Dong, Jiang; Noreikis, Marius; XIAO, YU; Yla-Jaaski, Antti

ViNav

Published in:
IEEE Transactions on Mobile Computing

DOI:
10.1109/TMC.2018.2857772

Published: 01/06/2019

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
https://doi.org/10.1109/TMC.2018.2857772
ViNav: A Vision-Based Indoor Navigation System for Smartphones

Jiang Dong, Member, IEEE, Marius Noreikis, Member, IEEE, Yu Xiao, Member, IEEE, and Antti Ylä-Jääski, Member, IEEE

Abstract—Smartphone-based indoor navigation services are desperately needed in indoor environments. However, the adoption of them has been relatively slow, due to the lack of fine-grained and up-to-date indoor maps, or the potentially high deployment and maintenance cost of infrastructure-based indoor localization solutions. This work proposes ViNav, a scalable and cost-efficient system that implements indoor mapping, localization, and navigation based on visual and inertial sensor data collected from smartphones. ViNav applies structure-from-motion (SfM) techniques to reconstruct 3D models of indoor environments from crowdsourced images, locates points of interest (POI) in 3D models, and compiles navigation meshes for path finding. ViNav implements image-based localization that identifies users’ positions and facing directions, and leverages this feature to calibrate dead-reckoning-based user trajectories and sensor fingerprints collected along the trajectories. The calibrated information is utilized for building more informative and accurate indoor maps, and lowering the response delay of localization requests. According to our experimental results in a university building and a supermarket, the system works properly and our indoor localization achieves competitive performance compared with traditional approaches: in a supermarket, ViNav locates users within 2 seconds, with a distance error less than 1 meter and a facing direction error less than 6 degrees.

Index Terms—Indoor mapping, indoor localization, indoor navigation, 3D modelling, mobile crowdsensing

1 INTRODUCTION

Indoor navigation systems for smartphones are crucial in complex indoor environments such as airports, shopping malls and museums. Unfortunately, the adoption rate of indoor navigation systems is still very low, even though initial efforts into deploying them were taken several decades ago. There are multiple reasons for the slow progress. First of all, indoor navigation requires fine-grained and up-to-date indoor maps for calculating navigation routes and searching for points of interest (POI). Certain solutions ask users to provide photos of floor plans in public venues as their indoor maps. However, most of them lack details, or have not been kept up to date. Therefore, they can rarely be used directly for indoor navigation. Second, even in the case where accurate indoor maps do exist, most systems rely on pre-scanned radio maps or pre-installed hardware [14], [19] for localization, which are expensive to install and cumbersome to maintain. Thus, we saw it worthwhile to investigate whether we could develop an alternative indoor navigation method that would not need pre-created indoor maps, pre-scanned radio maps or pre-installed hardware.

Smartphones today are equipped with high-resolution cameras, and mobile users are willing to share some of their photos publicly, e.g., via photo-sharing websites like Flickr or Instagram. Furthermore, researchers have proven that it is possible to use crowdsourced photos to build 3D models of indoor and outdoor environments via structure-from-motion (SfM) techniques [6], [9], [33], [36]. With SfM-based 3D models, it has a good potential to build maps and further navigation meshes. Furthermore, it is possible to provide image-based localization services based on feature matching. Inspired by this, we propose to create a system, ViNav, which utilizes crowdsourced visual data and SfM-based 3D models to solve the problems of indoor mapping, localization and navigation as a whole.

Our system is based on sensor-enriched 3D models that are bootstrapped and updated using crowdsourced visual and sensor data collected from smartphones. ViNav first takes crowdsourced images and videos as an input to build 3D models (in a form of 3D point clouds) of an indoor space of interest by using SfM techniques. It then creates navigation meshes for path planning based on the obstacle information extracted from the 3D models. Because the crowdsourced photos are taken at arbitrary locations, the generated 3D model may not cover every aspect of a space, and suffer from

1. http://docs.indooratlas.com/app/
4. SfM itself refers to the process of simultaneously computing a 3D point cloud (the structure) representing the pictured scene and the camera pose per each image (the motion). In this work, we use the term 3D model to represent the model that is built with SfM techniques.
uneven points density distribution. This might result in incomplete navigation meshes. To tackle this problem, we propose to track user trajectories using motion sensors and integrate pedestrian paths extracted from the crowdsourced user trajectories into the navigation meshes. We also enable multi-storey building navigation by utilizing the barometric pressure sensor readings when users are using our system to detect connecting paths (e.g., stairs and elevators) and to connect 3D models of different floors. ViNav also extracts POI information from the collected visual data and maps it into the 3D space, thus, users can easily locate POIs.

With SfM-based 3D models, the system provides image-based localization that identifies user’s position and facing direction using the photos taken in situ. The problem with this approach is that the response time increases as the sizes of the 3D models increase, because location is determined by matching 2D features in a query photo against 3D points contained in the 3D models. To enable fast localization, ViNav employs model partitioning based on the density of 3D points, and selects partitions for feature matching based on Wi-Fi fingerprinting. Experimental results demonstrate decreased localization response delays for this partitioning scheme even for large 3D models. By utilizing the image-based localization, the navigation mesh and the identified POI position, our system can provide a navigation path and guide mobile users to their destinations.

In this paper we present the design, implementation and thorough evaluation of ViNav. The contributions of this paper are summarized as follows:

- We address several technical challenges to enable ViNav: (1) we combine the pedestrian paths recognized from crowdsourced user trajectories to complement the crowdsourced 3D models of indoor environments; (2) we fuse barometric pressure data with user trajectories to detect connecting paths between floors; (3) we extract POI information from crowdsourced visual data and map the extracted POI into the 3D space; (4) we speed up the localization process by density-based model partitioning and fingerprint-based partition selection.

- We implement a prototype of ViNav to evaluate its feasibility and performance. Experimental results demonstrate that ViNav works properly in multi-storey buildings and shows good performance. According to our experiments carried out in two different indoor environments, ViNav locates a user within 2 seconds, with a location error of less than 1 meter and a facing direction error of less than 6 degrees.

2 System Overview

We argue that a good indoor navigation system should fulfill the following requirements: (1) it should be able to locate the users precisely and efficiently; (2) it should allow users to search for POI; (3) it should supply the users with flexible navigation routes. To date, when people are talking about indoor navigation, it is usually assumed that fine-grained and up-to-date indoor maps already exist for navigation purpose. For end users, these indoor maps are supposed to contain at least the structural information of the indoor environment, such as the layout of rooms in an office building and the locations of lavatories and emergency exits. Ideally, the indoor maps are expected to contain also information about the places of interests, such as a schedule of lectures in a certain classroom or a sales campaign of a shop. For the purpose of indoor navigation, the indoor maps used for calculating navigation routes are supposed to contain complete information of traversable paths and obstacles. However, such maps are not widely available.

The ViNav indoor navigation system fulfills the aforementioned requirements. The system is based on a client/server architecture. The client serves two purposes: (1) collecting visual data and other sensing data (i.e., accelerometer, gyroscope, barometer readings and Wi-Fi fingerprints), and uploading the collected data to the server for building sensor-enriched 3D models of indoor environment; (2) providing interfaces for POI searching; (3) indicating locations and presenting clear navigation paths to users.

Accordingly, as illustrated in Fig. 1, the server provides two modules: (1) building sensor-enriched 3D models which consists of SfM-based 3D reconstruction, geo-referencing of sensor fingerprints, and recognizing and locating places of interest. (2) 3D-model-based navigation, which calculates positions and navigation routes based on the sensor-enriched 3D models. These functions are explained briefly below, while more details will be given in Sections 3 and 4.

Building Sensor-Enriched 3D Models. First, ViNav builds an initial 3D model from the crowdsourced photos using SfM techniques. The 3D model will be updated constantly by newly collected photos, either from the Internet or from the ViNav clients. Second, ViNav detects user trajectories and adds them to the 3D model. In practice, sensor data including accelerometer and gyroscope readings are used as a base for detecting user trajectories. These trajectories are automatically calibrated by the photos taken on the way. Third, Wi-Fi fingerprints collected along the user trajectories are automatically geo-referenced and added to the 3D model to enable fast localization. Fourth, barometer readings along user traces are also collected to detect connecting paths between multiple floors. Finally, POI information and their corresponding locations are calculated based on the crowdsourced visual data and the generated 3D model.

3D-Model-Based Indoor Navigation. ViNav combines image-based localization with fingerprint-based approach for fast localization. The 3D model is partitioned into multiple smaller models based on the density of 3D points during an offline process. A localization request includes a photo and, if available, a group of Wi-Fi fingerprints collected on site. These Wi-Fi fingerprints are used to match with the geo-referenced Wi-Fi fingerprint database. Once a match is found and coarse locations are calculated, the model partitions that contain these locations are selected for executing image-based localization. After a source and a destination are identified, ViNav calculates navigation routes based on navigation meshes using path finding algorithms. The navigation meshes are first compiled from the obstacle information extracted from the 3D models, and then modified by adding the pedestrian paths extracted from crowdsourced user trajectories. In a multi-storey buildings, the locations of stairs and elevators are identified based on barometer readings collected along the pedestrian paths, to enable seamless navigation between multiple floors.

3 Building Sensor-Enriched 3D Models of Indoor Environments

This section describes how we utilize crowdsourced data in building sensor-enhanced 3D models. The 3D models are later used for implementing indoor navigation.
3.1 Initializing SfM-Based 3D Models

ViNav applies SfM techniques to build sparse 3D point clouds from unordered photos and videos. A typical SfM pipeline consists of 3 steps: feature extraction, feature matching, and bundle adjustment. First, highly distinctive and invariant features are extracted from images using algorithms like Scale-Invariant Feature Transform (SIFT) [21]. After that, the SfM pipeline tries to match features between image pairs. Some photos are automatically dropped if they do not have enough common matches with others. Finally, the matches are input for the bundle adjustment step that jointly produces optimal estimates of camera poses and locations of 3D points.

The output of the pipeline includes a sparse 3D point cloud and a list of estimated camera poses. In detail, a sparse 3D point cloud is a set of data points in a 3D coordinate system. Each 3D point is represented as a tuple consisting of the 3D coordinates (location), a color, and a list showing what images and which features in these images have been used for reconstructing the particular point. For each input image, there is a camera pose included in the output. A camera pose is described with the 3D coordinates of the camera position, a focal length, and a quaternion rotation.

We need to convert the 3D point cloud to the real world scale in order to understand the size of the indoor environment. ViNav utilizes the state-of-the-art tools to register ground control points (GCPs) in two ways. First is to label GCPs on images which contain the same object with a known size by labeling its corner points. The other way is to take photos at 4 different positions (that do not fall to a same 3D plane) with pre-measured real world euclidean coordinates. Either approach can be used to scale the 3D point cloud to a real world scale, based on the GCPs with known coordinates.

We extend our system to support automatic model update as described below. First, if the new photos provide more details of an area already covered by the latest 3D models, the new photos will be registered to the latest models based on feature matching. Second, if the new photos reflect changes in indoor scenes, or cover an area that has not been included in the latest 3D models yet, these photos cannot be directly registered to the latest models. In this case, ViNav will first create 3D models from the new photos and then try to merge them with the existing ones. For the first case (i.e., photos reflecting changes), ViNav will calculate the field of view for each of the new photos, and then apply a ray tracing algorithm to detect which points from the previous models lie inside the field of view. The detected points will then be removed from the latest 3D models. In case there are insufficient common features between models (i.e., covering different areas), these models will remain disconnected.

3.2 3D Model Based Sensor Fusion

After creating the 3D model, our system tries to enrich the model based on sensor fusion, as described below.

3.2.1 Detecting User Trajectories Using Motion Sensors

The ViNav client collects accelerometer and gyroscope readings from smartphones as users traverse in indoor environments. Based on these readings, the ViNav server calculates user trajectories based on dead reckoning [18], and uses photos taken on the way for trajectory calibration. The process of automatically detecting and calibrating user trajectories is implemented through two steps.

First, the ViNav server obtains the camera position of each photo taken on the way based on the image-based localization method, which will be described in Section 4.1. The camera positions are set to be the calibration points.

Second, between every two successive calibration points, the ViNav server computes a trace based on step count, heading offset, and stride length. Steps are counted based on accelerometer data [18], while the heading offset between two steps is calculated from gyroscope readings. The stride length is first set to its default value, e.g., 0.7 meter, and then...
updated according to the accumulated walking distance and step count. The position after each step is reckoned based on the previous position and the accumulated heading offset. The calculated trace will be calibrated based on the locations of the two calibration points [35].

Given a set of user trajectories, ViNav discovers pedestrian paths by selecting the paths that have been covered by at least a certain number of trajectories. By default we set the number to 2. Each pedestrian path is represented as a sequence of points. These paths are integrated into the 3D point clouds while compiling navigation meshes. More details about compiling navigation meshes will be presented in Section 4.2.

3.2.2 Geo-Referencing Sensor Fingerprints

The ViNav client collects barometer readings and Wi-Fi fingerprints while capturing accelerometer and gyroscope data along user trajectories. Based on time stamps, these fingerprints can be easily geo-referenced and attached to the 3D point clouds after the user trajectories have been automatically calibrated. The system monitors the changes in barometer readings, and estimates the locations of stairs and elevators based on the changes. Meanwhile, the geo-referenced Wi-Fi fingerprints are stored in a database which will be used for fingerprint-based 3D model partition selection, as described in Section 4.1.

3.3 Locating Stairs and Elevators in Multi-Storey Buildings

In multi-storey buildings, different floors are connected with stairs, escalators, and/or elevators. However, they are very challenging to be reconstructed with SfM techniques, due to featureless surfaces or repetitive and non-distinctive patterns. As a result, it is very challenging to obtain a single SfM-based 3D model that covers all the floors. Instead, we build one model for each floor and connect them together using connecting paths. In our system, we rely on barometer readings collected along user trajectories to identify the connecting paths between floors and to further provide cross-floor navigation services.

Fig. 2a shows examples of barometric pressure readings which are collected while taking an elevator or walking downstairs/upstairs in a three-storey building. We can always observe a significant difference of barometric pressure when going up or down. The value decreases when we move to an upper floor, and vice versa. Although the exact values corresponding to each floor varies with the time-of-day [25], by monitoring the changes over time, we can reliably detect the movement between floors.

From calibrated user trajectories we can obtain user’s position at time t. Whenever a significant change in the barometer readings is detected, we record the position of a user at that moment (i.e., \((x_1, y_1)\) in model A) and consider it as an endpoint of a connecting path between floors. When the barometer readings become stable in a pre-defined time window, we assume that the user has left the stairs or an elevator and we record a position (i.e., \((x_2, y_2)\) in model B) as the other endpoint of the connecting path. We set the time window of 2 seconds, which, according to our experiments, was most suitable to distinguish between walking on a staircase and walking in a corridor. Once we obtain enough connecting path endpoints (at least 3 are required for the clustering algorithm to work), we utilize DBSCAN [8] clustering algorithm to obtain mean locations of the endpoints and to filter out outlier measurements. The connecting paths are added to the 3D models, as shown in Fig. 2b.

3.4 Adding Points of Interest Information

A comprehensive indoor map is supposed to include also POI information, e.g., shop names, room numbers and even discount information in stores. Usually such information is manually tagged to the corresponding locations on the map. Since our system uses photos as input and these photos include useful information that describes the places, we propose to apply Optical Character Recognition (OCR) techniques [32], [37] to extract texts from these images and add them to the map.

The recognized texts are associated to corresponding locations in a 3D space as described in Algorithm 1. The first step is to apply an OCR tool to images to recognize texts. Due to the complex backgrounds of the images, some characters may not be correctly recognized. To filter out non-existing and incorrectly recognised words, we select only the words close to the lexicon entries [34]. We utilized English, Swedish and Finnish lexicons, as experiments were carried out in Finland. If a word is recognized from a photo, we obtain the pixel coordinates of the word in the photo, as shown by a dashed rectangle in Fig. 3.

On the other hand, after image feature extraction, a set of features can be extracted from the photo [21] and we can obtain a set of pixel coordinates of these features. The green dots in Fig. 3 describe these feature points. By comparing the coordinates of each feature point with the ones of the recognized word (in practice, we use the center of the detected sign), the nearest feature point to the word can be found. The yellow circle in Fig. 3 shows an example of a selected feature point. After that, the recognized word is associated with the selected feature point. As explained
in Section 3.1, each 3D point in the SfM model contains a relationship between the point and a list of images that see this point. Especially, the 2D pixel coordinates that represent the point in these images are also stored in the SfM model. Fig. 4 shows an example of the relationship between extracted features and their corresponding 3D points in the point cloud. As a result, the recognized word can be associated with a 3D position based on the relationship.

Algorithm 1. Pseudo Code of Adding POI Information

Input: Crowdsourced photos: \( P \), 3D model: \( M \)

Output: Location of place of interests: \( L \)

1: for \( P_i \in P \)
2: // Apply OCR to each photo
3: if Recognition\((P_i)\) == TRUE
4: // Get pixel coordinates of the detected text
5: \( \text{TextPixelCoord} = \text{GetCoord}(P_i) \)
6: // Get pixel coordinates of all the feature points extracted from \( P \)
7: \( \text{FeaturePixelCoordSet} = \text{GetFeatureCoord}(P_i) \)
8: // Find the feature point which is the closest to \( \text{TextPixelCoord} \)
9: point = Find\((\text{TextPixelCoord}, \text{FeaturePixelCoordSet})\)
10: // Locate the 3D coordinate of \( \text{FeaturePoint} \) from the 3D model
11: \( L = \text{Get3DCoord(point, } M) \)

After collecting a list of POI and their 3D positions in an indoor environment, we maintain them in a database. When a user wants to find a specific POI, we will return the position of the POI and show it on the map.

4 3D-MODEL-BASED INDOOR NAVIGATION

After building the sensor-enriched 3D models as described in Section 3.1, this section continues to discuss how indoor navigation can be conducted. We first describe methods of indoor localization in Section 4.1; then we present how navigation meshes are compiled in Section 4.2; in Section 4.3 we explain how we implement cross-floor navigation.

4.1 Indoor Localization

Given a query photo, the process of image-based localization is expected to return the location where the photo is taken and the direction in which the camera is facing. A straightforward method is to register the query image into the 3D point clouds based on feature matching. In theory, the feature matching is conducted between the query image and each of the images used for building the point clouds. If enough matches are found, the camera pose will be calculated and returned.

The size of a 3D point cloud, in terms of the number of 3D points, increases with the number of images involved. It results in consuming more resources not only to load the point cloud into memory but also to iterate over its 3D points. To enable parallel processing, ViNav operates on a point cloud that is divided into smaller sub-clouds, and executes feature matching operations only on certain selected partitions.

Algorithm 2. Pseudo Code of Indoor Localization Algorithm

Input: User’s query photo: \( U_{\text{photo}} \), Wi-Fi Fingerprints: \( U_{\text{Wi-Fi}} \)

Output: User’s location: \( U_{\text{location}} \), User’s facing direction: \( U_{\text{dir}} \)

1: // Apply k-NN to get a set of coarse locations based on Wi-Fi fingerprints
2: // \( \text{DB}_{\text{Wi-Fi}} \): Database of geo-referenced Wi-Fi fingerprints
3: \( L_{\text{coarse}} = \text{k-NN}(U_{\text{Wi-Fi}}, \text{DB}_{\text{Wi-Fi}}) \)
4: // Select model partitions based on coarse locations
5: \( \text{Partitions} = \text{SelectPartition}(L_{\text{coarse}}) \)
6: // Identify user’s location and facing direction based on feature matching between \( U_{\text{photo}} \) and each partition in \( \text{Partitions} \)
7: \([U_{\text{location}}, U_{\text{dir}}] = \text{StartSfMInParallel}(U_{\text{photo}}, \text{Partitions}) \)

ViNav partitions a 3D point cloud into smaller ones following two rules. First, each partition includes 3D points that are created based on the image features extracted from a limited number of photos. Second, the area covered by each partition should not be too small. In practice, we define the minimum length and width of the area covered by each partition. An example is visualized in Fig. 5.

As a first step of the localization process, ViNav selects partitions for feature matching as described in Algorithm 2. It applies K-Nearest Neighbour (k-NN) algorithm to acquire a set of coarse locations based on Wi-Fi fingerprinting. After that, ViNav selects all the partitions that cover the coarse locations. ViNav then executes SfM localization algorithm on each partition in parallel. The localization algorithm matches features extracted from the query photo to features in a point cloud and estimates the camera pose. Different SfM tools can be used to fulfill the purpose. In this work, we selected two state-of-the-art SfM tools, VisualSFM\(^8\) and openMVG\(^9\) for evaluation. The details are described in Section 5.5.

It is possible that several positions and rotations are returned, since several partitions were selected during the previous fingerprinting step and more than two of them contain enough common features for registering the query

---

8. http://ccwu.me/vsfm/
9. https://github.com/openMVG/openMVG
After the SfM model is converted to a real world scale, the points between two floors is added. A connecting path starts groups the according to the means of changing G on one floor located on another points and add them as or as weights of changing the floors. We set to all exit points in model A (line 2) is built, we can easily find an optimal path ViNav (line 11) and send it to the user. Our algorithm supports s may correspond to multiple connecting paths and differ- first removes the points which are lower than d and z G applies the algorithm proposed by Duck tal. Given a 3D point cloud of indoor environment, some points may represent obstacles, e.g., tables, chairs and walls. ViNav compiles a navigation mesh from a point cloud by extracting obstacles from these points, treating the rest of the blank areas as traversable areas. It is implemented through the following four steps.

1) ViNav first removes the points which are lower than a floor level or higher than a ceiling level. In our experiment, we set a threshold to 10 cm for the floor and 2 m for the ceiling.10 The thresholds are used to remove noise that may occur from floor objects, such as carpets and doorsteps, or from ceiling objects such as lamps. The rest of 3D points are projected onto the ground plane to acquire a 2D point set.

2) Pedestrian paths are extracted from any available trajectories. A pedestrian path includes a sequence of points. Assuming that there are no obstacles along the path, all the points along the path are removed from the 2D point set obtained in the previous step. ViNav removes the neighbouring points that are close to the central line of the path. In practice, we set the distance threshold to be 25 cm, thus making a clear path of 50 cm width. We chose 50 cm threshold, as according to our experiments, this width is still an ergonomic one for a single person to pass through. The chosen value also agrees with values for non egress passage width recommended by Lehto et al. [17].

3) ViNav applies the algorithm proposed by Duck ham [7] to generate non-convex polygons out of the 2D points derived in the previous step. The algorithm wraps a set of points with a polygon, specifying which points belong to the outline and which lie inside the polygon. Accordingly, we divide all the 2D points into groups, with each group corresponding to one obstacle. In practice, ViNav groups the points if the distance between any two points is less than the previously defined threshold, i.e., 50 cm.

4) After extracting the polygons, ViNav extrudes all 2D polygons towards the z-axis to form solid meshes. The generated meshes are saved in a Wavefront Object (.obj) file.

4.3 Multi-Floor Navigation

Based on the generated navigation meshes, our system implements an A* like path finding algorithm, using Recast-Detour library11 to calculate navigation paths. To support cross-floor navigation, we modify the A* path finding algorithm by taking into account the connecting paths, such as stairs and elevators. The locations of the connecting paths are identified as described in Section 3.3. The multi-floor routing algorithm consists of off-line processing stage, when a new floor or a connecting path is added, and an on-line stage when a user issues a multi-floor navigation request.

Off-Line Preprocessing and Building a Navigation Graph. We execute the offline step when a floor or a connecting path CP between two floors is added. A connecting path starts from a floor exit point Xf and ends at an entrance point of another floor Ef (blue solid lines in Fig. 6). In other words, it defines a floor change between those particular points. Any Xf may correspond to multiple connecting paths and different entry points Ef, as e.g., if one takes an elevator, it can stop at multiple floors. During the off-line stage we build a multi-floor navigation graph G that is later used for navigation. We add every floor entrance Ef and exit Xf points as nodes to graph G. Since a user’s route may include traversing multiple floors, we precompute navigation paths NP between every pair of Ef and Xf points and add them as edges to G. Finally, we add the previously detected CP to G to connect all floors to one navigation graph (blue lines in Fig. 6). Weights of the graph edges are calculated either as lengths of NP or as weights of changing the floors. We set different weights of CP according to the means of changing a floor (whether by an elevator or on foot). In this way, a user can give a preference to elevators over stairs or vice versa by simply changing a setting inside a client application. After G is built, we can easily find an optimal path between any two locations within a building.

On-Line Processing of a Multi-Floor Navigation Request. When a user travels from a position PA(xa, ya) on one floor (model A), to a destination PB(xb, yb) located on another floor (model B), a navigation path is calculated according to Algorithm 3. The algorithm first utilizes A* algorithm to find paths from PA to all exit points in model A (line 2) and from all entry points in model B (line 4) to PB. The paths are then added to G (lines 5-10). Since we now have a weighted directional navigation graph that includes the user’s location and the destination, we apply Dijkstra’s algorithm to find the shortest path between nodes PA and PB (line 11) and send it to the user. Our algorithm supports navigating between multiple floors and arbitrary floor

---

10. After the SfM model is converted to a real world scale, the points that represent the floor should have the height of less than 10 cm, while the points representing the ceiling should have the height of at least 2 m.

changing points even in cases when one has to take several elevators, escalators or stairs to reach a desired destination.


\begin{verbatim}
Input: Current location: \( P_A(x_a, y_a) \); Destination: \( P_B(x_d, y_d) \); Navigation meshes compiled from model A and model B, respectively: \( mesh_A, mesh_B \); Floor entry an exit points database \( DB \); navigation graph: \( G \).
Output: Navigation path: \( path \)

1: \( // \) Get all exit points in the starting floor
2: \( exitPoints_A = getFloorExitPoints(mesh_A, DB) \)
3: \( // \) Get all entry points in the destination floor
4: \( entryPoints_B = getFloorEntryPoint(mesh_B, DB) \)
5: \( // \) Add edges from the start position to the floor exits
6: \( for \ P_e \in exitPoints_A \ do \)
7: \( addEdge(G, edge(P_e, P_A)) \)
8: \( // \) Add edges from the floor entries to the destination
9: \( for \ P_i \in entryPoints_B \ do \)
10: \( addEdge(G, edge(P_i, P_B)) \)
11: \( // \) Run Dijkstra’s algorithm to find the shortest path through \( G \)
12: \( path = \text{FindShortestPathWithDijkstra}(G, P_A, P_B) \)
\end{verbatim}

5 EVALUATION

We implemented a prototype of ViNav to evaluate the feasibility of building a smartphone-based indoor navigation system using crowdsourced data. We have carried out field studies in different indoor environments including an office building and a supermarket. According to the field studies, the functionalities of ViNav have proved to work properly. We will focus on the performance evaluation of ViNav, and will introduce in this section the methodology and the experimental results, including a short summary of the feasibility study.

5.1 Methodology

ViNav can be used for locating users, searching for places of interests, calculating navigation routes, and providing AR navigation guidance. Performance of ViNav depends on the performance of the localization, the accuracy of recognized Pols, and the accuracy of the navigation meshes used for path planning. Because the navigation meshes are generated from the sensor-enriched 3D models (cf. Section 3), the accuracy of the navigation meshes is determined by the quality of the 3D models in use. Similarly, the accuracy of the navigation depends on the accuracy of the estimates of user’s position and facing direction. Therefore, for performance evaluation of ViNav, we measure the following metrics: (1) the accuracy of the indoor layout extracted from the 3D models, (2) the accuracy of detecting stairs and elevators in multi-floor buildings, (3) the performance of POI detection, and (4) the performance of the 3D-model-based indoor localization.

5.2 Accuracy of Navigation Mesh

We first evaluate the accuracy of the indoor layout extracted from 3D models. We collected data in two different indoor environments. The first one is an office building, where Department of Computer Science at Aalto University locates (we name it as CS building thereafter). It consists of 3 floors. During weekdays, more than 500 students and staffs visit the CS building on daily basis. The ground floor is approximately 1,100 m², covering a long corridor, a library, and a cafeteria. The indoor scenes of the cafeteria change frequently due to promotions and decorations. A construction work of a new food store was started during the field study. The second building is a big supermarket in Helsinki. Its size is around 1,600 m². There are more than 30,000 products selling in the supermarket.

5.2.1 CS Building

We recruited 5 volunteers to collect photos on the ground floor of CS building. The volunteers can choose by themselves where to take photos. For each location they chose, they were requested to take 5 to 10 photos with the camera facing different directions. Meanwhile, the ViNav mobile app was configured to record Wi-Fi fingerprints and the readings of accelerometer and gyroscope while the volunteers were walking in the CS building. The data collection started immediately after the first photo was taken, and can be stopped manually. Over a period of 3 weeks, the volunteers collected in total 2,197 photos and 4,239 groups of Wi-Fi fingerprints along 119 walking traces.

We built an initial 3D model of the CS building using the data collected by volunteers. Following the SFM pipeline described in Section 3.1, the initial 3D model (as shown in Fig. 7b), which includes 253,839 3D points, was successfully built from 1,968 (out of 2,197) photos. Based on the generated 3D models and the user trajectories, ViNav compiled a navigation mesh as described in Section 4.2. The navigation mesh is visualized in Fig. 8a. For comparison, a navigation mesh generated from the initial 3D model only is also shown in Fig. 8a. We can see that the pedestrian paths, especially the one along the corridor, are better reconstructed when user trajectories are used for compiling navigation meshes.

Localization and navigation services were provided to users after the initial model was built. The volunteers were invited to test the indoor navigation services. During the trial period, another 355 photos and 1,181 Wi-Fi fingerprints along 31 walking traces were collected. Accordingly, the 3D model will be updated based on new images. As illustrated in Figs. 7b and 7c, the 3D models have been successfully updated to reflect a significant change of indoor layout caused by the ongoing construction work of a Subway store.

Compared with the official floor plan released in 2008 (c.f. Fig. 7a), the 3D models generated by ViNav well reconstruct the outline of the CS building. To evaluate the accuracy of the reconstructed outline, we calculated the euclidean distances of walls surrounding the CS building based on the 3D models, and measured the ground truth using a laser rangefinder. After that, we calculated the distance error as defined in [28]. Specifically, the distance error for a correctly detected segment is defined as the average distance between the detected segment and its associated ground truth segment.

The overall distance error, denoted by \( E_D \), is calculated below.

\[
E_D = \frac{\sum_{i=1}^{N} |\text{length}(s_i)E_D(s_i)|}{\sum_{i=1}^{N} \text{length}(s_i)},
\]

where \( N \) is a number of correctly detected segments, and \( E_D(s_i) \) is the distance error for a correctly detected segment \( s_i \). According to the results, ViNav is able to reconstruct the
Our experiment results have shown that the initial data set, which includes 2,197 photos collected from locations that are nearly evenly distributed in the area, is enough for building a 3D model of the CS building with reasonably good quality. Concerning the complexity of indoor structure and the diversity of indoor scenes, the number of photos needed for building a good-quality 3D model varies with indoor environments. Furthermore, the quality of 3D models depends also on the manner of data collection, including where to take photos and what to be captured by the photos.

To analyze the impact of data collection, we combined the two photo sets collected by volunteers (we name it as data set C thereafter), and sampled them in 5 different ways. We compared the 3D models built from each subset accordingly. The 3D models created from data set C was used as the baseline. First, we grouped the photos by location and facing direction, and then evenly selected 50 percent of the photos. The set of sampled photos is referred to as Sample 1. Similarly, we evenly sampled 25 percent of the photos from data set C to form Sample 2. Compared with the data set C, the photos in Sample 1 and 2 were taken with lower density but still trying to cover the whole space. To emulate the data collection in a more random manner, we also built Sample 3 and Sample 4 by randomly selecting 50 and 25 percent of photos from data set C, respectively, regardless of location or facing direction. For comparison, we manually selected 20 percent of the photos from data set C by removing photos that captured similar scenes. The selected photos form Sample 5.

The number of 3D points generated from the 5 samples are listed in Fig. 9. From the figure we can observe that as less photos are used for 3D modelling, the output 3D models are more prone to be fragmented, which is clearly illustrated in cases where Samples 2, 3, and 4 are used as input. On the other hand, the 3D models built from Sample 1 remain almost as intact as the ones from data set C, from fragmentation perspective. Compared with the largest 3D models generated from Samples 3 and 4, the biggest 3D model created from Sample 5 contains more 3D points. This is because the scenes captured by different photos in Sample 5 are expected to have less overlaps. The impact on the density of 3D points will be discussed in Section 5.5.3.

Crowdsourcing helps collect data with a low cost. However, it brings uncertainty in the quality of the modelling result. Depending on the size and the complexity of indoor environments, it is hard to tell how many photos are enough. In principle, in order to use as few photos as possible for building SfM-based 3D models while maintaining the performance of localization, the input photos are expected to have as little overlap as possible while having sufficient common features to be matched.

5.2.2 Supermarket

We recruited another 8 volunteers to collect data from the supermarket. Different from the data collection in the CS building, the volunteers were asked to collect videos. The idea of collecting video instead of images is to check the feasibility of bootstrapping an initial model in a more efficient way. The volunteers collected 2 hours video in total in one morning. We extracted frames from these videos by comparing blurriness of frames and choosing the most clear one within a time window. 9,594 image frames were extracted...
in total. We built a model with 1,806,772 points, with 9,497 out of the 9,594 frames registered into the model.

Fig. 10b shows a top-down view of the generated point cloud. By comparing it with the official floor plan as shown in Fig. 10a, it is interesting to see that the structure of the supermarket is well reconstructed. Specifically, the overall distance error of the outline is 0.68 m based on Eq. (1). Fig. 10c shows an overlap between the point cloud and the official floor plan. There are some shelves which are different from the original floor plan. We made a site survey in the supermarket and found that our point cloud reflects the ground truth better. This is because the floor plan was created several years ago. Some of the shelves were changed during these years but the floor plan was not updated accordingly. The green dotted circles in Fig. 10c show an example, where a newly added shelf is not on the original floor plan but reconstructed correctly in the point cloud.

5.3 Locating Stairs and Elevators

In this section we will evaluate how accurately and how well we can build a multi-floor navigation graph with only crowdsourced input. We carried out the evaluation in a 3 floor building. We have built each of the floors as a separate model with a single floor navigation enabled. We then walked around the building and collected 50 cross-floor traces while taking stairs and elevator. The algorithm of trace collection is described in Section 3.3.

Table 1 shows all the discovered connecting paths, i.e., stairs and elevators. The building has an elevator between the 3 floors and several staircases in between floors. We have measured the distance error between each detected floor changing point and the actual point. For staircases, we considered the middle outer point on a first staircase step as the actual point, while in case of an elevator, a middle outer point of the elevator door. Our system detected all floor changing means inside the building. We observe larger errors only in cases of Floor 1 elevator and the staircase from Floor 3 to Floor 2, since at those places there are many featureless surfaces and accurate position was obtained only after a user walked several steps away from an elevator/staircase. The average distance error was 2.13 meters (1.31 std). The result shows that ViNav can reliably identify floor changing points which can be further used to enable the multi floor navigation in the building.

5.4 Accuracy of Mapping Points of Interest

We applied one state-of-the-art OCR tool, Google cloud vision,\(^\text{12}\) to recognize texts from the images used for 3D modelling. As a result, 5,247 words were extracted from the images of the CS building. After applying the predefined lexicon entries, 17 POIs were found, including room numbers and exit information. Comparing to CS building, we extracted 5,316 words from the supermarket. 135 POIs were found after the lexical filtering.\(^\text{13}\) Fig. 11 shows that some sale signs were detected in the supermarket because the text in the sign was recognized. The positions of these signs are calculated as shown in Fig. 11a.

As explained in Section 3.4, the location of a POI is set as the coordinates of the feature point which is the closest one to the center of the recognized word. As a result, the accuracy of POI mapping depends on the distance from the center point to the closest feature point. Because the density of the feature points surrounding the text is high in most cases, the error of POI mapping is reasonably low. Compared with manually-obtained ground truth, the average distance error of POI mapping is 0.26 meters with standard deviation of 0.16 in the CS building, and is 0.29 meters with standard deviation of 0.17 in the supermarket.

<table>
<thead>
<tr>
<th>Floor 1</th>
<th>No. of measurements</th>
<th>Position error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staircase to 2nd floor</td>
<td>5</td>
<td>1.3622</td>
</tr>
<tr>
<td>Staircase to 2nd floor</td>
<td>4</td>
<td>2.5137</td>
</tr>
<tr>
<td>Elevator</td>
<td>15</td>
<td>4.2298</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floor 2</th>
<th>No. of measurements</th>
<th>Position error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staircase to 1st floor</td>
<td>6</td>
<td>1.8135</td>
</tr>
<tr>
<td>Staircase to 3rd floor</td>
<td>13</td>
<td>2.6886</td>
</tr>
<tr>
<td>Staircase to 1st floor</td>
<td>18</td>
<td>1.3317</td>
</tr>
<tr>
<td>Staircase to 3rd floor</td>
<td>8</td>
<td>0.8875</td>
</tr>
<tr>
<td>Elevator</td>
<td>3</td>
<td>0.5212</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floor 3</th>
<th>No. of measurements</th>
<th>Position error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staircase to 2nd floor</td>
<td>12</td>
<td>2.1184</td>
</tr>
<tr>
<td>Staircase to 2nd floor</td>
<td>6</td>
<td>4.6336</td>
</tr>
<tr>
<td>Elevator</td>
<td>8</td>
<td>1.3539</td>
</tr>
</tbody>
</table>

\(^\text{12}\) https://cloud.google.com/vision

\(^\text{13}\) The supermarket provided us a list of products selling in it.
The length of one grid is 0.5 m, we chose a radius of 0.25 m.
measures how much time it takes for the server to return the result. The response delay of indoor localization depends on the computational complexity of the localization algorithm under test, as well as the computing power of the computing infrastructure in use.

ViNav combines Wi-Fi fingerprinting with image-based localization to achieve fast while accurate indoor localization. To provide an objective comparison with the state-of-the-art, we provided a reference implementation of the image-based localization (abbreviated as Image) in the scenario of CS building. The reference implementation of Image applies Breadth-first Search (BFS) to select candidates for feature matching, and tries to search through the whole 3D point cloud instead of selected partitions to get the best possible result. Both ViNav and the reference implementation of Image were tested with the same 3D models. In addition, to provide a comparison with the Wi-Fi fingerprinting based approaches, we utilized the Wi-Fi fingerprints collected in the test data set to implement a standalone Wi-Fi fingerprinting solution (abbreviated as Wi-Fi). The experimental results are presented as follows.

5.5.1 Hit Rate
If a valid location can be returned by the localization algorithm, we consider the input as a matching input for the algorithm; otherwise, it is considered as a non-matching input. If at least one image collected from the same measurement point was considered as a matching input, we count the corresponding measurement point as a locatable point. The hit rate is calculated as the number of locatable points divided by the total number of measurements points. It provides a good hint on how easy a user can be located at different positions of an indoor environment.

In Table 2 we compare the hit rate among ViNav, Wi-Fi, and Image in the CS building. In general, the hit rate is reasonably high in all the three cases. Compared with Wi-Fi, the two image-based schemes (i.e., ViNav and Image) fail to locate around 5 percent of the measurement points. This is because of the changes of indoor scenes occurred after the initial 3D model was built. Compared with the corridor, the layout of the cafeteria and the library changed more frequently, which explains the relatively low hit rates in such environments.
areas. Furthermore, Image also achieves slightly higher hit rate than ViNav. This is because ViNav only selects a subset of model partitions for feature matching in order to lower the response delay. The partitions filtered out by Wi-Fi fingerprints in ViNav may contain feature matches with the query photo. The hit rate in the supermarket for our system is higher than in the CS building. All the 71 measurement points are locatable.

Photos taken at the same position may get different localization results, due to different facing directions. We further check the number of matching inputs at each measurement point, which discloses how easy a user can be located at a specific position. As described in Fig. 14, 69 percent of the measurement points in the supermarket have all the test images located. If the query photo includes mostly the texture-less scenes, e.g., white walls, it is very likely that distinguishable features are lacking and it will become a non-matching input. The supermarket is an open space containing many shelves that do not block the view. As a result, most of the test images have enough common scenes for feature matching.

In the CS building, we took 12 photos at each measurement point. It is inevitable that a certain number of test photos are facing to texture-less scenes. As shown in Fig. 15, we can clearly see that the experiment in the library has higher percentage of measurement points with more matching inputs than in the corridor. This is because library is an open space while corridor only has distinguishable scenes when facing forward or backward. In practice, ViNav will suggest the user to take photos of distinguishable scenes. If the first photo is a non-matching input, ViNav will request the user to turn to a different direction to take a new photo.

### 5.5.2 Accuracy

As explained previously, we collected ground truth of locations and facing directions using self-developed toolkit in the CS building, and evaluated the accuracy of indoor localization based on the ground truth and the estimated location and facing direction (if available). As shown in Figs. 16a and 16b, the accuracy of ViNav is very close to that of Image. In both cases, for at least 90 percent of the valid inputs, the location error is less than 2 meters, while the direction error is less than 6 degrees. Similar to the hit rate (cf. Table 2), the accuracy of Image is slightly different from that of ViNav, due to the pre-filtering of model partitions used in ViNav.

The result of Wi-Fi does not include an estimate of the facing direction. The location error of Wi-Fi is less than 5 meters in approximately 70 percent of the test cases and less than 10 meters in 90 percent of cases. Obviously, the accuracy of Wi-Fi is relatively low, compared with the image-based approaches. On the other hand, because the hit rate of Wi-Fi is higher, it is reasonable to fall back to the coarse locations provided by Wi-Fi fingerprinting, when the system fails to obtain a more accurate location based on photos.

The location and facing direction errors of our system in the supermarket are described in Fig. 17. The accuracy is better than the ones in the CS building: 97 percent of the test cases have location errors of less than 1 m and facing direction errors of less than 6 degrees. Although the products on shelves are changing from time to time, there are still enough stable features for matching. For example, posters hanging on the roof and brand advertisements on shelves are very good hints for localization.

### 5.5.3 Robustness

To investigate the impact of the input for 3D modelling on the accuracy of localization, we chose the 3D models built from data set C as a reference, and built another two models from Samples 1 and 5, respectively, for comparison. We chose the library as the testing environment, and selected all the 840 test photos.

As illustrated in Fig. 18, the system achieves higher accuracy with the reference model than the ones built from Samples 1 or 5.15 Recall that more test data become matching inputs with a denser point cloud, as shown in Fig. 15. Hence, we can conclude that the positioning accuracy increases with the density of the point cloud. Nevertheless, the density does not increase if less photos are used for building the point cloud.

15. The indoor navigation system performs better than average in the library, because the scenes in the library are more distinguishable than the corridor for example. That is why the results in Fig. 18 are better than the ones shown in Fig. 16a.
Objects like whiteboards and chairs may appear (disappear) or move around frequently in the building after the 3D models are built. Additionally, people may accidentally show up in front of the camera when photos are taken for localization. To evaluate how the localization accuracy is affected, we conducted a set of tests where we brought a whiteboard to the library and moved it towards our camera slowly, trying to block the whole camera view gradually. We took 18 photos from the same location while the whiteboard was moving as described in Figs. 19a, 19b, and 19c. After that, we used ViNav to locate the camera with each of these photos. With 17 out of the 18 photos, ViNav successfully identified the location with comparable accuracy, as shown in Figs. 19d and 19e. ViNav failed to locate the camera only when the photo captured nothing more than the whiteboard. The localization accuracy remained at the same level as far as the location was successfully identified.

5.5.4 Response Delay

We deployed ViNav on a server equipped with an Intel Xeon processor E5-2650 (8-core, 2.6 GHz), 64 GB RAM, and a Tesla K20C GPU. We measured the response delay using all the test data, and compared the results between ViNav, Wi-Fi, and Image. Note that the testing data set includes both matching and non-matching inputs from the perspective of localization.

Fig. 20a shows that the response delay of Wi-Fi is less than 0.6 s for 87 percent of the input. For Image, as presented in Fig. 20b, it takes more than 40 s to generate a response. The delay comes from VisualSFM model preloading and image matching which increases with model size. ViNav supports 3D model partitioning and fingerprint-based partition selection. As shown in Fig. 20c, the response delay, including loading time, is decreased to 4 s for 73 percent of the tests when each model partition contains features extracted from less than 100 photos. The response delay excluding the loading time is on average 3.85 s with standard deviation of 1.5 s. In the supermarket, as shown in Fig. 20d, more than 85 percent of the matching inputs are finished within 1.5 seconds and more than 95 percent of them are finished within 2 seconds. The response delay is shorter, as OpenMVG supports pre-loading 3D models at the beginning, thus the loading time is saved.

5.5.5 Scalability

We evaluated performance and scalability of ViNav in the supermarket case. We started by simulating a single system user and kept increasing the number of simultaneous users. In order to take network latency into account we deployed ViNav on an amazon public cloud p2.xlarge instance. Fig. 21a shows system response times for different amounts of simultaneous system users. The selected p2.xlarge instance can support up to 7 users without exceeding a mean response delay of 5 seconds. We also conducted an experiment where we sent localization requests from a client within a single network hop to simulate an edge computing scenario. As Fig. 21b shows, such setup can support almost twice as many users, meaning that network latency has a notable impact on system performance. We have measured request payload sizes of 400 arbitrary localization requests. The mean request size was 129.62 KB (s = 60.75), of which the captured image accounts for more than 95 percent. Therefore, a server with a typical 100 Mbps connection could support up to 98 simultaneous ViNav users.

5.6 Summary

From the experiments and analysis presented above, we can reach the conclusion that it is feasible to build a well-performing 3D-model-based indoor navigation system, using only the photos and sensor data collected from smartphones. The quality of 3D models, nevertheless, is affected.
by multiple factors, including the number of photos used for generating the 3D models, and the coverage and the facing direction of the input photos. Although the coverage of crowdsourced photos may be limited to certain areas of interest, with more and more photos becoming available, there is an increasing chance of producing high-quality 3D models for the purpose of indoor navigation. Furthermore, the accuracy and response delay of ViNav are decent and acceptable compared to the state-of-the-art.

6 RELATED WORK

3D-Model-Based Indoor Mapping. 3D modelling has been widely used for indoor mapping. Conventionally, 3D models for indoor mapping are generated from the data captured by laser scanners and/or depth cameras through war-driving. For example, the system presented in [28] obtained 3D point cloud of indoor environment from laser scanners positioned in multiple locations, and extracted 2D floor plan from the 3D point cloud. KinectFusion [26] implemented real-time 3D surface reconstruction of room-sized scenes based on the depth data streamed from a moving Kinect sensor. The recently developed Simultaneous Localization and Mapping (SLAM) based indoor mapping tools, such as Google Cartographer and Xsens Scannect [4], are equipped with inertial measurement units in addition to laser scanners and depth cameras. The Cartographer-like backpack systems can automatically create floor plans when the backpack-wearers walk inside the buildings.

Differently from SLAM, SfM techniques enable 3D modelling of different environments using unordered 2D photos. These photos can be taken by commodity devices such as smartphones. Based on SfM, Agarwal et al. [1] constructed 3D models of Rome city from 150 K photos found from Internet photo sharing sites. Furukawa et al. [10] used SfM and multi-view stereo for reconstructing and visualizing an entire house interior. Martin-Brualla et al. [23] considered the scenario of producing disconnected 3D point clouds and proposed to connect these 3D pieces with the help of a given floor plan. Gao et al. [12] proposed a WiFi and accelerometer based approach for detecting inter-floor connecting paths between the models. Jigsaw [13] builds SfM-based 3D models for landmarks. After that, the local coordinates of these landmarks are converted to a global coordinate system with the help of motion sensor data. ViNav also adopts SfM for 3D modelling. Our work further complements SfM modeling by detecting user trajectories, geo-referencing Wi-Fi fingerprints and locating stairs and elevators. As a result, the sensor-enriched 3D models are used to accelerate image-based localization, detect pedestrian paths, and find connection areas between floors.

Indoor Localization and Navigation. Previous work has proposed to utilize fingerprints of Wi-Fi [2], magnetic field [5] or Bluetooth [3] for locating users in indoor environments. Fingerprinting-based approaches require a training data set comprising of measurements at known locations in a space of interest. Since labelling fingerprints is labour-intensive, Zee [29] proposed automatic inferencing of location, based on the combination of inertial sensor information and the constraints imposed by the map. Similarly, Nguyen et al. [27] applied dead reckoning approach to track users in indoor environment. Due to the error accumulation of the dead reckoning approach, Nguyen et al. utilized active learning for automatically identifying strategically important locations which should be labelled manually. Travi-Navi [38] employed traces of fingerprints recorded from entrance POIs to be used for leader-follower navigation. Rather than manually labelling Wi-Fi fingerprints, ViNav utilizes arbitrary photos taken along user trajectories for automatic calibration, utilizing the feature of image-based localization. Furthermore, ViNav provides navigation to arbitrary destinations with no requirement of following a pre-recorded trace.

Image-based localization systems allow users to locate themselves by simply taking photos from where they are. State-of-the-art systems such as Travi-Navi [38] employs image histogram matching, Liu et al. [20] utilized deep-learning approach for matching and tracking, while Huang et al. [15] proposed image feature matching for panoramic images to obtain user’s position. However, this process of image feature matching is slow due to heavy computation. To accelerate this process, Lu et al. [22] proposed to use the visual words and the approximate nearest neighbour methods. Similarly, Sattler et al. [31] proposed a framework for efficient 2D-to-3D matching based on visual vocabulary quantization and a prioritized correspondence search. Sextant [11] utilizes physical features (i.e., logos or paintings) as reference objects to measure user positions. The challenges of designing such a POI-based localization system include identifying distinguishable POIs, obtaining accurate positions of POIs, and etc. ViNav utilizes features extracted from images as references to locate users. Distinguishable features are more widely available than distinguishable POIs in different indoor environments. ViNav further utilizes Wi-Fi fingerprints for reducing the search space and minimizing response latency while utilizing SfM based pose calculation to provide high accuracy.
In addition to location, a walking direction of a user is also important for creating and updating navigation instructions. WalkCompass [30] detects the walking direction within a few steps using sensors available on smartphones, while Husen et al. [16] employed a dense Wi-Fi infrastructure to determine the orientation. In our work, user’s facing direction is considered to be equal to the facing direction of the smartphone camera, and is obtained from the camera pose provided by image-based localization.

7 Conclusion
We presented ViNav, a low-cost and whole-system solution for indoor navigation in this paper. ViNav is partially built on top of several existing techniques, e.g., SfM and fingerprinting. Nonetheless, ViNav brings new functions and addresses several technical challenges to enable mobile crowdsensing-based indoor navigation. It utilizes crowdsourced visual data to build 3D models of indoor spaces of interest, and detects pedestrian paths from crowdsourced user trajectories. It enables fast localization by taking advantage of the high hit-rate and low response delay of Wi-Fi fingerprinting. It also supports multi-floor navigation by locating stairs and elevators with the help of barometer readings. Moreover, with the information extracted from crowdsourced visual data, ViNav enables mobile users to search for POIs and navigate to them. The new elements introduced into ViNav make it functional and well performing, as demonstrated in the field study. ViNav would be a good primer for any system that aims to provide indoor navigation services based on crowdsourced visual and sensor data.

Acknowledgments
This work was supported by Nokia Center for Advanced Research.

References

IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 18, NO. 6, JUNE 2019

Jiang Dong received the doctoral degree in computer science from Aalto University, in 2017. His research interests include mobile crowdsensing, smart building, indoor location based services, and augmented reality. He is member of the IEEE.

Marius Noreikis received the MSc degree in computer science from Aalto University and the Royal Institute of Technology (KTH), in 2014. He is working toward the PhD degree in the Department of Communications and Networking, Aalto University. His research interests include mobile computing, scalable cloud computing, and computer vision. He is a member of the IEEE.

Yu Xiao received the doctoral degree in computer science from Aalto University, in 2012. She is currently an assistant professor with the Department of Communications and Networking, Aalto University. Her current research interests include mobile crowdsensing, augmented reality, and edge computing. She is a member of the IEEE.

Antti Ylä-Jääski received the PhD degree from ETH Zurich, in 1993. He has been a professor of telecommunications software, Department of Computer Science and Engineering, Aalto University, since 2004. His current research interests include green ICT, mobile computing, and service architecture. He is a member of the IEEE.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.