Travi-Navi: Self-deployable Indoor Navigation System

Yuanqing Zheng[†,§], Guobin Shen[†], Liqun Li[†], Chunshui Zhao[†], Mo Li[§], Feng Zhao[†]
[†]Microsoft Research, Beijing, China
[§]Nanyang Technological University, Singapore
{yuanqing1, limo}@ntu.edu.sg, {jackysh, liqul, chunzhao, zhao}@microsoft.com

Abstract

We present Travi-Navi – a vision-guided navigation system that enables a self-motivated user to easily bootstrap and deploy indoor navigation services, without comprehensive indoor localization systems or even the availability of floor maps. Travi-Navi records high quality images during the course of a guider's walk on the navigation paths, collects a rich set of sensor readings, and packs them into a navigation trace. The followers track the navigation trace, get prompt visual instructions and image tips, and receive alerts when they deviate from the correct paths. Travi-Navi also finds the most efficient shortcuts whenever possible. We encounter and solve several challenges, including robust tracking, shortcut identification, and high quality image capture while walking. We implement Travi-Navi and conduct extensive experiments. The evaluation results show that Travi-Navi can track and navigate users with timely instructions, typically within a 4-step offset, and detect deviation events within 9 steps.

Categories and Subject Descriptors

H.5.2 [**Information interfaces and presentation**]: User interface – *User-centered design*; C.3.3 [**Special-Purpose and Application-based Systems**]: Real-time and embedded systems

General Terms

Design, Experimentation, Performance

Keywords

Indoor navigation, Image direction, Self-deployable system

1. INTRODUCTION

There are many situations in real life, including business, social, and personal scenarios, where people have a strong need for navigation services. For instance, a shop owner may want to direct customers to his shop. At a social gathering, early arriving guests may want to guide latecomers. People have for a long time resorted to non-technical solutions in these situations. For example, shop owners post flyers with their addresses or briefly describe routes to attract customers, and latecomers can call friends for landmarks. Such approaches, however, have various shortcomings. For instance, customers need to find the addresses and follow a sequence of landmarks, which may not be a trivial task. If the building floorplans are unavailable or hard to read, the trip can be frustrating. Locating floorplans can also be troublesome in gigantic malls. Moreover, customers may take wrong turns when no one is around to guide them. When customers want to visit multiple shops, it is not easy to plan trips and minimize detours.

On the other hand, despite extensive research into indoor localization, the wide deployment of indoor localization and navigation systems have yet to be realized. Prior schemes attempted to build full blown localization systems before publishing location services. For instance, WiFi fingerprinting-based localization systems need to sample signal strengths and construct radio maps in advance [7, 36]. Such approaches require intensive initial efforts to bootstrap localization services. With the lack of pre-deployed comprehensive indoor localization and navigation services, we ask the following question: *Can we enable users to easily bootstrap their own indoor navigation services by themselves without dependency on a pre-deployed localization system or even the availability of floor maps?*

In this paper, we provide an affirmative answer through the systematic design and implementation of Travi-Navi – a vision-guided navigation system that enables a user to easily deploy his own indoor navigation services. Inspired by our own navigation experiences, in Travi-Navi, a guider records landmarks (pathway images, sensor readings, radio signals, etc.) along a path and shares them with followers. Using the guider's directions, our application runs on the follower's mobile device and automatically guides the follower by presenting pathway images and indicating turns, etc. Travi-Navi implicitly leverages the follower's visual recognition capability by providing them pathway images to correct slight direction ambiguity and enhance navigation experiences.

The Travi-Navi design naturally avoids the dependency on any pre-deployed localization services or even the floor maps but faces particular challenges. (1) Pathway images contain rich visual information for followers but rapidly drain the battery power. Images taken during a walk can get blurred due to camera shake. How to control energy usage while at the same time ensuring the image quality remains an important issue to address. (2) As the guider's trace only covers the sampled path while most areas are uncharted, it is very hard to accurately track followers on the guider's trace and generate synchronized directions. Providing correct directions at right time is important as both incorrect and untimely directions may lead to wrong turns. We expect our system to alert users in a timely manner when they veer off the correct path. (3) As users

MobiCom'14, September 07 - 11 2014, Maui, HI, USA.

http://dx.doi.org/10.1145/2639108.2639124.

may visit multiple destinations, a friendly navigation system is expected to intelligently plan routes. The navigation traces are independently provided by different guiders without coordination, so it is challenging to find efficient shortcuts among the traces. Ideally, our system should be able to guide users along the optimal path among the available traces to the destinations.

Based on prior research efforts, we have devised several sensor fusion techniques to address the above challenges. (1) To ensure high quality navigation images, we predict image quality based on motion hints from step detection, heading direction and rotation measurement. In particular, we infer stable shooting time using accelerometer readings and trigger image capture to ensure image quality and avoid blurred images. (2) To accurately track a follower and generate synchronized directions, we incorporate both magnetic field distortions and WiFi fingerprint sequences to complement IMU (Inertial Measurement Unit) sensor based dead reckoning and thereby accurately project the follower onto the guider's trace. (3) To find shortcuts and plan routes, we detect overlapping segments and crossing points of multiple traces shared by possibly different guiders. In particular, we detect overlapping segments by measuring their similarity of magnetic field signals and WiFi fingerprint sequences. We identify crossing points by exploiting the mutual trends of average gross distances between sequences of WiFi fingerprints: as a user gets closer to a crossing point, we observe a decreasing trend in average gross distance and an increasing trend when the user moves away. Thus, provided multiple navigation traces to multiple destinations, our application can automatically find shortcuts, weave the traces to form a holistic trace, and guide users to destinations with minimum detours.

We consolidate the above techniques and implement Travi-Navi on the Android platform. The sensor data collection, tracking, and navigation instruction generation are performed in real time. We conducted extensive experiments under various conditions (e.g., day and night, work day and weekend) and navigated users using Travi-Navi in both an office building and a large shopping mall. In experiments, Travi-Navi showed promising results with accurate and timely instructions (within a 4m offset) and prompt deviation alerts (within 9m).

The contributions of this work are not limited to the application centric design of Travi-Navi that enables users to easily deploy their own navigation services without pre-deployed localization systems. Rather, we see more potential as an effective crowdsourcing solution to gradually build up general purpose indoor localization systems. Realizing the challenge of constructing massive location database to bootstrap the localization services, many recent schemes have proposed crowdsourcing the data collection [29, 32, 35]. However, it is hard to find a sustainable incentive mechanism and to ensure data quality in these crowdsourcing schemes. Travi-Navi naturally alleviates such challenges, since users of Travi-Navi are self-motivated and traces collected by guiders can be verified by the followers. With enough penetration of such systems, we can aggregate navigation traces and gradually build up comprehensive indoor localization and navigation systems.

2. OVERVIEW

2.1 A usage example

Figure 1 illustrates a usage scenario of Travi-Navi, where a restaurant owner provides a navigation trace to customers.

When the restaurant owner (as a guider) arrives at an entrance of the mall (e.g., entrance C in Figure 1), he turns on Travi-Navi and collects navigation traces. As he holds the mobile phone in an upright position and walks to the restaurant, Travi-Navi captures

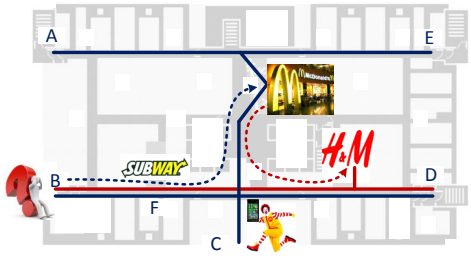


Figure 1: A usage scenario of Travi-Navi. McD collects traces along aisles in blue, and those by H&M are in red. A shortcut (with a red dashed line) between McD and H&M is identified from two guiders' traces.

pathway images and samples WiFi fingerprints and IMU sensors. Travi-Navi automatically processes all sensor data and packs them into a navigation trace. If there are many entrances (e.g., entrances A, B, D, and E), the guider may want to survey multiple paths from each entrance. As it does not need to cover every corner inside the mall, the trace collection load is still light. We note that the primary scenarios of Travi-Navi are planar applications (i.e., navigation on single floors). For shopping malls with multiple floors, the guider may treat lift lobbies and staircases as entrances. We plan to navigate users across multiple floors in our future work.

A customer (as a follower) downloads the traces and navigates to the restaurant. Travi-Navi locks on to the customer at the entrance (i.e., entrance B in Figure 1). As the customer naturally holds his mobile phone in his hand and moves forward, Travi-Navi tracks his progress on the guider's trace by comparing instant sensor readings (IMU sensor and WiFi signals) against the navigation trace. Travi-Navi shows the guider's pathway images and the directions (see UI in Section 4.1). The customer may download multiple traces for different destinations (e.g., the restaurant and the nearby shop in Figure 1). In such a case, Travi-Navi is able to identify a shortcut depicted as a red dashed line.

2.2 Design challenges

Our design avoids dependency on pre-deployed localization services or floorplans. Unlike general-purpose localization schemes, which try to locate users on to the floorplan, our design needs to only track the follower with respect to the guider's pathway and give timely directions. As a result, a guider (e.g., the restaurant owner) can easily survey the possible routes connecting followers to the destination. We see that the route contains necessary information for navigation purposes, and the guider can skip the other areas on the floor.

Such a substantially relaxed requirement for trace collection enables a guider to easily deploy his own navigation service. Yet, it makes locating users on the floorplan very challenging and renders most convention localization schemes infeasible. For instance, WiFi fingerprint based localization schemes cannot accurately locate users, as the WiFi fingerprints only cover a small portion of the whole floor. Thus, given a WiFi fingerprint, localization schemes cannot determine whether it is on the pathway or uncharted areas. Conventional trilateration localization methods cannot be applied to locate users either, since the location of WiFi access points cannot be determined solely based on a route trace. Even with an exhaustive site survey of the whole floor, current localization schemes cannot ensure a high localization accuracy to differentiate pathways (e.g., two close corridors with a distance <5m). Deploying dedicated infrastructure [1, 34] can indeed improve localization accuracy but it incurs a substantial deployment cost. As a result, many

basic requirements in navigation become challenging to fulfill with route traces.

Our design explores the possibility of leveraging visual recognition compatibility of the followers to assist navigation. This allows the active involvement of followers rather than to have them passively following directions of mobile devices. Provided pathway images, followers can easily take correct pathways from many nearby pathways, which may not be possible if we resort to localization systems. As guiders capture images while they walk, the images can easily get blurred. One naive approach might be to first capture many frames regardless of image quality and later filter out these blurred ones, which incurs high energy cost in image capture as well as high computational overhead in image processing. To solve this problem, we infer the optimal time to capture the image with accelerometer readings and trigger image capture to ensure image quality.

Our system intelligently identifies shortcuts and plans optimized routes for users visiting multiple destinations. The trip planning would be trivial if the floorplans and accurate localization services are immediately available. However, as navigation traces are independently provided by different guiders without coordination (e.g., the blue trace to the restaurant and the red trace to the nearby shop in Figure 1), we need to design efficient and robust methods to merge the traces so as to find shortcuts. Unlike trip planning on floorplans, our design aims to determine the relative positions between route traces rather than their absolute locations on the floor.

3. Travi-Navi DESIGN

3.1 Functional Architecture

Figure 2 sketches the functional architecture. In the following, we briefly describe the key components.

Motion engine. Travi-Navi builds on prior research into IMU sensor (i.e., accelerometer, gyroscope, and compass) based dead-reckoning. It adopts a simple yet robust step detection method based on the rising edges of filtered accelerometer readings. It detects turns using a virtual bounding pathway that is more robust to sensor noise than inferences from instant heading changes. More importantly, the motion engine also outputs motion hints to assist image capture to reduce power consumption and ensure image quality.

Trace packing. Travi-Navi captures pathway images and samples WiFi and IMU sensors as guiders walk along pathways to destinations. It employs lightweight motion hints to indicate the best time for image capture to reduce power consumption. It automatically packs sensor data into navigation traces, where they are step-indexed by the relative positions to the beginning of the trace. The navigation traces can be shared directly to users or via a cloud server.

Navigation engine. With a guider's navigation trace, the navigation engine processes a user's instant sensor readings and generates navigation instructions. It first locks on users to the trace, typically at the entrances, using WiFi fingerprints. As entrances are typically distant to one another (>20m), Travi-Navi can confidently lock on users. As users walk towards a destination, it incorporates WiFi fingerprints and magnetic field distortions to complement IMU-based dead-reckoning. It accurately tracks users with respect to the guider's trace and generates corresponding position-indexed pathway images and instructions. If users want to visit multiple destinations, it automatically detects shortcuts and plans routes.

Note that the motion engine and the trace packing module are common to both the guider and the follower applications. There is a slight difference in sensor data collection. Guiders need to scan WiFi fingerprints at higher frequencies, whereas followers may scan WiFi less frequently to save energy. The navigation engine is solely used by followers.

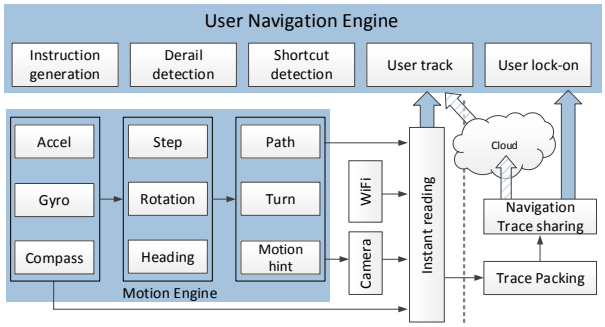


Figure 2: Architecture of the Travi-Navi system. The motion engine is common to both the guider and the follower.

3.2 Vision-guided navigation

Humans have superior visual recognition capability. Provided pathway images, a follower can easily differentiate two pathways that are close to each other and identify the correct one. Our system leverages the wide adoption of cameras on mobile devices to provide intuitive visual directions to followers.

Motion hints for image capture. As images are captured while walking, the pathway images may get blurred due to camera shake. To capture a clear image, one may capture image frames at a high rate and filter out the blurred ones, which involves extensive computations and high power consumption. To solve this problem, we design a simple and effective method to predict image quality before capturing pathway images to reduce power consumption and ensure image quality. The idea is inspired by the observation of the correlation between image quality and different walking phases. Figure 3 shows the consecutive frames taken during one walking step. Due to body vibration and subsequent camera shake, the first few images are blurred, while the last few are sharper. When we examine a sequence of frames taken during several steps, we observe alternating image qualities exhibiting periodic patterns. We aim to capture only sharp pathway images and display them to users.

We quantify image qualities using the detectable image features (such as edges and corners) extracted by computer vision techniques. Intuitively, given the same shooting target, a larger number of detectable features indicate sharper images. For instance, the feature extraction algorithm detects 315 keypoints in the first frame (Figure 3(a)) and 405 keypoints in the last frame (Figure 3(f)), respectively. Figure 4 plots the number of features in each frame against the differential of the accelerometer magnitude. When a user steps down, we observe a spike in the differential of the accelerometer magnitude. In the figure, right after the user steps down, the image qualities drop mainly due to body vibration. Generally, the images taken during the sharp turnings are similarly blurred due to the rotational motion of the camera, though image quality soon recovers after the user finishes a turn.

In Travi-Navi, we predict an optimal time for image capture by exploiting the average-crossing point on the rising edge of accelerometer magnitude. At such instants, mobile devices are generally stabilized during a step. Thus, Travi-Navi can automatically trigger image capture while a guider is walking. In addition, Travi-Navi also measures orientation and rotation changes and avoids capturing images during rapid turns. We note that although the low quality images are not captured, Travi-Navi does record the turns and inform users during navigation. In addition, when the

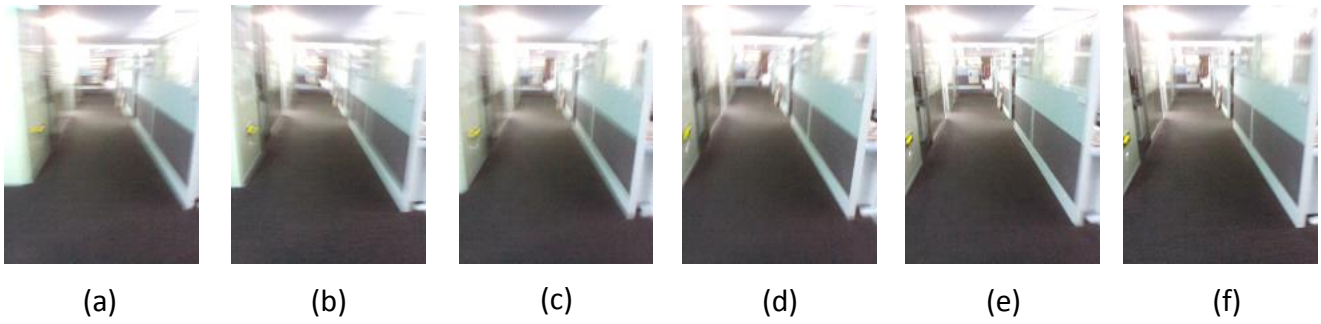


Figure 3: Consecutive frames during one walking step (taken by Samsung Galaxy S2).

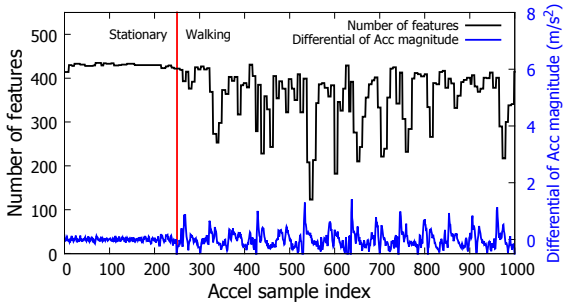


Figure 4: Image quality drops when user steps down.

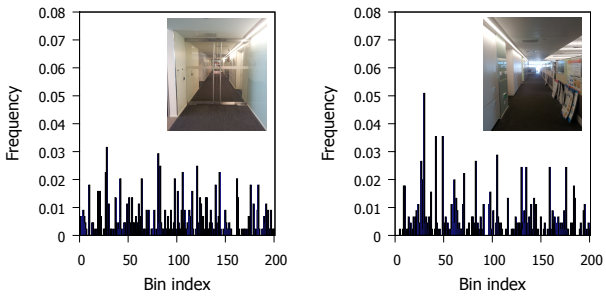


Figure 5: Image feature histogram.

guider finishes turning (normally a 5-step duration) and high image quality is again ensured, Travi-Navi quickly resumes image capture without missing the start of new pathways.

Usage of captured images. The clear images captured in a guider's trace serves two purposes. The primary purpose is navigation guidance, which implicitly exploits a follower's visual recognition capability to help navigation. To this end, pathway images and sensor signals are step-indexed in a guider's trace. Travi-Navi tracks followers with respect to the guider's trail and present proper pathway images to followers during navigation.

Since indoor environments exhibit distinct visual features (e.g., ambient color, geometric shapes, etc) [6], smart mobile devices may infer a follower's position on a guider's trail by comparing an image taken by the follower against the guider's images. Thus, when a follower (e.g., a Google Glass user) can easily take images during a trip, Travi-Navi may also incorporate images into the user tracking process. A mobile phone user is unlikely to take images continuously. However, the user may want to take pictures and retrieve the most similar images from the guider's traces. Then, the user can select and confirm the image of the pathway the user is actually on and start the navigation from there.

Image matching and retrieval. We extract image features of local geometric shapes as well as ambient colors and carry out image matching. Image matching operates in a pipeline. First, discriminative points are extracted for each image, called keypoints [19, 27, 20, 18]. The keypoints in each image can be viewed as visual "words" describing the image [17]. Similar visual "words" are clustered to form a "dictionary". Then, each image is described with a histogram, measuring the frequency of each word in the dictionary. We compute the similarity of two images by comparing their histograms. The histograms of images are represented with fixed-sized vectors whose length equals the dictionary size (i.e. 200) as shown in Figure 5.

To retrieve the most similar images, Travi-Navi incorporates color histograms (describing ambient colors) to extend the keypoint histograms (describing local geometric shapes). As images taken on each pathway capture similar ambiences, Travi-Navi clusters the images taken on the same pathway into one group. Travi-Navi labels image groups with pathway segment IDs and trains the linear support vector machine (SVM) as an image classifier. Travi-Navi also adds image shooting angles inferred from device orientation to image matching. In practice, the pathway images may appear similar and in this case Travi-Navi may fail to differentiate them accurately. Travi-Navi can enrich the diversity by incorporating other sensing modalities such as WiFi and magnetic field measurements.

3.3 Lock-on, tracking and deviation handling

Lock-on at entrances. When a follower enters a mall and starts Travi-Navi, Travi-Navi first needs to lock on to the follower. As the entrances of buildings (e.g., malls) are generally distant (>20m), it is likely to observe distinct WiFi signals at different entrances. Thus, Travi-Navi scans WiFi signals and compares them against the WiFi signals in guider's trace and locks on to the follower at the entrance. For shopping malls with multiple floors, guiders may not collect traces directly from mall entrances. Instead, they may collect the traces from lift lobbies or staircases, i.e., entrances to the floor. Lifts and staircases can be reliably identified using WiFi coverage information and accelerometer readings [32]. We discuss how we lock on to users in uncharted areas in §3.4.

Tracking follower. Travi-Navi adopts the particle filter [25] to track the user's progress on the guider's trails. After locking on the user at an entrance, Travi-Navi generates particles spread around the entrance to approximate the user's location. Each particle represents a possible position of the user and is updated according to IMU sensor based dead-reckoning [11, 16, 25, 32, 33]. Each particle is weighted according to the measurements of multiple sensing modalities. As such, the less likely particles are gradually filtered out and the centroid of the particles approximate the actual position of a follower. We describe the step detection and heading measurement techniques for completeness.

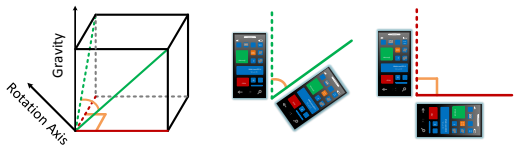
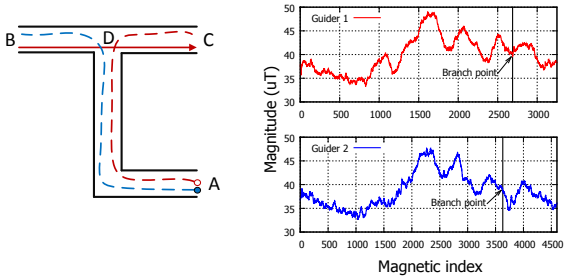


Figure 6: Orientation measurement. Travi-Navi projects the gyroscope measurements to the ground plane.



(a) Walking Paths (b) Magnetic Measurements

Figure 7: Two walking paths with overlapping segments and corresponding magnetic field measurements.

Dead-reckoning: Travi-Navi uses the weighted moving average to filter out noise in accelerometer readings. It tracks the average and variance of accelerometer magnitude. Stationary and walking states are differentiated using an empirical threshold on variance. To detect steps, it searches rising edges of accelerometer magnitude. For heading measurement, as users naturally hold mobile phones, the device heading generally aligns with walking directions. However, due to the severe magnetic field distortion inside buildings, the compass does not always point the heading direction [9, 39]. As the gyroscope is immune to magnetic field distortion and can measure rotations by integrating angular velocities, we leverage the gyroscope and fuse it with compass [38]. We have observed that natural holding gestures tend to tilt up phones, which may misalign the device reference frame and the world coordinate system as illustrated in Figure 6.

We have found that when a user holds a phone with a 45° angle to the earth surface and takes a 90° turn, the gyroscope only measures 60° around the rotation axis. We project the rotation measurement to the world coordinate system to measure the actual turning angles leveraging the gravity sensor [21].

To compensate for the differences in step length and heading measurement noise, a zero mean gaussian noise is added to each particle's step length and the measured heading directions in dead-reckoning. If Travi-Navi detects steps, each particle is then updated to a new location. After that, each particle is weighted according to the follower's instant WiFi signal as well as magnetic field readings. Travi-Navi measures the centroid of weighted particles to approximate a follower's position. The weighted particles are resampled after weight normalization. The tracking process is repeated until the user reaches the destination.

Magnetic signal weighting: Next, we describe how Travi-Navi weights each particle according to observed magnetic field readings. Although the magnetic directed dead-reckoning suffers severe errors inside buildings, the magnetic distortions are stable and provide discriminative power [9, 32]. Figure 7 shows the walking paths of two guiders with overlapping pathway segments, and the magnetic field distortions observed on the trails. We see that the magnitude of magnetic field varies along the trails. The patterns of the common segment are very similar, which confirms the stability of the magnetic field [9, 32].

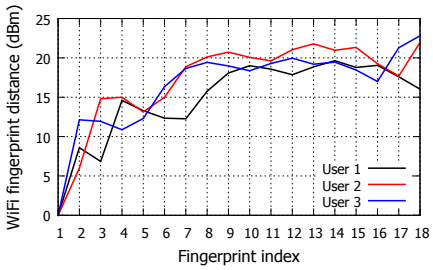


Figure 8: WiFi fingerprints are collected as users walk along a pathway. We see WiFi similarity decreases as the walking distance increases.

Leveraging the distinctive patterns of the magnetic field, Travi-Navi finds the most similar subsequence of magnetic measurements on the guider's trace to track a follower. Travi-Navi uses the dynamic time warping (DTW) algorithm [28] to compare the sequence similarity to compensate for the difference in speeds between the guider and the follower. DTW searches for the best alignment that minimizes the total cost using standard dynamic programming and the cost is defined as the difference of aligned magnetic magnitudes. When particles need to be updated, their weights are set according to their DTW costs via a Gaussian kernel. In particular, for the ith particle, its weight is set as follows:

$$weight_m^i = e^{-\frac{d_i}{k}} \quad (1)$$

where d_i is the DTW cost and k is a tunable parameter.

In our experiments, we observed that the magnetic field readings may exhibit similar patterns on distant pathway segments. A longer walking trail enriches diversity and improves discernibility. As followers are tracked on a step basis, the gain from further increasing the monitoring window (e.g., > 5steps) becomes marginal. We balance the computation overhead and discernibility by setting an empirical 5-step window.

WiFi signal weighting: With WiFi signal strength measurements, Travi-Navi gives greater weights to the more likely particles and iteratively filters out the less probable ones. We denote the user's and the guider's WiFi fingerprints as $F_t^u = (R_1^u, R_2^u, \ldots, R_n^u)$ and $F_t^g = (R_1^g, R_2^g, \ldots, R_n^g)$, where R_i^j is the RSSI of the ith AP observed by j at time t. In practice, we set -99dBm to the RSSI of undetectable APs. The distance between fingerprints F^u and F^g is measured by [35]

$$Dis(F^u, F^g) = \frac{1}{n}\sum_{k=1}^{n} |R_k^u - R_k^g|. \quad (2)$$

Similarly, we measure the geographical distance (i.e., Euclidian distance) of dead-reckoning positions between P^u and P^g by

$$Dis(P^u, P^g) = ||P^u - P^g||. \quad (3)$$

Figure 8 plots the WiFi distance $Dis(F_1, F_t)$ of three different walks along the same pathway. We see that the distances of WiFi fingerprints to the first fingerprint gradually increase as the users walk farther. Due to signal noise, larger distances may not always mean farther locations (e.g., $Dis(F_1, F_4) > Dis(F_1, F_7)$ for user 1). Based on this trend, however, we see that within a small number of steps (e.g., 5 steps) larger distances of WiFi fingerprints generally indicate farther locations on a pathway. In other words, the geographic distance and WiFi signal distance should be *correlated* on a pathway. Based on this observation, Travi-Navi weights the particles with the correlation of the Euclidian distances and the WiFi fingerprint distances. Let $\overrightarrow{D}^i_{\text{Euc}} = \{Dis(P^i, P_t^g)\}$ and $\overrightarrow{D}^i_{\text{WiFi}} = \{Dis(P^i, P_t^g)\}$, $(1 \le t \le m)$ denote the Euclidian

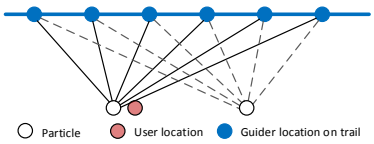


Figure 9: The more probable particle is weighted higher based on the correlation of Euclidean and WiFi distance.

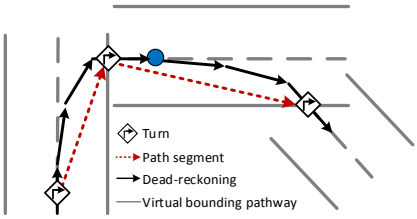


Figure 10: Turn detection. Travi-Navi detects turns if the user walks through the virtual bounding pathways.

distances and WiFi distances from the ith particle to the guider's m dead-reckoning positions and associated WiFi fingerprints, respectively. Then, the ith particle is weighted as follows

$$weight_w^i = \begin{cases} Corr(\overrightarrow{D}^i_{\text{Euc}}, \overrightarrow{D}^i_{\text{WiFi}}) & \text{, if} > 0 \\ 0 & \text{, otherwise} \end{cases} \quad (4)$$

As illustrated in Figure 9, a particle closer to a user's real position would be weighted higher than a distant one, as the distances from the more probable particle would be more correlated to the measured WiFi distances. Note that the use of correlation as a similarity metric naturally incorporates multiple WiFi fingerprints, enhancing robustness against WiFi signal strength fluctuations.

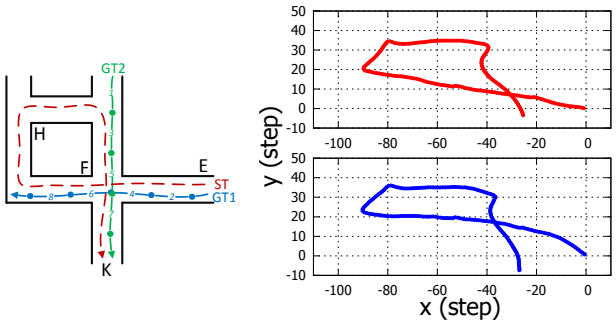
Magnetic and WiFi Fusion: In Travi-Navi, we assign equal weights to magnetic and WiFi signals and multiply the two weights to fuse them as follows

$$weight^i = weight_m^i \cdot weight_w^i$$

We note that one may evaluate the reliability of magnetic and WiFi signals, e.g., by measuring their variation over time, and assign different weights accordingly. Such a weighting strategy may improve the tracking accuracy compared with the current equal weighting method. In practice, the current method provides sufficient tracking accuracy for the purpose of Travi-Navi. To reduce power consumption, a follower may reduce the WiFi sampling rate and accordingly reduce the weight of WiFi measurements. If the WiFi scan is saved, particles are weighted using only magnetic signals. If query pathway images are captured, we may assign higher weights to the retrieved pathway accordingly. It should be noted that the optimal weighting strategy remains as an open question.

Deviation detection and handling. If followers deviate from the correct path, Travi-Navi should be able to detect the deviation events and promptly alert the followers. Additionally, followers may intentionally take short detours to avoid obstacles. We detect deviations using two simple intuitions. First, if there are mismatched turns between a guider's trace and a follower's dead-reckoning, Travi-Navi changes the tracking state to a pre-alert state. Second, if the user indeed deviates, the centroid position of particles will not change much, which is substantially different from normal walking. Thus, we conclude a deviation event when observing mismatched turns and no or slight centroid position changes for a few steps.

Turn detection: Many methods detect turns by measuring instant heading changes during a walk, which often results in false detections due to sensor noise. Based on the observation that the pathways and corridors are generally long, straight and narrow areas, we attempt to confine dead-reckoning trails with virtual bounding pathways, as depicted in Figure 10. In particular, Travi-Navi builds a 4-step wide virtual bounding pathway with the direction of the first two steps (to take into consideration body swing rhythm) [5]. Travi-Navi tests whether a user walks through the virtual bounding pathway after updating the user's position. If the user deviates from the original direction and walks outside the virtual bounding pathway, Travi-Navi detects turns and builds a new virtual bounding pathway with the heading direction of the last two dead-reckoning steps; otherwise Travi-Navi considers the user to be walking without turning. Virtual bounding pathway based detection is more robust to the noise in heading measurements. In wide areas, e.g., parking lots, users tend to take short paths and their dead-reckoning trails can be confined with several concatenated virtual bounding pathways. The pathway images are more helpful in wide open areas, as the users can easily identify the scenes.



(a) Actual Walking Path (b) DR Captured ST Traces

Figure 11: Traces with crossing shortcuts.

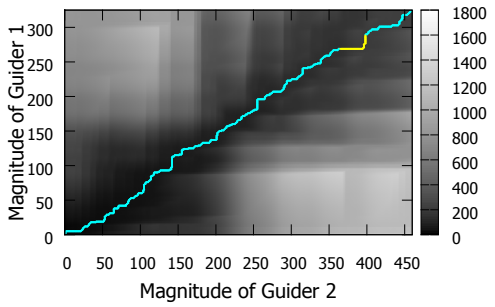
Deviation handling: When Travi-Navi detects a possible deviation event with a detection window of 4 steps, it immediately marks the deviating position and reflects the change in UI by changing the guiding arrow from a solid to dashed lines. It also displays the dead-reckoning trail to inform the user. Followers may intentionally take short detours to avoid obstacles and return to correct routes. If the particles revive in a short amount of time, Travi-Navi resumes user tracking and transits to the navigation state. Travi-Navi confirms a deviation event with a detection window of 6 steps. Once a deviation event is confirmed, it prompts the user and shows instructions to guide the user back to the deviating position. Meanwhile, it also shows the pathway image of the deviating position.

Travi-Navi does not assume the user will truly follow the instructions to walk back. The user may continue the exploration. Travi-Navi tries to relock on to the user by continuing the particle filtering process as the user walks. When the particles reconverge on the guider's trail indicating that the user has returned to the correct path, it prompts the user regarding the successful relock on and continues the navigation.

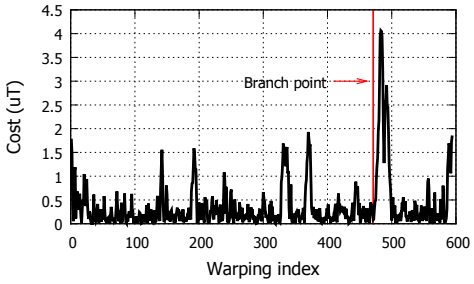
3.4 Finding shortcuts

A navigation service should be able to find shortcuts and plan trips for users. We identify two situations in which we may find shortcuts for users.

Case 1: Find shortcuts from multiple traces with overlapping segments. Users may want to visit multiple places (e.g., a restaurant after shopping). As illustrated in Figure 7(a), a user wants to visit shops at location B and C starting from entrance A. The user downloads two traces (A-B and A-C), and navigates to B first. Now to visit C, Travi-Navi needs to find the shortcut and navigate the user from B to C directly.



(a) DTW cost matrix.



(b) Cost of each warping

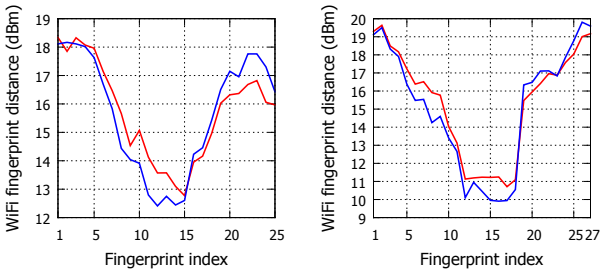
Figure 12: Magnetic field readings on common pathways are similar and can be used to detect shortcuts.

Case 2: Find shortcuts from crossing traces. Multiple guiders' trails may only cross each other and lack long overlapping segments. This case is more common for self-helping users such as car finders. Figure 11(a) depicts such cases where GT1 and GT2 are two crossing traces from different guiders, and ST shows a self-helping user trace. In ST, the user visits a shop at H and then another shop at K. To navigate the user back to entrance E, ideally we would guide the user to crossing point F and then to E.

The task of finding shortcuts would become simple if the navigation traces to multiple destinations can be properly merged. The key is to accurately find the overlapping segments (for Case 1) and the crossing points (for Case 2).

Finding overlapping segments. One intuitive approach to merge the guiders' traces is through trace replay – to emulate the navigation process of one guider on another guider's trace. However, this approach does not work well. First, the different signals prior to the overlapping segment (due to the different starting points) will drive divergent particle distribution. It therefore negatively affects the convergence of particles when it comes to the overlapping portion. Consequently, the identified overlaps will be much shorter than the actual overlapping segments. Short overlaps may even be missed. Secondly, the particle filtering, especially before particle convergence, involves relatively high computation. Unlike user tracking, which amortizes the computation over the whole navigation process, Travi-Navi needs to efficiently merge the traces in shortcut detection. Prior methods detect distinguishable signal landmarks inside buildings (e.g., unique WiFi fingerprints, motion patterns) as anchors to correct dead-reckoning drifts and align traces. However, as reported, such landmarks may be few in number and not always detectable [32, 35, 29].

Travi-Navi adopts lightweight trace merging algorithms to efficiently find overlapping segments and reliably merge multiple traces, even if no landmarks can be found. In particular, we design a magnetic signal based detection and WiFi signal based verification process.



(a) North to South path (b) East to West path

Figure 13: "V"-shaped trend of WiFi fingerprint distances at crossing points.

Magnetic field based detection: As in the tracking case, we also apply Dynamic Time Warping (DTW) [28] to counter for possibly warped magnetic signal sequences, except that here, DTW is applied to the whole sequence. To reduce the computation and memory overhead involved in processing the data sampled at the highest rate, we perform a 10-fold downsampling, i.e., averaging over 10 samples, without affecting the shapes of two magnetic sequence outlines.

Figure 12(a) plots the cost matrix of DTW after the downsampling for matching the two magnetic sequences shown in Figure 7(b). The line in Figure 12(a) highlights the minimum cost warping of the two sequences. In the upper-right corner of the figure, we observe a "⌟"-shaped turn (highlighted in yellow) on the minimum cost line. This is because after the branch point the magnetic fields exhibit distinct patterns and no good matches can be found between the two sequences. Figure 12(b) plots the costs involved in each warping alignment along the minimum cost line. We see that due to the lack of good matches, the costs immediately increase after the branch point. Also, unlike other narrow short spikes due to noise, the tallest spike remains high for a relatively long period. Travi-Navi sets empirical thresholds for the cost and window size to detect the branch point.

WiFi based verification: Travi-Navi performs post verification using *a sequence of* WiFi observations to ensure correct detection of overlapping segments. If DTW indeed finds correct overlapped segments, the WiFi fingerprints observed on one guider's trace would be very similar to those observed on the other.

In Travi-Navi, guiders perform frequent WiFi scans whereas followers may scan less frequently. We compensate for different WiFi scan speeds on different devices by pairing WiFi scans at the closest positions when calculating the average WiFi signal distances. Evidently, the average distance between all WiFi pairs is small for correct alignment. However, if the segments are misaligned, the average distance will be large. The more misaligned, the larger the average distance. Through experiments, we have decided on an empirical threshold (10dBm) to determine a good alignment. Note that, unlike the tracking case where a single WiFi fingerprint is utilized, here we have to rely on *a sequence of* WiFi scans for robustness.

For overlapping segments with opposite directions, Travi-Navi can detect the overlaps by reversing one of the two traces and apply the above detection method.

Finding crossing points. Since the crossing traces only have small overlapping areas, the above overlapping segment detection methods cannot detect crossing points. While the traces contain WiFi fingerprints, they cover only a very small portion of the whole indoor radio environment. We cannot confirm a crossing point by testing if some fingerprints in one trace are localized on another trace with conventional WiFi localization techniques [7]. We note

that the small distance in WiFi fingerprints is necessary but not sufficient to identify shortcuts.

Recognizing that closer positions should show more similar WiFi fingerprints as evidenced in Figure 8, and further inspired by [29] in which the trend of AP signal strengths is used to detect WiFi-Marks, we design a crossing point detection method by comparing the trend of *average gross distances* of fingerprints in one trace to the fingerprints in the other. Figure 11 depicts an example of two crossing traces GT1 and GT2. Two guiders sample WiFi fingerprints as they walk along GT1 and GT2, respectively. We calculate the average gross distance – the average of the pair-wise WiFi distances to a window of consecutive fingerprints in the other trace, e.g., the average WiFi distance of one fingerprint on GT1 to those on GT2. Pair-wise WiFi distance is calculated using Eq. 2.

Figure 13 shows the average of gross WiFi distances between WiFi fingerprints in the portions near the crossing point of the two user traces shown in Figure 11(b). We clearly see the "V"-shaped turning of the trend. It is important that such trends should be *mutual*, i.e., the same trend should exist when checking fingerprints on one trace against those on the other, and vice versa. When two paths meet at a "T" crossing point, Travi-Navi cannot identify the shortcut by detecting the mutual decreasing-and-increasing trends in WiFi fingerprint distances. In practice, however, since the two paths are likely to join and form overlapping segments, Travi-Navi can detect such shortcuts leveraging magnetic readings.

Navigation instruction on shortcuts. Once overlapping segments or crossing points among multiple traces are found, we can merge them by grafting one trace to another trace. The only subtlety of navigation on shortcuts is that certain trace segments may need to be played in reverse.

User lock-on from uncharted areas. A follower may not start Travi-Navi at entrances and want to navigate from uncharted areas using a guider's trace. In this case, the follower may first walk around and record a walking trace. Travi-Navi reuses the above shortcut identification modules to detect whether the follower's walking trace has any overlapping segments or crossing points with the guider's trace. Once Travi-Navi confirms that the follower is on the guider's trail, it can lock on the follower and starts navigation.

4. SYSTEM EVALUATION

In this section, we present the evaluation of key functional components of Travi-Navi. We then evaluate Travi-Navi in representative indoor environments for a better understanding of Travi-Navi's effectiveness and limitation.

4.1 Implementation

We implement Travi-Navi on the Android platform (version 4.2.2). The current version of Travi-Navi involves 6k lines of Java implementing all the functional modules of functional architecture (Figure 2). We adopt the OpenCV library (version 2.4.6) written in C to implement the image processing and image matching via JNI. In particular, we adopt the ORB algorithm [27] as it is orders of magnitude faster than SURF and SIFT [19] and can extract image features in real time on mobile devices (e.g., Samsung Galaxy S2 with one dual-core 1.2GHz processor). We adopt the bag-of-visual-words algorithm [17], which uses k-means to cluster image features. For image classification, we adopt the linear SVM with the default optimal parameter setting of OpenCV [4]. The image resolution is set to 320×240 which balances the image quality and the processing overhead. Each image is saved as a JPEG file and the file size is around 20KB. We enable the auto-focus in image capture. Travi-Navi packs the guider and the user functional modules into one application and reuses the common modules. All experimental results were obtained via online processing on mobile devices and visualized offline if not specified otherwise.

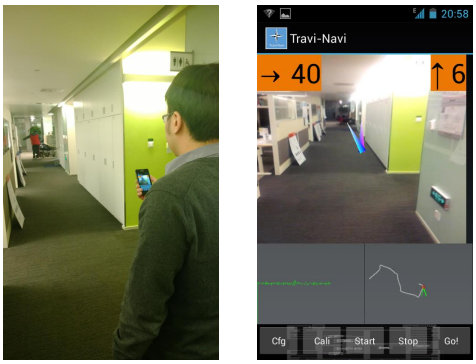


Figure 14: A user trying out Travi-Navi (Left), and the UI snapshot of our Travi-Navi client (Right).

The user interface presents navigation instructions as illustrated in Figure 14. The center of the UI shows guider's pathway image. As the follower moves forward, Travi-Navi tracks the follower and updates the guider's pathway image accordingly. In addition to the pathway image, Travi-Navi also presents the turning instructions (i.e., turn right by 40 degrees at the next turn, presented as $\rightarrow 40$) and the number of remaining steps to the next turn (i.e., 6 steps, presented as $\uparrow 6$). In addition, it also measures the instant heading direction (blue arrow) and presents the guider's heading direction on the path (purple arrow).

4.2 Evaluation

We tested Travi-Navi on a variety of Android mobile devices (Samsung Galaxy S2, S4, Note3, HTC Desire, and HTC Droid Incredible 2). Due to limited availability, Travi-Navi has not yet been evaluated on wearable devices such as Google glass.

We conducted experiments on both an office building floor and the first floor of a large shopping mall with a testing area of about 1900m^2 and 4000m^2 during different times of day. The lighting conditions during business hours allow guiders to capture bright pathway images. The guiders record traces with mobile phones held in an upright attitude and capture pathway images using back cameras as they walk to destinations. As the surroundings can be occluded, guiders need to find a less crowded time to capture pathway images. Followers naturally hold mobile phones and follow navigation instructions. For the purpose of this evaluation, Travi-Navi recorded the followers' traces as well. Overall, we collected 12 navigation traces covering all the main pathways of the testing areas' entrances. The total length of navigation traces was about 2.8km. We recruited 4 volunteer followers who walked many times on different routes, with a total walking distance of around 10km in the experiments.

4.2.1 Motion hints

Image quality. Images taken by guiders during the walk provide visual guidance to users. Due to camera shack, the images could become blurred. We evaluate whether motion hints could improve the image quality. We use the number of detectable ORB features in images as the quality metric to quantify the image quality. Generally, computer vision techniques can detect more features in sharper images than blurred ones. We also tested other image quality metrics (e.g., the maximum value of Laplacian transformation), which are highly correlated with our metric but more subject to image noise.

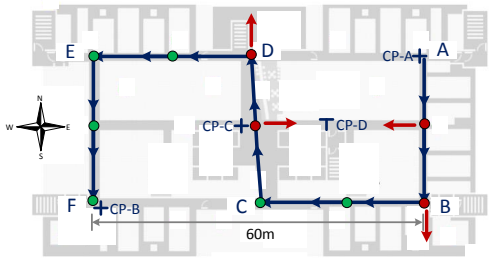


Figure 15: Office building navigation trail.

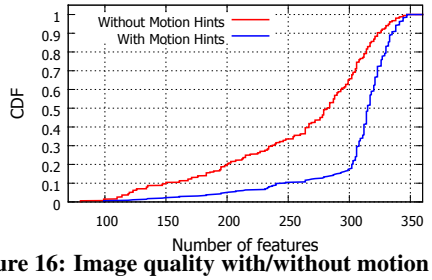


Figure 16: Image quality with/without motion hints.

We walked along the same route and captured images with/without the assistance of motion hints, respectively. We measured the number of features in the images captured during walks inside the office building. Figure 16 plots the CDFs of the number of features with/without the assistance of motion hints. The results show that the motion hints can predict better shooting opportunities and effectively capture sharper pathway images.

Pathway image retrieval. Smart mobile devices may roughly infer a user's location (e.g., pathway-level localization) using image matching techniques. We integrate the local geometric shapes described using ORB features and the ambient color histograms to form image features. We capture images triggered by motion hints at each step on the trail from A $\rightarrow$ B $\rightarrow \cdots \rightarrow$ F in our office building as depicted in Figure 15. We label the images with the pathways (i.e., A $\rightarrow$ B, B $\rightarrow$ C, . . . , E $\rightarrow$ F) and train the linear SVM. SVM training takes several seconds on mobile devices (Samsung Galaxy S2) but we note that it only involves a one-time training cost. A guider may train the SVM using a server and share the SVM classifier with users to classify images in real time on mobile phones.

Users take images on pathways and query Travi-Navi for their locations. Before triggering image capture, Travi-Navi consults the shooting angle by looking up the instant device orientation. If the device is facing either the floor or ceiling, Travi-Navi does not perform image capture or matching, and inform the users to adjust the shooting angle. In the experiment, each of the 4 users took 25 pathway images to query locations. In Figure 17, we found that Travi-Navi achieves pathway-level localization with reasonably high accuracy by combining the ORB feature and a color histogram. The accuracy on path C $\rightarrow$ D is lower. Examining the images on the path, we found that the lower accuracy is mainly due to the line of sight blockage. Travi-Navi needs to capture a query image with a substantial portion that has not been occluded to retrieve the correct image in the navigation trace.

4.2.2 User tracking and navigation

User lock-on. Travi-Navi locks on users at entrances by comparing observed WiFi fingerprints against those in navigation traces. We collected WiFi fingerprints at 6 entrances of the shopping mall. The number of observable WiFi APs varied from 6 to 13 at the entrances. The geographic distance between the entrances closest to each other was around 20m. We randomly selected one WiFi fingerprint at each entrance and computed the WiFi distances to the other fingerprints at the same or different entrances. The experiment results are plotted in Figure 18, which shows the maximum WiFi distance to the fingerprints collected at the same entrance and the minimum WiFi distance to fingerprints collected at different entrances. We found that the WiFi fingerprint distances at the same entrance were substantially smaller than to fingerprints at different entrances. This gap allows Travi-Navi to lock on to users at the entrances (separated by >20m) with 100% accuracy. We also tested in our office building at the main entrance and 4 emergency exits on the same floor (separated by >25m) and achieved correct lock-ons as well.

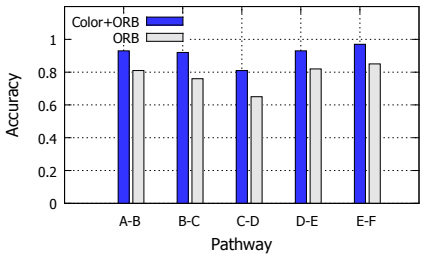


Figure 17: Pathway image retrieval performance.

Tracking follower. Travi-Navi tracks a follower's progress on the guider's trace and gives directions. We measured the tracking error to evaluate the promptness of instructions and tested whether Travi-Navi can navigate users to their destinations. A guider collected navigation traces in the office building following the trails plotted in Figure 15. The guider took 202 steps and 2min to walk the trail (A $\rightarrow$ F), with a total length of around 150m. In practice, a guider (e.g., restaurant owner) should take the most convenient path for followers (e.g., customers). To obtain the ground truth, the guider taps the mobile phone to record timestamps when the guider reaches the checkpoints depicted as green and red dots in Figure 15.

Travi-Navi navigated the 4 followers starting from A to F. We note that the followers were not informed of the navigation routes, the final destination, or the checkpoint locations. In the experiments, all the followers successfully reached the destination under the direction of Travi-Navi. During the navigation, Travi-Navi recorded timestamps along with the instructions presented to users. Meanwhile, a shadow person followed behind the volunteers and recorded the timestamps when the volunteers arrived at checkpoints.

We measured the tracking errors by the offset between the instructions presented to followers and the instructions recorded by the guider at each checkpoint. The offset results are shown in Figure 19, in the unit of steps as instructions are updated on a step basis. The experiment results show that Travi-Navi has small tracking errors within 4 steps. We carried out a user study with the 4 users after they reached the destination. According to our user study, user 1 and user 2 specified the direction as "almost synchronized", while user 3 and user 4 experienced slightly delayed and early directions, respectively. As users normally took around 0.6s per step, the time offset of instructions was within 3s for all users.

Deviation detection. Travi-Navi should quickly detect deviation events and notify users. In the experiment of deviation detection, Travi-Navi navigates the 4 volunteering followers from A to F as in the previous user tracking experiment. To measure the effectiveness of deviation detection, we intentionally asked the 4 users to deviate at 4 bifurcating points depicted as red dots and follow the pathways indicated as red arrows in Figure 15. Travi-Navi enters the pre-alert stage with a detection window of 4 steps and confirms a deviation with a detection window of 6 steps. Thus, we set the

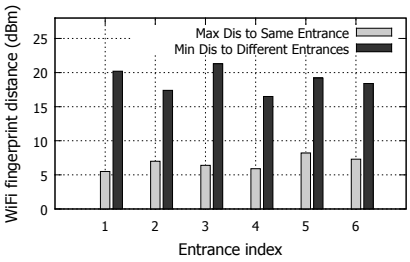


Figure 18: Distances between WiFi fingerprints at 6 entrances.

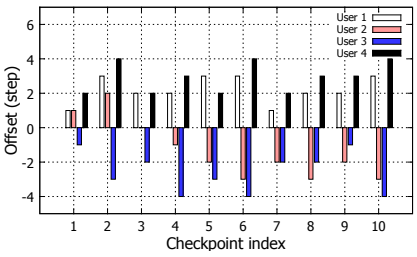


Figure 19: User tracking accuracy for different users at 10 checking points.

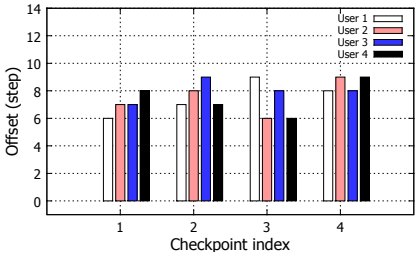


Figure 20: Deviation detection for different users at 4 turns.

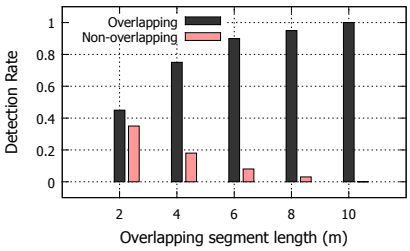


Figure 21: Overlapping segment detection results.

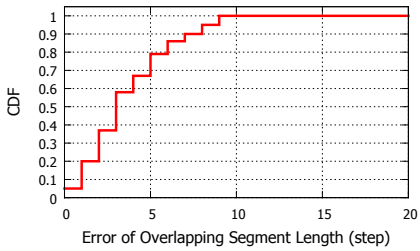


Figure 22: CDF of error of overlapping segment length.

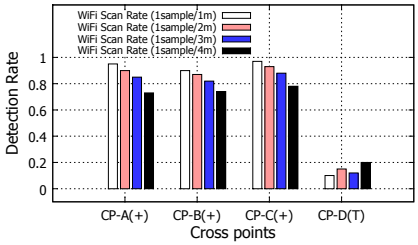


Figure 23: Crossing point detection results.

deviation detection window to be 6 steps. Travi-Navi recorded the instructions (including deviation alerts if any) presented to users.

We present the steps that users took before Travi-Navi alerted the users in Figure 20. We found that Travi-Navi detected the deviation events within 9 steps. We also measured deviation detections that were false positives. We walked along the trail from A to F for 4 rounds (approximately 600m in total) and observed only 3 false alarms. Note that in Travi-Navi, we set a deviation detection threshold of 6 steps. When we decrease the threshold, Travi-Navi triggers more false alarms, while in return being more prompt in detecting deviation events.

4.2.3 Shortcut identification

We examine whether Travi-Navi can accurately identify shortcuts among guiders' traces.

Overlapping segments. We collect 100 walking traces with different overlapping segment lengths and conduct shortcut identification. As a user may only want to be navigated to a small number of destinations, Travi-Navi only needs to reliably identify the shortcuts among a dozen traces. We note that two walking trails of opposite directions may also have overlapping segments. To solve this problem, Travi-Navi reverses one of the two traces and identifies overlapping segments with opposite walking directions (§3.4).

Figure 21 plots the success rates with varying overlapping lengths. We found that Travi-Navi achieved a 90% success rate when the overlapping length was around 6m, and detected all overlapping segments when the length was 10m. The success rate decreased as the overlapping length became shorter. We also measured the false detection rates by matching traces with varying lengths against non-overlapping traces. In the figure, shorter traces tend to have higher false detection rates. When the trace length was 6m, the false detection rate decreased to 8% and no false detection was observed when the length reached 10m.

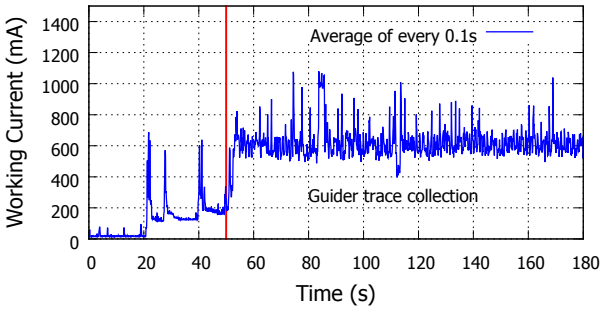
Among the correctly detected overlapping segments, we plot the CDF of differences between the actual overlapping segment length and the detected one in Figure 22. We found that about 50% of detected overlapping segments are within 3 steps and all detection errors are within 9 steps. Considering the corridor widths in practice, such accuracies are sufficient for shortcut detection.

Crossing points. We evaluate Travi-Navi's performance in detecting crossing points with the mutual trends in WiFi fingerprint distance. To conduct the experiment, we scanned WiFis at 3 crossing points (CP-A, CP-B, and CP-C) indicated by "+" in Figure 15. In addition, we also scanned WiFi at a "T"-shaped crossing point (CP-D). To study the detection performance with different WiFi scan frequencies, we first collected one WiFi fingerprint every 1m (i.e., 1sample/1m), and downsampled WiFi fingerprints to emulate lower scanning frequencies. Figure 23 plots the success rates of detecting the crossing points. We found that the detection rates at the "+" crossing points were above 80% with 1 WiFi scan per 3m. Travi-Navi cannot detect "T"-shaped shortcuts, since such a crossing point does not exhibit decreasing-and-increasing trends *mutually* (§3.4). In practice, however, two paths meeting at "T"-shaped crossing points are likely to join and form overlapping segments, and Travi-Navi can reliably detect the overlapping shortcuts leveraging magnetic readings.

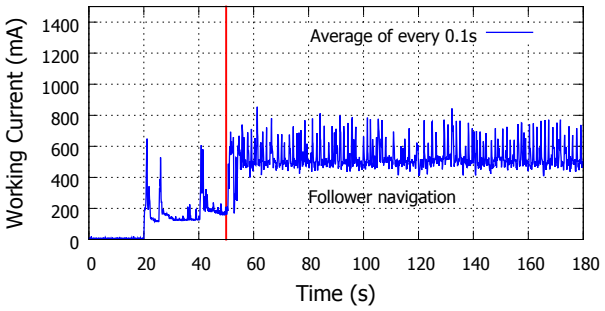
4.2.4 Energy consumption

As guiders collect navigation traces with IMU, WiFi, and cameras, it incurs relatively high power consumption. We measure the power consumption on various models of mobile phones using the Monsoon power monitor [3]. The power monitor directly supplies power to the phones and accurately tracks the current and voltage. To accurately measure the power consumption of Travi-Navi, we first turn off all background services and applications. The WiFi module was turned on and the screen brightness was set to auto-adjustment mode. All the sensor modules (e.g., IMU and WiFi) were sampled and processed in real time. As we need to wire the phones for measurement, we have to make the phones stationary. We synthetically triggered image capture and updated pathway images in Travi-Navi every 0.6s.

Figure 24 plots the working currents measured on the Samsung Galaxy S2 in different working modes as an example. The current was sampled at 5KHz using the power monitor and averaged over



(a) Guider mode.



(b) Follower mode.

Figure 24: Power measurement with Samsung Galaxy S2.

every window of 0.1s. In Figure 24(a), the phone was in sleep mode during the period from 0s to 20s. We woke up the phone at around 20s. We then unlocked the phone at around 25s. We launched Travi-Navi at 40s and started the trace collection at 50s. The trace collection finished at 180s. Similarly, Figure 24(b) plots the power measurement when Travi-Navi was working in the follower mode from 50s to 180s. We repeated the experiments 10 times and characterized the power draws in different modes in Table 1. The guider mode of Travi-Navi drew power at around 616.50mA and the expected battery life was 2.92h. The follower mode incurred less power consumption at around 532.20mA and the expected battery life was 3.38h for the SGS2 with an 1800mAh battery.

We measured the expected battery life of the Samsung Galaxy Note3 with a 3200mAh battery and HTC Droid Incredible 2 with a 1450mAh battery. The measurement results are presented in Table 2. Powered by a large battery, the expected life of Note3 was around 4.75h in guider mode, and 5.62h in follower mode, respectively. The expected battery life of HTC Droid Incredible 2 was 3.14h in guider mode and 3.89h in follower mode, respectively. The expected battery life time was shorter for the HTC Droid Incredible 2 compared with Note3, mainly because of its smaller battery.

The current version of Travi-Navi has not yet been extensively optimized for energy efficiency. Travi-Navi may reduce the image sampling rate to save energy and benefit from the energy efficient mobile vision techniques [18]. When magnetic field distortions exhibit sufficient discriminative patterns, Travi-Navi may disable WiFi scans and weight particles solely with magnetic measurements. Travi-Navi can also leverage the adaptive particle filtering techniques (e.g., KLD-sampling [12]) to reduce power consumption. Travi-Navi can benefit from energy efficient co-processor architectures for sensor fusion as well [30].

Table 1: Power consumption measurements on Samsung Galaxy S2 with an 1800mAh battery. We measure the average power, the average current, and the expected battery life.

Mode	Period	Power	Current	Battery Life
Sleep	0s–20s	75.39mW	20.41mA	88.20h
Idle	20s–50s	676.49mW	183.15mA	9.83h
Guider	50s–180s	2276.35mW	616.50mA	2.92h
Follower	50s–180s	1965.13mW	532.20mA	3.38h

Table 2: Expected battery life measurements in different modes on 3 models of mobile phones.

	SGS2	Note3	Incredible2
Battery Capacity	1800mAh	3200mAh	1450mAh
Battery Life (Guider)	2.92h	4.75h	3.14h
Battery Life (Follower)	3.38h	5.62h	3.89h

5. RELATED WORK

Indoor localization is an extensively studied field. Many systems such as Cricket [22] and PinPoint [37] achieve high position accuracy with dedicated hardware deployment. Many methods such as Radar [7] and Horus [36] leverage existing WiFi infrastructure. Some recent works explore the magnetic field for indoor localization [9, 26]. All these methods require labor-intensive site surveys. Participatory systems like PlaceLab [15], ActiveCampus [13], and LiveLabs [2] evaluate localization in the real world. Leveraging rich sensing modalities on smartphones, SurroundSense [6] infers logical surroundings from ambient signatures.

Many crowdsourcing-based indoor localization systems [32, 8, 35, 25, 10, 23] lack incentives to attract enough participation especially when the services cannot provide benefits. Unlike those works, Travi-Navi enables users to easily deploy their own navigation services without comprehensive localization services and thereby directly benefit from service deployment, while the crowdsourcing systems typically require huge number of volunteers who may not directly benefit from participation.

Technique-wise, Travi-Navi draws strength from prior inertial sensor based tracking [16, 35, 25], which typically assumes the availability of floor maps and confine dead-reckoning drifts with map constraints. Instead, Travi-Navi intelligently fuses magnetic and WiFi signals to accurately track a user's progress on a guider's navigation trace. Some works [32, 29] seek to construct indoor maps and bootstrap localization services exploiting walking traces shared by crowds. They identify various landmarks to merge user traces, which is similar to one of the shortcut identification cases in Travi-Navi.

Escort [10] navigates users to their friends inside buildings without relying on accurate localization services. Escort corrects dead-reckoning drifts leveraging crowd encounters and special audio beacons. Riehle *et al.* [26] navigate blind users leveraging magnetic distortions and give audio instructions. Unlike previous methods, Travi-Navi augments indoor navigation with motion vision by providing followers with the guider's pathway images. Travi-Navi attempts to enable a user to easily bootstrap navigation services without infrastructure support. Travi-Navi synthesizes radio and magnetic signals to correct IMU based dead-reckoning drifts and identifies shortcuts.

Recent advances in mobile sensing and computer vision open new research opportunities. OPS [20] locates distant objects by synthesizing GPS localization and images taken from different angles with the Structure from Motion (SfM). TagSense [24] tags images with locations and activities inferred from smartphone sen-

sors. FOCUS [14] indexes crowdsourced videos via content-based clustering with the complementary synergy of SfM and sensor hints. GigaSight [31] proposes a framework for crowdsourcing videos that enables deep content-based search. LiKamWa *et al.* [18] analytically characterize energy consumption of image and video capture and propose quality-energy tradeoff strategies. Travi-Navi augments indoor navigation with mobile vision, and avoids unnecessary power consumption and computation leveraging motion hints inferred from IMU sensors.

6. CONCLUSION

This paper describes our attempts to design, implement, and evaluate Travi-Navi, a vision-guided indoor navigation system. The key idea is to enable self-motivated users to easily deploy indoor navigation services without assuming a comprehensive indoor localization service or even the availability of floor maps. We incorporate magnetic field distortion and WiFi signals in particle filtering to ensure accurate user tracking. We further develop effective methods to detect shortcuts among overlapping and intersecting navigation traces. We devise a method to automatically capture high quality images while walking using motion hints. We implicitly leverage the users' visual recognition capability by providing them pathway images to correct slight direction ambiguity and enhance navigation experiences. Extensive experimental results and feedback from user trials confirm the effectiveness of Travi-Navi. We plan to further optimize Travi-Navi and reduce power consumption in the future.

7. ACKNOWLEDGEMENT

We would like to thank the anonymous shepherd and reviewers for providing constructive feedback and valuable input for improving the quality of this paper. We acknowledge the support of the NTU Nanyang Assistant Professorship (NAP) grant M4080738.020 and Microsoft research grant FY 12-RES-THEME-001.

8. REFERENCES

[1] High Accuracy Indoor Positioning (Nokia BLE). `https://research.nokia.com/`.

[2] LiveLabs. `http://livelabs.smu.edu.sg/`.

[3] Monsoon Power Monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[4] OpenCV. `http://opencv.org/`.

[5] G. Ananthanarayanan, M. Haridasan, I. Mohomed, D. Terry, and C. A. Thekkath. StarTrack: A Framework for Enabling Track-based Applications. In *ACM MobiSys*, 2009.

[6] M. Azizyan, I. Constandache, and R. Roy Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *ACM MobiCom*, 2009.

[7] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *IEEE INFOCOM*, 2000.

[8] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor Localization Without the Pain. In *ACM MobiCom*, 2010.

[9] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor Location Sensing Using Geo-magnetism. In *ACM MobiSys*, 2011.

[10] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury. Did You See Bob?: Human Localization Using Mobile Phones. In *ACM MobiCom*, 2010.

[11] I. Constandache, R. R. Choudhury, and I. Rhee. Towards Mobile Phone Localization without War-Driving. In *IEEE INFOCOM*, 2010.

[12] D. Fox. KLD-Sampling: Adaptive Particle Filters. In *NIPS*, 2001.

[13] W. G. Griswold, P. Shannahan, S. W. Brown, R. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong. ActiveCampus - Experiments in Community-Oriented Ubiquitous Computing. *Computer*, 37(10), 2004.

[14] P. Jain, J. Manweiler, A. Acharya, and K. Beaty. FOCUS: Clustering Crowdsourced Videos by Line-of-sight. In *ACM SenSys*, 2013.

[15] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *PERVASIVE*, 2005.

[16] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors. In *ACM UbiComp*, 2012.

[17] F.-F. Li and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *IEEE CVPR*, 2005.

[18] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy Characterization and Optimization of Image Sensing Toward Continuous Mobile Vision. In *ACM MobiSys*, 2013.

[19] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE ICCV*, 1999.

[20] J. G. Manweiler, P. Jain, and R. Roy Choudhury. Satellites in Our Pockets: An Object Positioning System Using Smartphones. In *ACM MobiSys*, 2012.

[21] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. In *ACM SenSys*, 2008.

[22] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-support System. In *ACM MobiCom*, 2000.

[23] A. Purohit, Z. Sun, S. Pan, and P. Zhang. SugarTrail: Indoor Navigation in Retail Environments without Surveys and Maps. In *IEEE SECON*, 2013.

[24] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. TagSense: A Smartphone-based Approach to Automatic Image Tagging. In *ACM MobiSys*, 2011.

[25] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort Crowdsourcing for Indoor Localization. In *ACM MobiCom*, 2012.

[26] T. H. Riehle, S. M. Anderson, P. A. Lichter, N. A. Giudice, S. I. Sheikh, R. J. Knuesel, D. T. Kollmann, and D. S. Hedin. Indoor Magnetic Navigation for the Blind. In *IEEE EMBC*, 2012.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE ICCV*, 2011.

[28] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[29] G. Shen, Z. Chen, P. Zheng, T. Moscibroda, and Y. Zhang. Walkie-Markie: Indoor Pathway Mapping Made Easy. In *USENIX NSDI*, 2013.

[30] H. Shen, A. Balasubramanian, E. Yuan, A. LaMarca, and D. Wetherall. Improving Power Effciency using Sensor hubs without Re-coding Mobile Apps. *Technical Report of University of Washington*, 2014.

[31] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. Scalable Crowd-sourcing of Video from Mobile Devices. In *ACM MobiSys*, 2013.

[32] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-drive: Unsupervised Indoor Localization. In *ACM MobiSys*, 2012.

[33] H. Wang, Z. Wang, G. Shen, F. Li, S. Han, and F. Zhao. WheelLoc: Enabling Continuous Location Service on Mobile Phone for Outdoor Scenarios. In *IEEE INFOCOM*, 2013.

[34] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge Location System. *ACM Trans. Inf. Syst.*, 10(1):91–102, Jan. 1992.

[35] Z. Yang, C. Wu, and Y. Liu. Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention. In *ACM MobiCom*, 2012.

[36] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *ACM MobiSys*, 2005.

[37] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala. PinPoint: An Asynchronous Time-Based Location Determination System. In *ACM MobiSys*, 2006.

[38] P. Zhou, M. Li, and G. Shen. Use It Free: Instantly Knowing Your Phone Attitude. In *ACM MobiCom*, 2014.

[39] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: A Generic Service for Indoor Outdoor Detection. In *ACM SenSys*, 2012.