IEEE WCNC Best Paper Award, ACM SenSys Best Paper Runner-up Award and IEEE INFOCOM Best Demo Award.

Zhiguo Shi (M10-SM15) received the B.S. and Ph.D. degrees in electronic engineering from Zhejiang University, Hangzhou, China, in 2001 and 2006, respectively. Since 2006, he has been a Faculty Member with the Department of Information and Electronic Engineering, Zhejiang University, where he is currently a Full Professor. His current research interests include signal and data processing, and smart grid communication and network.