Last-Mile Navigation Using Smartphones

Yuanchao Shu[⋆†], Kang G. Shin[†], Tian He[‡], Jiming Chen[⋆]

[⋆] Zhejiang University, Hangzhou, China
[†] The University of Michigan, Ann Arbor, USA
[‡] University of Minnesota, USA

ABSTRACT

Although GPS has become a standard component of smartphones, providing accurate navigation during the last portion of a trip remains an important but unsolved problem. Despite extensive research on localization, the limited resolution of a map imposes restrictions on the navigation engine in both indoor and outdoor environments. To bridge the gap between the end position obtained from legacy navigation services and the real destination, we propose FOLLOWME, a "last-mile" navigation system to enable plug-and-play navigation in indoor and semi-outdoor environments. FOLLOWME exploits the ubiquitous, stable geomagnetic field and natural walking patterns to navigate the users to the same destination taken by an earlier traveler. Unlike existing localization and navigation systems, FOLLOWME is infrastructure-free, energy-efficient and cost-saving. We implemented FOLLOWME on smartphones, and evaluated it in a four-story campus building with a testing area of $2000m^2$. Our experimental results with 5 users show that 95% of spatial errors during navigation were 2m or less with at least 50% energy savings over a benchmark system.

Categories and Subject Descriptors

C.3 [**Special-purpose And Application-based Systems**]: Real-time and embedded systems; H.5.2 [**Information Interfaces And Presentation**]: User Interfaces—*User-centered design*

General Terms

Design; Experimentation; Performance

Keywords

Navigation; Smartphone; Plug-and-Play System; Sensory Data

1. INTRODUCTION

Location-relevant services and applications have been studied extensively within the mobile computing community over the past several years. Most researchers have focused on the localization of devices/users. To date, with the help of space satellites, meter-level positioning accuracy can be achieved in outdoor environments (e.g., using GPS, GLONASS, etc.), and numerous approaches have been proposed to localize users in indoor environments [1–7].

Accurate localization, map information and path planning algorithms make real-time navigation a reality, but most research efforts on navigation have been designed for path planning and optimization, given starting and destination locations and wide-area map information. Due to the lack of micro-map information, however, navigation cannot always be successful, especially for the last portion of a trip. For example, Google Map is able to navigate users to a building (more specifically, an entrance of the building) from a location tens of miles away, but fails to find a feasible path to the final destination, e.g., a meeting room inside the building. We call this imperfection the *last-mile* navigation problem.[1]

The main reason for this problem is the incomplete map information, causing incorrect inference on existence and reachability of an exact destination. Technically, most navigation services can guide a user only to a place that is connected by at least one indexed path, such as a trunk road. If the end location is not in the map database (even if it is accessible in real world), navigation systems cannot connect the coordinates of two locations.

Despite the extensive efforts made on indoor localization, few of the resulting solutions have been deployed because of their labor-intensive (hence time-consuming) bootstrapping, especially for indoor map construction. Although multiple model-based [5, 8] and crowdsourcing [6, 9, 10] techniques have been proposed, the need for precise building structure information and a sustainable incentive crowdsourcing mechanism limits their applicability [11]. Even if deployed, indoor localization systems may face an onerous calibration process (for those radio-based fingerprint systems) and need to deploy path-planning algorithms to enable online navigation services. For example, Apple iBeacon can perform localization (*not* navigation) only in the vicinity of pre-deployed iBeacon devices, whereas Path Advisor [12] only provides offline navigation paths with given start and end locations.

To meet the ever-growing demand for navigation, we propose a new lightweight, plug-and-play, last-mile navigation system, called FOLLOWME, to bridge the gap between the user's final destination and the end location provided by current navigation services. The main idea of FOLLOWME is to use "scent" or "crumbs" left behind by previous travelers (a.k.a. leaders). In FOLLOWME, A leader records sensory data with his/her smartphone during the last-mile trip. The location specific features extracted from the sensory data are combined with the leader's walking patterns (e.g., steps, turns, going upstairs/downstairs) to build a reference trace. In

MobiCom'15, September 7–11, 2015, Paris, France.

DOI: http://dx.doi.org/10.1145/2789168.2790099.

[1]The *last mile* was originally used in the field of communications to refer to the final leg of communications connectivity to retail customers. The last mile is often the speed bottleneck in communication networks.

the design of FOLLOWME, we adopt the use of the ubiquitous geomagnetic field, which is locally disturbed in indoor and semi-outdoor environments (i.e., close to a building or in a semi-open building, as in a metropolitan area populated with tall buildings) [13–16]. We also consider how to incorporate other location-specific signals including FM and GSM in Section 8.

During the navigation phase, the follower will first arrive at the same starting location (e.g., a building entrance) as at least one leader using legacy navigation services (e.g., Google Maps). From then on, FOLLOWME installed in the follower's smartphone will compare and synchronize current sensor readings with the reference (i.e., the leader's) trace and guides the user, in real time, from the starting location to the final destination. This way, irrespective of incomplete map information, FOLLOWME is able to navigate users[2] to any point of interest (PoI) as long as it has been visited earlier by a leader. For example, a meeting coordinator can provide the attendees with the data trace from the building entrances to the meeting room. Vendors and restaurant owners can collect data traces on their own from several entrances of a shopping mall to their stores and then share them with the public on the cloud to entice customers. The leader and the follower can also be the same person. For example, one can record a trace from a parking spot to the airport terminal and use it for reverse navigation back to one's car after a multi-day trip.

Unlike other leader–follower navigation systems [17–19], FOLLOWME relies neither on an infrastructure (e.g., maps, WiFi APs) nor any additional hardware (e.g., beacons, landmarks). In addition, FOLLOWME exploits people's natural walking patterns, thus minimizing constraints imposed on users.

This paper makes the following three main contributions:

- We identify the last-mile navigation problem and propose an infrastructure-free solution using only sensors on commodity smartphones;
- We devise a novel online magnetic-field-based, step-constrained trace synchronization technique for walking progress estimation and navigation; and
- We implement FOLLOWME on Android smartphones and evaluate it in a four-story campus building with 5 participants. 95% of spatial errors during navigation within 2m is achieved with at least 50% energy savings over a benchmark system.

The remainder of this paper is organized as follows. We first provide motivation for a new indoor navigation system like FOLLOWME in Section 2. We then present the architecture of FOLLOWME in Section 3 and detailed illustrations in Section 4 and Section 5. Section 6 describes the implementation of a prototype system, and Section 7 presents an in-depth evaluation of FOLLOWME. Several practical issues and related work are discussed in Section 8 and Section 9, respectively. Finally, we conclude the paper in Section 10.

2. MOTIVATION

Given that various sensors and methods have been studied or proposed for navigation, why is another navigation system like FOLLOWME necessary? We justify the need for FOLLOWME by answering a few natural questions as follows.

- *Is GPS good enough for navigation?* In most outdoor scenarios, the answer is yes. However, as stated in the Introduction, GPS-like satellite systems offer high positioning accuracy, but the map's insufficiency limits their potential to explore the last-mile navigation and impairs the user experience. GPS also consumes lots of power, suffers from the urban canyon effect, and has poor coverage in indoor environments.
- *Are guide maps sufficient for navigation in an indoor environment?* Guide maps are helpful for indoor navigation, but they are not always handy. Users also need to fully understand the guide map, pinpoint the starting point and the destination, plan a route, and remember the route. This imposes additional burden on the users, especially on the elderly or children. Thus, FOLLOWME is complementary to the guide map with minimal learning cost to users.
- *Is indoor localization helpful?* Yes, but localization is not the same as navigation. Most indoor localization systems (ILSes) are built based on full knowledge of a floor map. In essence, insufficient incentives for map construction and time-consuming bootstrapping hinder their deployment and acceptance. Moreover, in many cases, a complicated and large ILS is an overkill for navigation needs. In other words, users have to bear the additional cost of the entire map construction, even they only need guidance along a single path. Therefore, a lightweight and plug-and-play navigation system such as FOLLOWME is necessary to fill the gap between high navigation demands and limited ILSes deployments.

[2]We use followers and users interchangeably throughout the paper.

3. SYSTEM OVERVIEW

This section provides an overview of FOLLOWME, first presenting the system architecture in Section 3.1 and then a navigation example in Section 3.2 to illustrate how FOLLOWME operates.

3.1 System Architecture

FOLLOWME consists of a trace-collection module and a navigation module, both of which exploit multiple sensors in a smartphone, such as accelerometer, gyroscope, magnetometer, barometer, etc. Figure 1 shows the overall architecture of FOLLOWME.

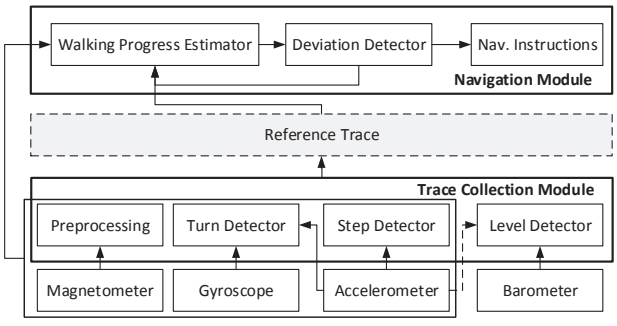


Figure 1: Architecture of FOLLOWME.

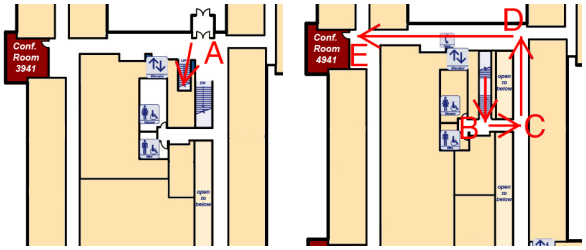
Trace-Collection Module: This module works during the leader's walking trip to generate the reference trace. The leader activates the module when s/he starts walking. During the leader's entire walking period, FOLLOWME records the readings of the magnetometer, accelerometer, gyroscope, and barometer. The module then launches a series of signal processing, including magnetic data preprocessing (Section 5.2), step recognition (Section 6.1), turn and level-change detection (Section 6.2 and Section 6.3), and generates a reference trace after the leader's arrival at the destination. The

reference trace contains timestamped preprocessed geomagnetic data as well as steps, turns and level changes extracted from different sensors.

Note that we can also leverage combinations of sensors other than those listed in Figure 1 for walking pattern recognition. For example, together with a barometer, an accelerometer can help detect whether or not the user is climbing stairs [20]. Due to space limitation, here we focus on the design of the navigation module and briefly mention the implementation of recognition and detection algorithms in Section 6.

Navigation Module: This module takes the reference trace as an input and helps the followers navigate from the starting location to the same destination as the leader. Typical starting locations are those where GPS/Google Map stops provision of precise navigation, such as the end of a road or the entrance of a building. Like the trace-collection module, it contains a mag-processor and step detector that process the data from magnetometer and accelerometer, respectively, during the follower's walking trip. All the input data are sent to the walking progress estimator (Section 5.2), which estimates the portion the user has walked relative to the entire reference trace. Based on the estimation results, a deviation detector determines whether or not the follower is off course. If not, the navigation module provides navigation directions (i.e., turn, go up/down stairs) on the smartphone screen to guide the user to the destination, or reversely guides the user back to the correct trace and then to the destination.

3.2 A Navigation Example



(a) The 3rd floor. (b) The 4th floor.

Figure 2: An indoor navigation example.

For example, suppose a meeting will be held in Room 4941 on the fourth floor of our Department building, and U_a is the coordinator of this meeting. Before the meeting, U_a walks from the building entrance (location A on the third floor) to the conference room as the red arrowed line shows. During the leader's walk, the trace-collection module records the geomagnetic data along the route and identifies walking features (i.e., 1 flight of stairs and 3 left turns). Later, a meeting participant (follower) U_b, who has received the reference trace from U_a, arrives at the same entrance. From that point on, the navigation module will estimate U_b's location relative to the reference trace (i.e., walking progress) in real time and guide U_b to Room 4941. The walking progress estimation is made by matching geomagnetic observations of U_b to those from the reference trace. Based on the estimation result, a "go upstairs" icon will appear first, followed by a "left turn" icon when U_b is approaching locations B, C and D. Once U_b reaches location E, FOLLOWME generates a notification and terminates the navigation process.

Figure 3 shows how the navigation module works in the time domain. Suppose U_a takes t_1 seconds to arrive at position B

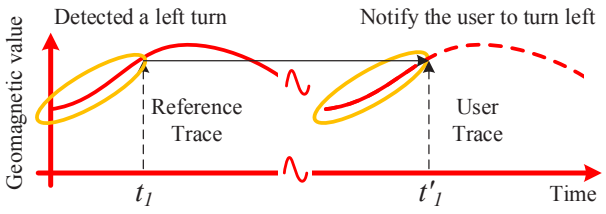


Figure 3: Illustration of the walking progress estimation.

on the path and makes a left turn during the trace collection. In the navigation phase, based on the similarity of geomagnetic observations (in both ellipses), the navigation module infers that U_b reaches B at time t'_1, and hence instructs U_b to turn left. In Section 4, we first introduce the reference trace construction and then elaborate on the navigation module in Section 5.

3.3 Assumptions and Limitations

As illustrated with the above example, FOLLOWME has a very different operational model from most existing navigation systems. Instead of enabling generic navigation, FOLLOWME is built as a plug-and-play system for dedicated working scenarios, which inevitably introduces several limitations. First, the user cannot be navigated without a reference trace with the same origin and destination. This may in turn cause a scalability problem as the number of possible routes inside a building could be very large. In this part of the paper, we first focus on the design of FOLLOWME, and discuss the scalability issue in Section 8.2. Second, due to the unavailability of map information, the user, if deviated from the reference route, has to return to the route while using FOLLOWME. Even though it is annoying in certain circumstances, the novelty of this mechanism is to enable navigation without expensive map building and localization. In addition, FOLLOWME has some other specific requirements, e.g., the algorithm requires that the difference between the number of steps taken by the leader and the follower be relatively stable. In situations where such constraints are not met, the performance and accuracy of FOLLOWME is likely to be adversely affected. This is described in more detail in Section 5.2. In Section 8, we will discuss future potential improvements based on our evaluation results.

4. REFERENCE-TRACE CONSTRUCTION

We now describe the reference-trace construction and then introduce the geomagnetic data processing and real-time navigation in Section 5. Section 6 will detail the design of algorithms for step recognition as well as turn and level-change detection.

Table 1: Data format of the reference trace

Geomagnetic data	$m =< t_i, m_i^x, m_i^y, m_i^z >$
Step data	$s =< t_i, s_i >, s_i = 1, 2, 3, \ldots$
Turn data	$tr =< t_i, tr_i >, tr_i =$ turn degrees
Level change data	$l =< t_i, l_i >, l_i = \{up, down\}$

During the leader's walking, the trace-collection module automatically builds the reference trace by (i) recording geomagnetic readings, which are tuples of timestamps and magnetic values; (ii) detecting steps, turns and level-changes based on readings of the accelerometer, gyroscope and barometer; (iii) outputting all detection results (time-stamps, turn directions, etc.) and geomagnetic

readings to a file upon leader's arrival at the destination. The data format of the reference trace is summarized in Table 1. Each reference trace is labeled with a (starting point, destination) pair (e.g., from Entrance A to Room 4941). Once the reference trace is generated, the leader can share it on the cloud for future navigation purposes. In the current FOLLOWME implementation, we send the trace file directly to the followers, but it can be deployed as a cloud service in future.

Note that there could be certain detection delays during use of FOLLOWME. For example, a peak detection algorithm will incur a step detection delay; turns can only be recognized after the user completes each turn and the response time of level-change detection is subject to the user's climbing speed of the staircase. However, since each step and turn lasts for a short time and the level-change detection works only in the trace-collection module, these delays affect the navigation performance insignificantly, which is confirmed by our experimental study.

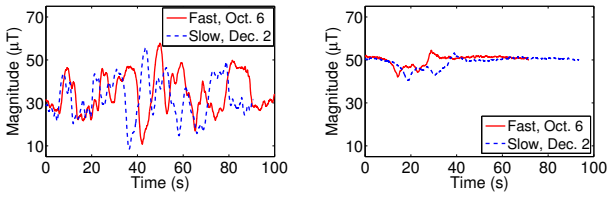
5. REAL-TIME NAVIGATION

Since followers will likely walk at different speeds from the leader, and may even stop en route from time to time, the key challenge in FOLLOWME is to estimate their walking progress. In the navigation phase, the walking progress estimator synchronizes each follower's walking to the reference trace by matching their geomagnetic observations. Based on the estimation results, real-time guidance (i.e., advance notices of turns and level-changes) and an estimated walking time are also provided on the smartphone screen. Section 5.3 further discusses on how FOLLOWME recovers when the user goes off the trail.

5.1 Preliminaries of the Geomagnetic Field

Below we discuss the identified properties of magnetic fields before presenting a walking progress estimation algorithm.

Locally Disturbed Yet Stable Magnetic Field: Due to its global availability and stability, the geomagnetic field is used in several indoor localization systems [13–15, 21–23]. Most of them exploit magnetic field anomalies caused by the local disturbances of ferromagnetic building materials, and localize users/devices based on a pre-established fingerprint map.



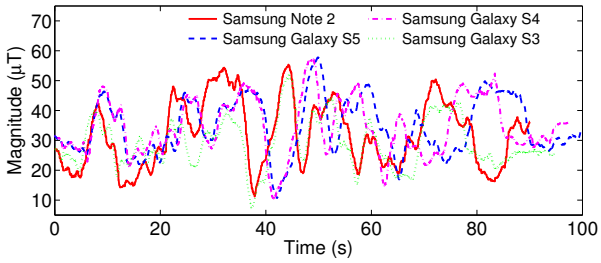
(a) Collected during walk on an indoor corridor.

(b) Collected during walk on an outdoor walking trail. The user passes a building 5m away between 10s and 40s.

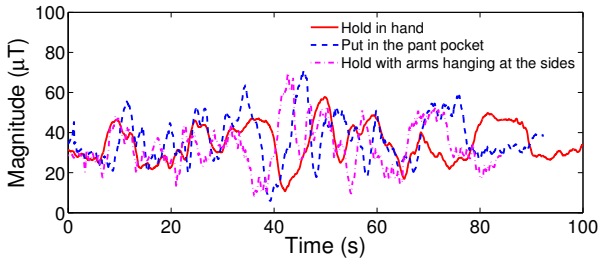
Figure 4: Geomagnetic traces collected during walking.

Figure 4 shows 4 magnetic traces in both indoor and outdoor environments. The traces in each figure were collected during walks along the same path on different dates nearly two months apart. Figure 4 shows that anomalies of the geomagnetic field caused by building construction materials exist in both indoor and outdoor environments and that the magnetic field and the local disturbances are very stable over time as the construction layout remains unchanged. In addition, the impact of mobile objects on the magnetic field is very limited. See [13, 24–27] for more detail. These local disturbances, stability over time and robustness make the magnetic field a good candidate for trace synchronization in FOLLOWME.

Note that although the magnetic field is directional and 3D magnetic signals are measured by magnetometer, it is difficult to fully leverage magnetic field readings on three axes, as the frame of reference of the magnetometer may not always align with the global coordinate system. Ensuring that alignment would require either accurately tracking the device's attitude (i.e., orientation or posture) at all times or to constrain the device usage at some fixed attitude (e.g., hand-held horizontally with Y-axis towards heading direction). The former is difficult due to sensor drift and the latter greatly affects users' experiences. Therefore, only the magnitude of the magnetic signal may be used in practice.



(a) Device diversity



(b) Usage diversity

Figure 5: Challenges of using the geomagnetic field.

Outdoor Low Discernibility and Biased Measurements Due to Device and Usage Diversities: Despite its favorable properties, use of the magnetic field poses several challenges. First, the magnetic field is less interfered in outdoor space than indoor space. For example, in Figure 4(b), outdoor magnetic distortions can only be observed when the user passes by buildings. To handle this problem, FOLLOWME uses a simplified IODetector [16] (only using the magnetism detector) to differentiate outdoor open areas from indoor/semi-outdoor spaces. In indoor or semi-outdoor environments, FOLLOWME leverages a novel step-constrained trace synchronization approach for navigation, while in outdoor open spaces where accurate directions can be obtained by a compass, it uses legacy inertial-sensor-based tracking (i.e., dead reckoning) [1]. Since the majority of last-mile navigation problems occur in indoor/semi-outdoor environments, we will focus on the magnetic-field-based, step-constrained trace synchronization.

The second challenge is the biased magnetic field measurements caused by device and usage diversities. In other words, different devices will show different readings for the same magnetic

field. This is verified in Figure 5(a), showing the magnitude of the collected magnetic signals along the same path using different smartphones. Even for the same device (Galaxy S5), the resulting signal varies from the device's height (as shown in Figure 5(b)). However, the trend (i.e., shape) of the measured magnetic field remains consistent across devices and usages, providing opportunities for navigation.

5.2 Walking Progress Estimation

To provide real-time navigation instructions (i.e., for turns, stair-climbing) and estimate the remaining walking time, we need to know how far the follower has walked relative to the length of the reference trace. For this, we devise a step-constrained trace synchronization algorithm based on *Dynamic Time Warping* (DTW). The algorithm matches magnetometer readings from the follower's phone to the pre-loaded magnetic field data from the reference trace.

DTW is a class of algorithms proposed to align and measure the similarity between two time series. Specifically, DTW matches each sample in one time series to one or more samples in another ordered sequence using dynamic programming. The objective of DTW can be stated as: *Given two time series* $S_a = S_a[i], i = 1, \ldots, L_a$ *and* $S_b = S_b[i], i = 1, \ldots, L_b$, *DTW aims to find a monotonic mapping function* $f : I[1, L_a] \rightarrow I[1, L_b]$ *between* S_a *and* S_b *so as to minimize* $\sum_{i=1}^{L_a}(S_a[i] - S_b[f(i)])^2$ *where* $I[1, L_a]$ *is the integers from 1 to* L_a.

Despite its wide usage, DTW cannot be directly applied to synchronize the user and the reference traces for three reasons. First, DTW works offline on two given time series. However, in FOLLOWME, the length of the user trace continuously increases during his walk. Second, DTW maintains a two-dimensional warping cost matrix of which the size is a quadratic function of the number of samples. This incurs an unacceptably high computational overhead on the smartphone, especially when the walking distance gets longer. Third, DTW compares two sequences based on their absolute values. Due to the device and usage diversities, the observations of biased magnetic fields have negative effects on the matching results. Due to the lack of a full picture of the user trace (especially at the beginning of the walk), simple mean-removal techniques cannot solve the problem.

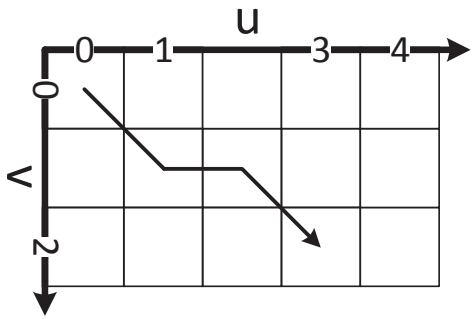


Figure 6: Updating the warping cost matrix in the step-constrained trace synchronization.

To exploit the advantages of the magnetic field and deal with the above problems, FOLLOWME proposes a step-constrained trace synchronization algorithm. The algorithm has three main characteristics. First, to deal with device heterogeneity and usage diversity, we filter out the high-frequency components of the magnetic field sensing and then utilize the differential magnetic field information that is independent of the absolute values. Second, to reduce the computation overhead, we set a global path constraint in DTW based on the step-detection results. This way, the algorithm runs in linearithmic time and we need to maintain similarity scores only within a certain range of the warping cost matrix, which is also memory-efficient. Third, to obtain a better matching result, we adaptively change the search band in the warping cost matrix according to the similarities among traces.

Let us consider Figure 6 as an example to illustrate the matching process. Figure 6 shows a 3×5 warping cost matrix. Entry (i, j) in the matrix indicates the similarity between the $(i-1)$-th sample in the reference trace v and the $(j-1)$-th sample in the user trace u. In this example we maintain a search band with only a fixed width of 2, hence setting the path constraint coefficient $c = 2$. At the beginning of the process, we compute values of the first $c^2 = 4$ entries on the northwest corner and set the pointer of the matrix $p = (p_v, p_u)$ to $(1, 1)$. Then, we find the entry with the minimal value among all entries that are within the distance of $c = 2$ to the pointer on both p_v-th row and p_u-th column. For example, we find the entry $(1, 1)$ is smaller than both $(1, 0)$ and $(0, 1)$ entries. Therefore, both p_v and p_u increase from 1 to 2 and $(1, 2)$, $(2, 1)$ and $(2, 2)$ entries are being calculated. Now, we find the entry $(1, 2)$ contains the smallest value among $(2, 1)$, $(2, 2)$ and $(1, 2)$ entries. Hence, in the next iteration, only p_u increases (i.e., the pointer moves to the right entry $(2, 3)$) and we update $(2, 3)$ and $(1, 3)$ entries.

In this example, all calculated entries are shaded in grey, and the mapped results are also labeled. For example, $f(u_3) = v_2$. This way, we synchronize the walks of the leader and the follower, and can further provide real-time turning and stair climbing instructions when the follower approaches a crossing or stairs or elevators.

Note that in this example, we move the matrix pointer p based only on matching results. A step constraint is also used in our algorithm. The rationale behind the step-constraint is simple: *the difference between the number of steps taken by the leader and the follower within a small given physical distance should be bounded.* For example, for a 5-meter walk, it is unlikely that the difference of steps taken by two users is as large as 10. Hence, in the algorithm, if $f(u_i) = v_m$ and $f(u_j) = v_n$, we first get the number of steps s'^v taken by the leader between t_m^v and t_n^v, and compare it with the number of steps s'^u taken by the follower between t_i^u and t_j^u. Since samples (u_i, v_m) and (u_j, v_n) should be taken at nearby physical positions (if not the same), the difference between s'^u and s'^v should be bounded. Therefore, if the difference is larger than a predefined threshold, we move the pointer horizontally/vertically in the next iteration.

Algorithm 1 and 2 illustrate the detailed step-constrained online trace synchronization algorithm. At first, Algorithm 1 imports reference trace v and preprocesses the magnetic field trace m^v. The function $MagPreprocess$ computes the magnitude, smooths the data through a low-pass filter and calculates the difference between neighboring magnitude values. The matching continues running before the user reaches the destination (lines 5–28). In the main loop, if a new column was calculated ($p_u^{inc} == true$) in the previous iteration, the algorithm calculates and outputs the matching result uv, and then reads a new sample (lines 8–12). Between line 14 and 27, the algorithm calculates a partial row or column of the warping cost matrix D (i.e., compare magnetic field values and compute similarities). The computation is based on standard DTW recursion formula (line 18 and line 25), restricted to using only the entries which have already been computed. The search band is updated in each iteration according to the function $DirInc$.

Input : reference trace $v = \{t^v, m^v, s^v, tr^v, l^v\}$
Output: uv where $uv_i = j$ indicates the i-th sample in u maps to the j-th sample in v

1 $p_u = 0, p_v = 0, p_u^{inc} = true$;
2 **for** *each sample* $m_i^v \in m^v$ **do**
3 $\quad m_i^{vp}$ = `MagPreprocess` (m_i^v);
4 **end**
5 **while** $p_v \leq size(v)$ **do**
6 $\quad$ **if** p_u^{inc} **then**
7 $\quad\quad uv_i = \arg\min D[k][p_u], k \in [p_v - c, p_v]$;
8 $\quad\quad$ **for** *each observation* u_i **do**
9 $\quad\quad\quad$ get a_i^u, m_i^u;
10 $\quad\quad\quad s_i^u$ = `StepDetection` (a^u);
11 $\quad\quad\quad m_i^{up}$ = `MagPreprocess` (m_i^u);
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 $\quad$ **if** `DirInc` (p_u, p_v) != *IncColumn* **then**
15 $\quad\quad p_v++$;
16 $\quad\quad p_u^{inc} = false$;
17 $\quad\quad$ **for** $j = p_u - c; j \leq p_u; j++$ **do**
18 $\quad\quad\quad D[p_v][j] = \min(D[p_v - 1][j - 1], D[p_v - 1][j], D[p_v][j - 1]) + (m_{p_u}^{up} - m_{p_v}^{vp})^2$;
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 $\quad$ **if** `DirInc` (p_u, p_v) != *IncRow* **then**
22 $\quad\quad p_u++$;
23 $\quad\quad p_u^{inc} = true$;
24 $\quad\quad$ **for** $j = p_u - c; j \leq p_u; j++$ **do**
25 $\quad\quad\quad D[j][p_u] = \min(D[j - 1][p_u - 1], D[j - 1][p_u], D[j][p_u - 1]) + (m_{p_u}^{up} - m_{p_v}^{vp})^2$;
26 $\quad\quad$ **end**
27 $\quad$ **end**
28 **end**

Algorithm 1: Magnetic-field-based trace synchronization

To determine the forward direction, function $DirInc$ in Algorithm 2 first checks the step conditions between line 8 and 13. Specifically, given the current pointer location (p_v, p_u), we first extract the last t_w-second walk and the corresponding number of steps taken by the leader. Based on the synchronization results between traces u and v, we than identify the time taken by the follower to complete this distance. Finally, we compare the numbers of steps taken by the leader and the follower (line 8–13). If $vSteps$ is much less than $uSteps$, the row number increases (line 9); if $uSteps$ is much less than $vSteps$, the column number increases (line 12). If the step condition is satisfied, the algorithm sets the forward direction based on the position of the minimal similarity scores (lines 14–22). If the local minimal cost is achieved at $D[p_v][p_u]$, both the row and the column numbers increase. If it is on the p_v-th row, the row number increases, else the column number increases. In the current implementation of FollowMe, we empirically set $t_w = 2$ and $StepBound = 3$.

Figure 7 shows a snapshot of the warping cost matrix during the execution of Algorithm 1. The valley in the center of this figure (the arrows) is the optimal warping path. We can see that the algorithm dynamically changes the search band and calculates entries only within a certain range of the band (i.e., between the two red lines). To quantify the computational overhead, we ran the algorithm offline on a PC and Galaxy S5 using traces with different lengths and record the running time in Table 2. The linear

1 **Function** *DirInc*(p_u, p_v)
2 $\quad$ **if** $i \leq cIni$ **then**
3 $\quad\quad$ **return** *IncBoth*;
4 $\quad$ **end**
5 $\quad$ given $uv_i = k$, calculate
6 $\quad vSteps = s_k^v - s_{k'}^v$ where $t_k^v - t_{k'}^v = t_w$;
7 $\quad uSteps = s_i^u - s_{i'}^u$ where $uv_{i'} = k'$;
8 $\quad$ **if** $vSteps + StepBound \leq uSteps$ **then**
9 $\quad\quad$ **return** *IncRow*;
10 $\quad$ **end**
11 $\quad$ **if** $vSteps - StepBound \geq uSteps$ **then**
12 $\quad\quad$ **return** *IncColumn*;
13 $\quad$ **end**
14 $\quad (m, n) = \arg\min(D[x][y])$ where $x = p_v, y \in [p_u - c + 1, p_u]$ or $y = p_u, x \in [p_v - c + 1, p_v]$;
15 $\quad$ **if** $m < p_v$ **then**
16 $\quad\quad$ **return** *IncColumn*;
17 $\quad$ **end**
18 $\quad$ **if** $n < p_u$ **then**
19 $\quad\quad$ **return** *IncRow*;
20 $\quad$ **else**
21 $\quad\quad$ **return** *IncBoth*;
22 $\quad$ **end**
23 **end**

Algorithm 2: Trace synchronization function

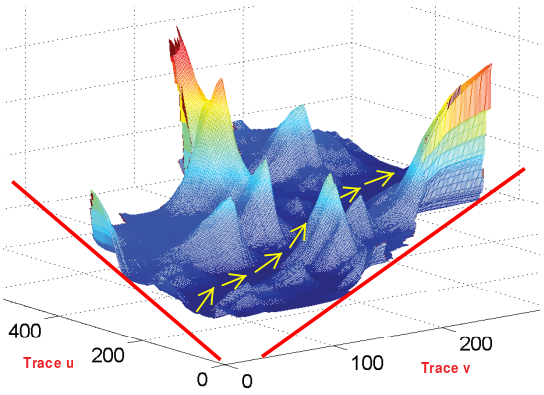


Figure 7: 3D plot of a warping cost matrix.

relation between the trace length and the running time proves the effectiveness of the algorithm.

Table 2: Computation time (s)

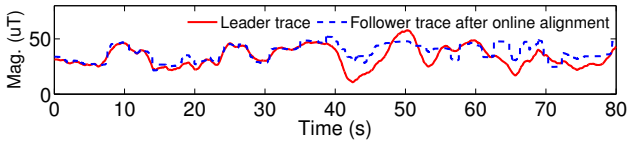
# of samples	1000	2000	3000	4000	5000
PC	1.28	4.17	6.96	9.83	12.51
Smartphone	8.21	12.85	18.12	23.29	27.96

5.3 Deviation Detection

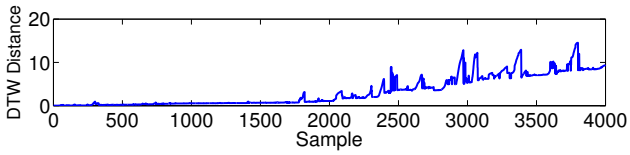
In spite of advance notice of turns and level-changes provided by FollowMe, users might still accidentally miss a turn or go off the user's trace on purpose (e.g., to see something interesting along the way). To handle this problem, we design a deviation detector in FollowMe which performs the following tasks. First, it automatically detects if the user goes off the trail. When

happens, it notifies the user to make a U-turn and navigates him back to the correct path. Specifically, FOLLOWME replays (in the reverse direction) turning or stair climbing actions the user took on the smartphone screen. When to display these instructions is computed by the walking progress estimator, which synchronizes the geomagnetic observations before and after the U-turn. Note that the reverse navigation component can also be used to guide the user back to a previously visited place. Since the reverse navigation works the same as the normal forward navigation, in what follows we will focus on deviation detection.

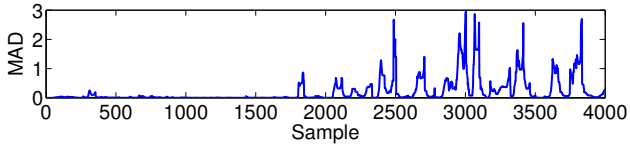
We detect deviations by tracking similarities of geomagnetic observations from the reference and the user traces. The rationale behind this is simple: different paths possess distinct patterns of geomagnetic intensity. To this end, FOLLOWME keeps monitoring the warping cost matrix D during the trip and uses the median absolute deviation (MAD), which is a robust measure of the variability of a univariate sample of quantitative data, to quantify the increase of DTW distance.



(a) Original traces after alignment



(b) DTW distance



(c) MAD of the DTW distance

Figure 8: Deviation detection.

Take Figure 8 as an example. The user deviated from the correct path after walking for 35 seconds. From Figure 8(a), we can clearly see the difference between the two traces even after time warping. Further, in Figure 8(b), the DTW distance is shown to increase significantly after 1800 samples (i.e., 35 seconds with a sampling frequency of 50Hz). Based on DTW distance, MAD is calculated using a window of 50 samples (Figure 8(c)). Deviation is detected if the cumulative MAD within a certain period grows above a certain threshold. In the current implementation of FOLLOWME, we set the length of this period to 5 seconds and the threshold to the cumulative MAD within a prior neighboring time window.

6. IMPLEMENTATION

To illustrate the application of FOLLOWME, we build a reference system on the Android platform and implement both the trace-collection module and the navigation module. In this section, we first briefly describe the implementation of step, turn, and level-change detections, and then introduce the implementation of online navigation in Section 6.4. Note that other motion-detection techniques can also be leveraged in FOLLOWME.

6.1 Step Detection

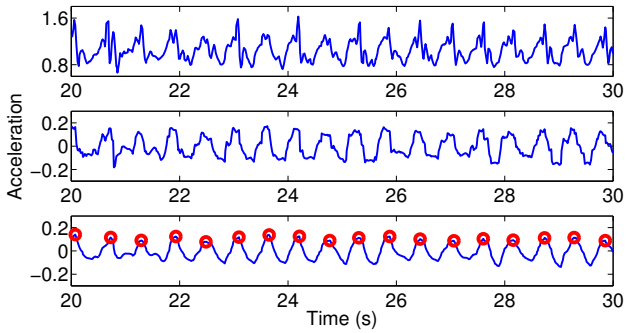
Step information of both the leader and the follower is used in the walking progress estimator (in Section 5.2) to navigate users. FOLLOWME recognizes user steps based on the accelerometer readings. Since the maximum amplitudes along all 3 axes of the accelerometer occur when the heel strikes the ground [28], we devise a peak recognition algorithm to detect these strikes/steps. To make the step detection independent of the phone's orientation, only the magnitude of the 3-axis acceleration reading (i.e., $a = \sqrt{(a_x^2 + a_y^2 + a_z^2)}$) is considered.

The signal magnitudes first pass through a smoothing filter in the preprocessing step. To extract better the low-band step component, a low-pass filter is then applied to filter out high-frequency accelerations caused by the phone's random movement. The low-band component of the acceleration magnitude is calculated online using the following first-order difference equation:

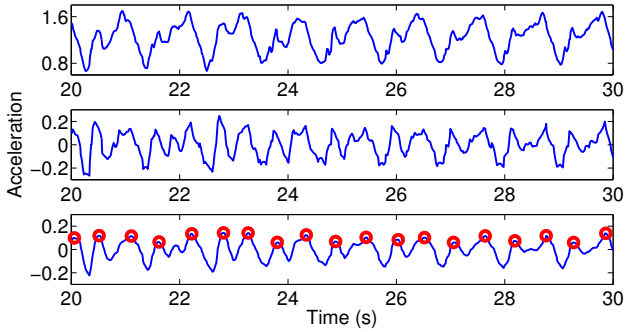
$$a_i^l = a_{i-1}^l + \alpha \times (a_i - a_{i-1}^l) \quad (1)$$

where a_i is the i-th original acceleration magnitude and a_i^l is the value after passing through the low-pass filter. The default value of α is set to 0.25.

After the low-pass filter, a peak recognition algorithm with a sliding window is used to detect peaks in the filtered data. Specifically, a_i^l is recognized as a peak (i.e., a user step) if it is larger than all samples located in the range of $[t(i) - t_w/2, t(i) + t_w/2]$. Since the user step frequency is in general lower than 3Hz, the window size t_w in the current implementation is set to $0.3s$.



(a) Hold smartphone horizontally.



(b) Hold smartphone in hands.

Figure 9: Data processing for step detection.

Figure 9 shows processed acceleration signals. In both figures, the first row displays the original acceleration output from the smartphone, and the smoothed acceleration data and the corresponding low-band component are shown in the second and third rows, respectively. Recognized peaks are highlighted in the third row in red dots. In Figure 9 our step detection algorithm is shown to perform well even when the user swings his arm with the phone in hand.

6.2 Turn Detection

In outdoor environment, turn information can be extracted from the electronic compass. However, due to ferromagnetic interference, reliable direction output cannot be obtained from the compass in indoor environments. Thus, in indoor environments, we detect turns by jointly considering signals from the accelerometer and the gyroscope to exploit their relative strengths: gyroscope readings can be integrated to produce angle estimates that are reliable over a short term, whereas the accelerometer suffering from random vibrations can be trusted over a long term to provide attitude estimation.

Turn detection is achieved based on the fact that the rotation axis of the body during a turn is always directed toward the center of the Earth (i.e., in the direction of gravity). In other words, users as well as smartphones always rotate around the Z-axis of the local vertical, local horizontal (LVLH) frame. Since the gyroscope measures the angular velocities of rotation on each axis of the smartphone's body frame, we first determine the attitude of the smartphone using the value of gravity on 3 axes of the accelerometer on the phone's body frame, and then transform the angular velocity from the body frame to the LVLH frame to determine turns.

In FOLLOWME, we adopt a rotation matrix to describe the orientation of the smartphone from the LVLH frame to its body frame.[3] Detailed derivations of frame transformation are omitted due to space limitation. Note that due to the lack of reliable compass reading, the phone's attitude is confined to a conical surface in the LVLH frame and cannot be uniquely identified. However, without the need for the yaw angle ψ, we are still able to detect turns because the rotation always happens about the Z-axis in the LVLH frame. In addition, to avoid the Gimbal Lock problem [30], we need to be cautious about the rotation sequence of three axes. As turn detection is independent of Z-axis, a simple solution is to exchange the X-axis and Y-axis (i.e., change from the right-hand coordinate system to the left-handed coordinate system) if the gravity acceleration factor on X-axis is larger than that on Y-axis. Once the smartphone attitude is obtained, we transform the angular velocity and compute the amount of rotation via integration.

6.3 Level-Change Detection

Due to its low power and excellent relative accuracy [20, 31], a barometer is adopted in FOLLOWME for level-change detection. Figure 10 shows a barometer trace collected during a user's walk from the third floor to the second and then climbing up to the fourth floor. We also record the timestamps when the user starts walking on the stairs and arrives at a new level as the level-change ground truth. In this figure, we can clearly see the increase/decrease of the atmospheric pressure when the altitude changes.

In FOLLOWME, we devise a two-pass bi-directional searching algorithm to detect level-changes. In this algorithm, we first smooth each atmospheric pressure data p_n by averaging all samples within the previous 4 seconds (as the dark curve in Figure 10). The algorithm then traces back to find the maximal difference between p_n and samples collected within the last T_0 seconds. If a gap $|p_n - p_m|$, indicating a level change, is greater than the threshold p_{tr}, we conduct a forward search afterwards to determine the altitude-changing period.[4] The algorithm outputs a value r_l that indicates whether the altitude increases/decrease ($r_l = -1/1$) or not ($r_l = 0$). Particularly, if less than 5 steps are taken by the user during the level-change period, we record an elevator up/down. A sample detection result of level changes is shown above the curves in Figure 10. In this figure, the dark (blue) area refers to the duration when the user is walking downstairs, and the light (yellow) area indicates walking on an upward staircase. From this figure, we find the detected stair walking periods fit the ground truth well. More on detection performance will be provided in Section 7.

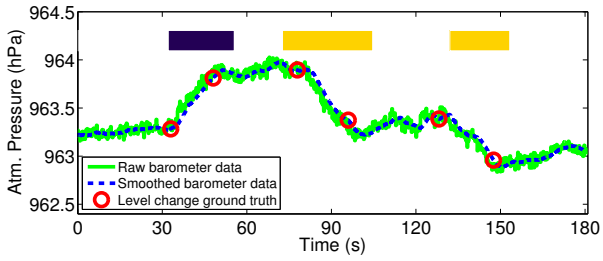


Figure 10: Data processing for level change detection.

6.4 Navigation Module

We implement FOLLOWME on a Samsung Galaxy S5 running Android (version 4.4.2). In both the reference trace-collection mode and the navigation mode, FOLLOWME runs two threads: one for sensory data collection (using callback function $onSensorChanged()$) and the other to take care of the signal processing asynchronously. We down-sample the sensor data at 50Hz to reduce the computation overhead of the smartphone. The reference trace is saved to the internal storage shared between these two modes. For the walking progress estimator, we set the path constraint coefficient $c = 600$ and $cIni = 200$.

Figure 11 shows the FOLLOWME GUI where navigation instructions are provided: after loading the reference trace and pressing the start button, turning instructions are updated on the right of the screen (the turn icon) during user walk. The follower is allowed to walk at different speeds and even stop anytime during the walk. If the follower is required to walk up/down stairs, a staircase icon shows up on the left of the screen.

7. EVALUATION

We have conducted experiments in a 4-story campus building with the testing area of $2000m^2$. A snapshot of the experimental environment can be seen in Figure 12. Figure 18 also shows preliminary evaluation results of FOLLOWME in semi-outdoor and outdoor open spaces.

Five users participated in our evaluation of the performance of detection of steps, turns, level-changes, and navigation, including trace-synchronization accuracy, timeliness of navigation instructions and deviation-detection performance. To evaluate the performance of navigation, 10 different reference traces were generated by randomly selected users who held their phones

[3]Other techniques such as quaternion rotations can also be leveraged to determine the phone's attitude [29].

[4]The resolution of vertical distance changes depends on the threshold value p_{tr}. In our current implementation, we empirically set $p_{tr} = 0.3$.

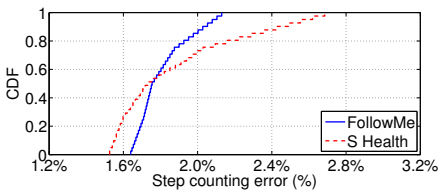


Figure 13: Performance of step detection.

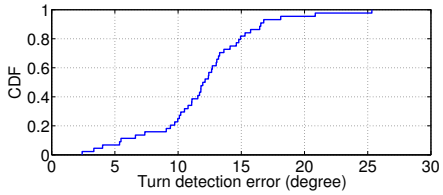


Figure 14: Performance of turn detection.

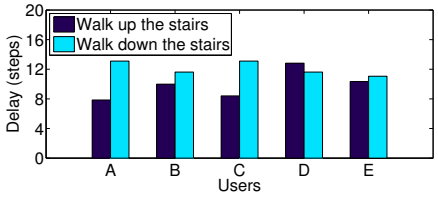


Figure 15: Delay of level-change detection.

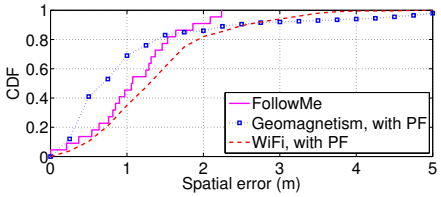


Figure 16: CDF of spatial error in navigation.

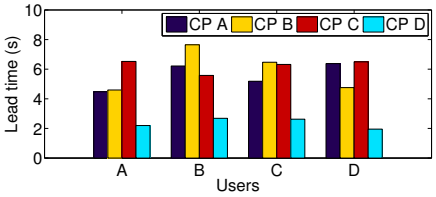


Figure 17: Lead time of navigation instructions at different checkpoints.

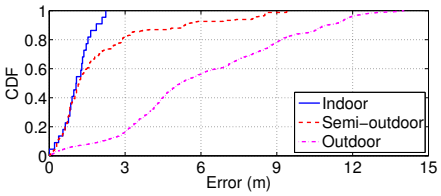


Figure 18: Tracking error of the walking progress estimator.

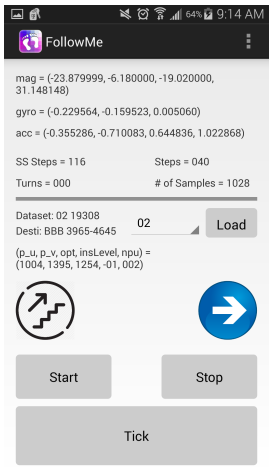


Figure 11: A snapshot of FOLLOWME.

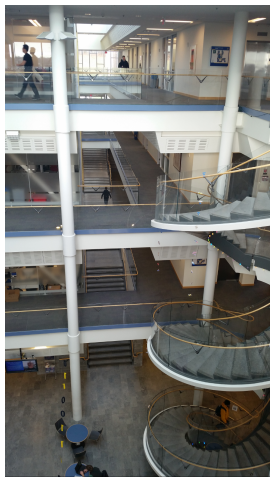


Figure 12: A snapshot of the experimental environment.

in front of their bodies. However, we did not impose any constraints on the phone's orientation (e.g., holding the phone horizontally or vertically) during trace collection and navigation. Although FOLLOWME does not perform level-change detection in the navigation mode, we recorded data from the barometer, accelerometer, and gyroscope, and ran detection algorithms offline for the purpose of evaluation.

7.1 Accuracy of Detection of Steps, Turns, and Level-Changes

We first evaluate the performance of step, turn, and level-change detection.

The step-counting errors are shown in Figure 13. We also compare the error of the pedometer in S Health, a built-in fitness application in Galaxy S5. The figure shows that FOLLOWME achieves comparable step-counting to S Health. The error of FOLLOWME is below 2.2% which indicates the step-counting error per 100 steps is less than 3 steps. In the navigation module of FOLLOWME, such a small error can be compensated for by matching the geomagnetic field features.

The turn-detection results are plotted in Figure 14. Specifically, we obtained the ground-truth angles of turns on each test path using the floor plan, and compared them with the turns detected by the trace collection module. From Figure 14, we find that the turn-detection component achieves a 90 percentile accuracy of 15°. This error is caused by two factors. First, the actual angle of the user's turn may not be identical to that of the path. Second, the accelerometer and gyroscope are susceptible to perturbation which can also lower accuracies.

Our experiments show 100% accuracy in detecting level-changes. To examine another detection performance metric, Figure 15 plots the delay in detecting level-changes. Due to subtle pressure changes, detection of level-changes took more time than both step and turn detections. For example, the average delay of detecting walking downstairs is 12 seconds. Since the level-change detection works in the trace-collection module, the relative large delay has little influence on the navigation performance.

7.2 Navigation Performance

Let us consider the real-time navigation performance. We first recorded the ground truth locations of the leader at different times in the trace-collection process. During the follower's walk, FOLLOWME compared the instantaneous magnetic field measurements and the data collected by the leader to estimate the follower's relative position. This way, we can measure the offset between the follower's true locations and the locations obtained by the matching algorithm. For the purpose of comparison, we also implemented tracking algorithms using the geomagnetic field or WiFi as benchmarks [21]. Both approaches take a fingerprint map and the floor plan as inputs, and use particle filtering (PF) to continuously estimate the device's location. Step and turn detection algorithms in FOLLOWME are also leveraged for particle movement and weight updating.

Figure 16 plots the spatial errors of navigation systems. This figure shows that the spatial errors are less than 2.5m in FOLLOWME. Compared to benchmarks, we find that FOLLOWME outperforms the WiFi-based tracking and achieves a comparable 80 percentile tracking accuracy with the geomagnetic-based algorithm. In addition, the maximal spatial error of FOLLOWME is smaller

than both of the other two approaches. This is because the one-dimensional trace synchronization intrinsically limits the search space, thus reducing the estimation error.

Figure 17 examines further the timeliness of navigation instructions and plots the lead time of the instructions at different checkpoints (i.e., crossings and staircases). We find that most navigation instructions are provided 4 seconds ahead of the action. However, for the checkpoint (CP) D, the average lead time of the notification is around 2 seconds, since it is close to a previous turn.

We also conducted experiments in semi-outdoor and outdoor open spaces to evaluate the navigation performance of FOLLOWME. Figure 18 plots the CDF of spatial error in all three environments. FOLLOWME is shown to perform best in indoor environments, where 95% of the absolute spatial errors are less than 2m. However, in outdoor open spaces, only 50% of the errors are within the range of 5m and the error can be as large as 14m. This is due mainly to the unobservable geomagnetic anomalies in outdoor open space.

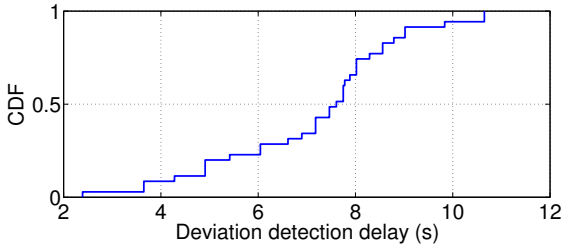
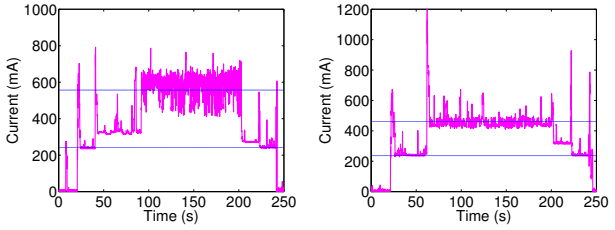


Figure 19: Delay of deviation detection.

We now consider the performance of deviation detection. Specifically, Figure 19 shows the CDF of detection delay. Figure 19 shows that all deviations were successfully detected in 11 seconds. In fact, in 60% of cases, it took 7–9 seconds for FOLLOWME to discover a divergence.

7.3 Energy Consumption

We now evaluate the energy consumption of FOLLOWME. To measure power consumption, we use a Monsoon Power Monitor as a power supply for the smartphone and tracks both current and voltage. During the experiment, we turned off all background applications as well as extra hardware components, such as WiFi, GPS, etc.



(a) Current measurement of navigation.

(b) Current measurement of trace collection.

Figure 20: Power consumption of FOLLOWME.

Runtime current measurements of FOLLOWME are plotted in Figure 20. In Figure 20(a), the smartphone is in sleep mode during the period from 0 to 20s. FOLLOWME began to run at 40s and loaded the reference trace at around 65s. The navigation module started at 90s and continued running until 200s. FOLLOWME was turned off at 220s. In the trace-collection phase, the module started running at 60s and stopped at 200s, as shown in Figure 20(b).

We compared the energy consumption of FOLLOWME and Travi-Navi, a vision-based leader–follower indoor navigation system [19]. As different smartphones (Samsung Galaxy S4 vs. Galaxy S2) are used in two tests, we exclude the energy cost of Android core services, and focused only on incremental energy consumption (i.e., the range between blue lines) when both navigation systems began to run.

Figure 20 shows that the runtime currents of FOLLOWME are 303.4mA and 224.6mA in navigation and trace-collection phases, respectively. According to the data provided in [19], FOLLOWME achieved energy savings of nearly 50% (224.6mA vs. 433.4mA) in the trace-collection phase and of 15% (303.4mA vs. 349.5mA) in the navigation phase. This is because FOLLOWME did not use energy-hungry sensors, such as WiFi and the camera, but instead the lightweight step-constrained trace synchronization algorithm. FOLLOWME can also offload the data preprocessing to co-processors to further reduce energy consumption, which is part of our future work.

8. DISCUSSION AND FUTURE WORK

Although FOLLOWME provides good navigation performance, there is room for further enhancements. Discussed below are several practical issues and limitations of the experiments that warrant further investigation.

8.1 Limitations of the Experiments

Despite the various encouraging results reported in Section 7, the evaluation of FOLLOWME is still preliminary. First, we have not thoroughly assessed the impact of temporal and device diversities on navigation accuracy. Although the step-constrained trace synchronization algorithm plays an active role in walking progress estimation with inconsistent walking speeds, it has not been evaluated systematically for a large group of users (e.g., between the elderly and children) and different combinations of parameters (e.g., t_w and $StepBound$ in Algorithm 2).

In addition, we have not fully tested FOLLOWME in less defined indoor spaces where navigation becomes more challenging. For example, simple instructions, such as left/right turns, may not be enough for successful navigation in a large open indoor space. Users may also have to make a detour if obstacles temporarily block a path. The first problem can be solved as the turn-detection algorithm recognizes turns of different angles, and displays the corresponding arrows on the screen. It imposes the additional constraints on the user (e.g., the smartphone should be held horizontally in hand). For the latter case, we can make the trace synchronization and deviation detection algorithms tolerate transient deviations. Once the user walks back to the correct path (e.g., within 10 seconds), the navigation module automatically discards the detour part, and continues to synchronize the observed sensory data with the reference trace. The design of a detailed algorithm for this is part of our future work.

8.2 Open Issues Related to Reference Traces

We summarize some open problems related to the reference trace, which plays a key role in FOLLOWME.

8.2.1 Scalability

As mentioned in Section 3.3, FOLLOWME is designed as a plug-and-play system which meets the need for individuals' navigation.

However, we need to design a more efficient way to store reference traces when they are large in number and size. For example, we can divide traces and store only one copy of common segments of different traces in the database. This way, FOLLOWME is able to create new traces from scratch by concatenating segments. Besides, how to combine traces collected at different locations to help create an indoor fingerprint map is also an interesting subject to pursue.

8.2.2 *Quality Control*

In current design of FOLLOWME, each reference trace is built based on a single user's walking trace. However, no mechanism has been provided to guarantee the quality of the reference traces. A reference trace which is averaged over multiple runs will help improve the navigation performance, although merging traces with time-varying walking speeds is non-trivial.

8.2.3 *Information Privacy*

If users are allowed to share reference traces offline in an uncontrolled manner, FOLLOWME may pose a risk of privacy leakage to the users and third-party organizations. For example, some companies might not want anyone to share any information about their floor plans or indoor sensory data. Therefore, how to build a safeguard to preserve information privacy is an interesting issue to explore.

8.3 Mixed Modality with Additional Location-Specific Features

We exploited the geomagnetic field to synchronize the reference trace and the follower's trace. While enjoying the pervasiveness and stability of geomagnetic field, FOLLOWME faces several ensuing problems. For example, the weak geomagnetic disturbances impair the usability of FOLLOWME in outdoor open spaces. In future, we would like to incorporate more location-specific signals and build a mixed modality for walking progress estimation. For example, we can jointly consider the geomagnetic intensity and GSM/FM signal strength and synchronize multiple traces. In indoor environments, opportunistically-sensed WiFi can also be fused with geomagnetic signals. Even for a given specific type of location feature, we can synchronize multiple traces that have the same destination, thus automatically determining the user's starting point according to their similarities.

9. RELATED WORK

Navigation and localization have been extensively studied in the area of robotics. By fusing odometer outputs with IMU sensing results, robots can compute travel distances, perform accurate localization, and navigate themselves to the desired destination based on the map information. In several robotic systems, additional sensing techniques using laser [32], infrared [33], and camera [34] are also used for ranging and navigation purposes. However, humans' locomotion is much more complicated. The limited sensing capabilities and energy buffer of smartphones also add difficulties to both localization and navigation.

Numerous localization techniques using smartphones have been proposed in the area of mobile computing [1, 3, 5–7, 9, 35–39]. They can be broadly categorized as (i) infrastructure-based localization such as using GPS, cellular and WiFi signals, or (ii) infrastructure-independent localization such as Dead Reckoning (i.e., IMU-tracking). Each of these two types of localization has its own advantages and disadvantages. GPS can provide accurate positioning in outdoor open spaces but encounters fading signals in indoor environments, whereas Dead Reckoning suffers from cumulative errors and is affected by the smartphone's usage. Despite extensive explorations of integrating these approaches, the lack of map information makes the last-mile navigation a difficult problem. In FOLLOWME, we exploit ideas from these techniques (e.g., using IMU to detect steps, turns) as well as several recently-proposed smartphone services [28, 29] to build a lightweight, plug-and-play, last-mile navigation system.

Geomagnetic field anomalies are used for localization and navigation of both robots and smartphones. In [40], the authors leverage observations of the ambient magnetic field for indoor localization, but they can handle only simple straight pathways. Glanzer *et al.* [41] introduced a pedestrian navigation system with human motion recognition, although a pre-established magnetic field map is equipped to correct the severe disturbance of indoor direction sensing. Magnetic signatures were leveraged in [23] to identify locations and rooms. Although mobile phones are used to measure magnetic field intensity, the system relies on pillars and offers only room-level positioning accuracy. Grand *et al.* [22] proposed a lightweight magnetic map construction method and used an online particle filter to estimate the location of a handheld device. Similarly, a particle-filtering-based engine was designed in [21] to localize and track users with a given geomagnetic fingerprint map. In general, these techniques require special hardware [13, 24–27] or dense samples of magnetic data to build a fingerprint map with high training overhead [13, 21, 23, 42]. In contrast, FOLLOWME utilizes the features of geomagnetic field without customized hardware and avoids the time-consuming map construction process.

The leader–follower model has been adopted in several existing navigation systems [17–19, 43]. In [18] and [43], researchers used customized devices of magnetic sensing to navigate blind people and autonomous vehicles. In [17], an electronic escort system was proposed by using crowd encounters information and dead-reckoning techniques, but it requires pre-deployed audio beacons, which impairs its usability. Travi-Navi [19], a vision-guided navigation system, enables a user to easily bootstrap and deploy indoor navigation services without building the entire localization system. FOLLOWME is very different from Travi-Navi in several aspects. First, Travi-Navi uses compute-intensive particle filtering as the navigation engine. Unlike multiple indoor tracking systems proposed to use particle filtering, floor plan (e.g., information of walls) is no longer available to the system and hence cannot help expedite the convergence of particles. The free movement of particles in a two-dimensional space intrinsically introduces difficulties in navigation. To alleviate this problem, Travi-Navi uses WiFi AP information and has to continuously estimate the walking stride length. However, the former is known to be time-varying whereas the latter can be only roughly calculated even with a known user's height. On the contrary, FOLLOWME is a lightweight system that performs matching in a one-dimensional space based on accurate step detection and ubiquitous geomagnetic field. Second, FOLLOWME minimizes the constraints imposed on users by providing wider usage (i.e., in multi-level buildings, semi-outdoors). On the other hand, guiders in Travi-Navi need to hold the smartphone vertically and steadily during walking to achieve a better image quality, and the followers' intelligence is implicitly exploited for image recognition.

10. CONCLUSION

In this paper, we present FOLLOWME, a lightweight, plug-and-play, last-mile navigation system. It guides users by providing the "scent" left behind by leaders or previous travelers. In particular, it introduces a novel online magnetic-filed-based trace synchronization scheme for estimating the user's walking progress.

FOLLOWME also recognizes natural walking patterns to help reduce the computational overhead and to navigate subsequent users. We have implemented FOLLOWME on Android smartphones, evaluated it in a four-story campus building. 95% of FOLLOWME's spatial errors were found to be 2m or less during navigation and FOLLOWME saves at least 50% of energy consumption compared to a benchmark system.

Acknowledgment

This work was done during Yuanchao's visit to the University of Michigan, which is supported in part by 973 Program under grant 2015CB352503 and scholarship funded by China Scholarship Council. We thank the anonymous reviewers and the shepherd for their constructive comments and support.

References

[1] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors. In *ACM UbiComp*, 2012.

[2] Moustafa Youssef and Ashok K. Agrawala. The Horus WLAN Location Determination System. In *ACM MobiSys*, 2005.

[3] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *IEEE INFOCOM*, 2000.

[4] Sungro Yoon, Kyunghan Lee, and Injong Rhee. FM-based Indoor Localization via Automatic Fingerprint DB Construction and Matching. In *ACM MobiSys*, 2013.

[5] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. Indoor Localization without the Pain. In *ACM MobiCom*, 2010.

[6] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. Zee: Zero-effort Crowdsourcing for Indoor Localization. In *ACM MobiCom*, 2012.

[7] Jie Xiong and Kyle Jamieson. ArrayTrack: A Fine-Grained Indoor Location System. In *USENIX NSDI*, 2013.

[8] Daniel Turner, Stefan Savage, and Alex C. Snoeren. On the Empirical Performance of Self-calibrating WiFi Location Systems. In *ACM LCN*, pages 76–84, 2011.

[9] Zheng Yang, Chenshu Wu, and Yunhao Liu. Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention. In *ACM MobiCom*, 2012.

[10] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-Markie: Indoor Pathway Mapping Made Easy. In *USENIX NSDI*, 2013.

[11] Liqun Li, Guobin Shen, Chunshui Zhao, Thomas Moscibroda, Jyh-Han Lin, and Feng Zhao. Experiencing and Handling the Diversity in Data Density and Environmental Locality in an Indoor Positioning Service. In *ACM MobiCom*, 2014.

[12] HKUST. Path Advisor. http://pathadvisor.ust.hk/.

[13] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor Location Sensing Using Geo-magnetism. In *ACM MobiSys*, 2011.

[14] Binghao Li, Thomas Gallagher, Andrew G Dempster, and Chris Rizos. How Feasible Is the Use of Magnetic Field Alone for Indoor Positioning? In *IPIN*, 2012.

[15] Xiaofan Jiang, Chieh-Jan Mike Liang, Kaifei Chen, Ben Zhang, Jeff Hsu, Jie Liu, Bin Cao, and Feng Zhao. Design and Evaluation of a Wireless Magnetic-based Proximity Detection Platform for Indoor Applications. In *ACM IPSN*, 2012.

[16] Pengfei Zhou, Yuanqing Zheng, Zhenjiang Li, Mo Li, and Guobin Shen. IODetector: A Generic Service for Indoor Outdoor Detection. In *ACM SenSys*, 2012.

[17] Ionut Constandache, Xuan Bao, Martin Azizyan, and Romit Roy Choudhury. Did You See Bob?: Human Localization Using Mobile Phones. In *ACM MobiCom*, 2010.

[18] Timothy H. Riehle, Shane M. Anderson, Patrick A. Lichter, Nicholas A. Giudice, Suneel I. Sheikh, Robert J. Knuesel, Daniel T. Kollmann, and Daniel S. Hedin. Indoor Magnetic Navigation for the Blind. In *IEEE EMBC*, 2012.

[19] Yuanqing Zheng, Guobin Shen, Liqun Li, Chunshui Zhao, Mo Li, and Feng Zhao. Travi-Navi: Self-deployable Indoor Navigation System. In *ACM MobiCom*, 2014.

[20] Kartik Muralidharan, Azeem Javed Khan, Archan Misra, Rajesh Krishna Balan, and Sharad Agarwal. Barometric Phone Sensors – More Hype than Hope! In *ACM HotMobile*, 2014.

[21] Yuanchao Shu, Cheng Bo, Guobin Shen, Chunshui Zhao, Liqun Li, and Feng Zhao. Magicol: Indoor Localization Using Pervasive Magnetic Field and Opportunistic WiFi Sensing. *IEEE Journal on Selected Areas in Communications*, 33(7):1443–1457, July 2015.

[22] E. Le Grand and S. Thrun. 3-axis Magnetic Field Mapping and Fusion for Indoor Localization. In *IEEE Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012.

[23] B. Gozick, K.P. Subbu, R. Dantu, and T. Maeshiro. Magnetic Maps for Indoor Navigation. *IEEE Transactions on Instrumentation and Measurement*, 60(12):3883–3891, 2011.

[24] Sam Ann Rahok, Yoshihito Shikanai, and Koichi Ozaki. Trajectory Tracking Using Environmental Magnetic Field for Outdoor Autonomous Mobile Robots. In *IEEE/RSJ IROS*, 2010.

[25] S. Suksakulchai, S. Thongchai, D. M. Wilkes, and K. Kawamura. Mobile Robot Localization Using an Electronic Compass for Corridor Environment. In *IEEE ICSMC*, 2000.

[26] I. Vallivaara, J. Haverinen, A. Kemppainen, and J Roning. Simultaneous Localization and Mapping Using Ambient Magnetic Field. In *IEEE Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2010.

[27] S.A. Rahok and O. Koichi. Odometry Correction with Localization Based on Landmarkless Magnetic Map for Navigation System of Indoor Mobile Robot. In *International Conference on Autonomous Robots and Agents (ICARA)*, 2009.

[28] Nirupam Roy, He Wang, and Romit Roy Choudhury. I am a Smartphone and I Can Tell My User's Walking Direction. In *ACM MobiSys*, 2014.

[29] Pengfei Zhou, Mo Li, and Guobin Shen. Use It Free: Instantly Knowing Your Phone Attitude. In *ACM MobiCom*, 2014.

[30] Wikipedia. Gimbal Lock. http://en.wikipedia.org/wiki/Gimbal_lock.

[31] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L. Ananda, Mun Choon Chan, and Li-Shiuan Peh. Using Mobile Phone Barometer for Low-power Transportation Context Detection. In *ACM SenSys*, 2014.

[32] Shung Han Cho and Sangjin Hong. Map Based Indoor Robot Navigation and Localization Using Laser Range Finder. In *ICARCV*, 2010.

[33] A. M. Flynn. Combining Sonar and Infrared Sensors for Mobile Robot Navigation. *Int. J. Rob. Res.*, 7(6):54–64, December 1988.

[34] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual Navigation for Mobile Robots: A Survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

[35] Yin Chen, Dimitrios Lymberopoulos, Jie Liu, and Bodhi Priyantha. FM-based Indoor Localization. In *ACM MobiSys*, 2012.

[36] Jeongyeup Paek, Kyu-Han Kim, Jatinder Pal Singh, and Ramesh Govindan. Energy-efficient Positioning for Smartphones Using Cell-ID Sequence Matching. In *ACM MobiSys*, 2011.

[37] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards Mobile Phone Localization without War-Driving. In *IEEE INFOCOM*, 2010.

[38] Qiang Xu, Alexandre Gerber, Zhuoqing Morley Mao, and Jeffrey Pang. Acculoc: Practical Localization of Performance Measurements in 3G Networks. In *ACM MobiSys*, 2011.

[39] Yuanchao Shu, P. Coue, Yinghua Huang, Jiaqi Zhang, Peng Cheng, and Jiming Chen. Demo: G-Loc: Indoor Localization Leveraging Gradient-based Fingerprint Map. In *IEEE INFOCOM*, 2014.

[40] Janne Haverinen and Anssi Kemppainen. A Global Self-localization Technique Utilizing Local Anomalies of the Ambient Magnetic Field. In *IEEE ICRA*, 2009.

[41] G. Glanzer and U. Walder. Self-contained Indoor Pedestrian Navigation by Means of Human Motion Analysis and Magnetic Field Mapping. In *Workshop on Positioning Navigation and Communication (WPNC)*, 2010.

[42] M. Angermann, M. Frassl, M. Doniec, B.J. Julian, and P. Robertson. Characterization of the Indoor Magnetic Field for Applications in Localization and Mapping. In *IEEE IPIN*, 2012.

[43] W. Storms, J. Shockley, and J. Raquet. Magnetic Field Navigation in an Indoor Environment. In *IEEE UPINLBS*, 2010.