This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

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[1] 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. Secondly, 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[2] or Instagram[3]. 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)[4] 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 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.

- *J. Dong and A. Ylä-Jääski are with Department of Computer Science, Aalto University, Finland. E-mail: firstname.lastname@aalto.fi.*
- *M. Noreikis and Y. Xiao are with Department of Communications and Networking (COMNET), Aalto University, Finland. E-mail: firstname.lastname@aalto.fi.*

1. http://docs.indooratlas.com/app/
2. https://www.flickr.com/photos/franckmichel/6855169886/
3. https://www.quora.com/How-many-photos-are-being-uploaded-on-Instagram-daily

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

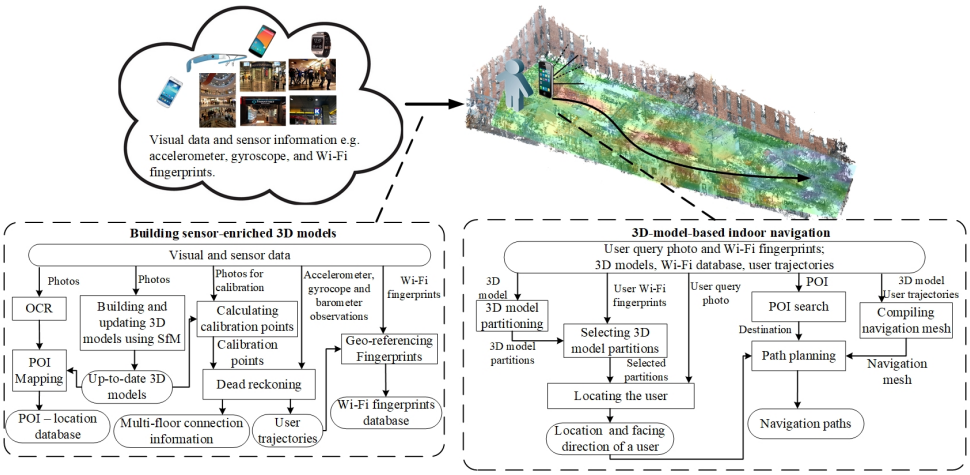


Fig. 1: Overview of a mobile crowdsensing-based indoor navigation system for smartphones.

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 fulfils 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 Section 3 and 4.

Building sensor-enriched 3D models. Firstly, *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. Secondly, *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. Thirdly, Wi-Fi fingerprints collected along the user trajectories are automatically geo-referenced and added

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

to the 3D model to enable fast localization. Fourthly, 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-enriched 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. Firstly, 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[5]. *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[6]. 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[7]. 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.

Firstly, 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.

Secondly, 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].

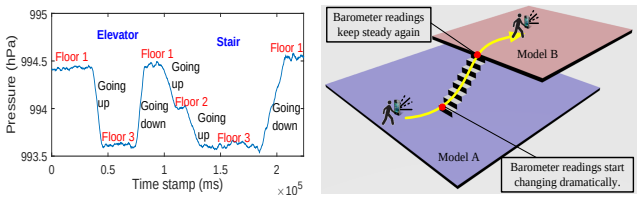
Given a set of user trajectories, *ViNav* discovers pedestrian paths by selecting the paths that have been covered by at least a certain number of trajectories. By default we set the number to 2. Each pedestrian path is represented

5. ViNav does not require the actual position of the building in the global coordinate system.

6. http://openmvg.readthedocs.io/en/latest/software/ui/SfM/control_points_registration/GCP/

7. http://ccwu.me/vsfm/doc.html#usage

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing



(a) (b)

Fig. 2: Detecting transitions between floors based on barometer readings. (a) Pressure variations when changing floors. (b) Marking a connecting path (red dots) between floors.

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.

Algorithm 1 Pseudo code of adding POI information

Input: Crowdsourced photos: P, 3D model: M
Output: Location of place of interests: L

```
for P_i ∈ P
//Apply OCR to each photo
    if Recognition(P_i) == TRUE
//Get pixel coordinates of the detected text
        TextPixelCoord = GetCoord(P_i);
//Get pixel coordinates of all the feature points extracted from P_i
        FeaturePixelCoordSet = GetFeatureCoord(P_i);
//Find the feature point which is the closest to TextPixelCoord
        point = Find(TextPixelCoord, FeaturePixelCoordSet);
//Locate the 3D coordinate of FeaturePoint from the 3D model
        L = Get3DCoord(point, M);
```

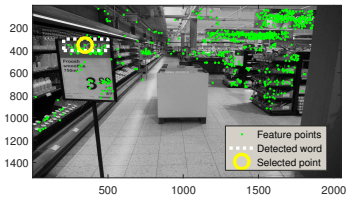


Fig. 3: An example of locating a recognized word. The dashed rectangle indicates the area where a word is recognized. The dots represent the feature points extracted from the image. The circle shows the feature point which is the closest one to the center of the recognized word.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

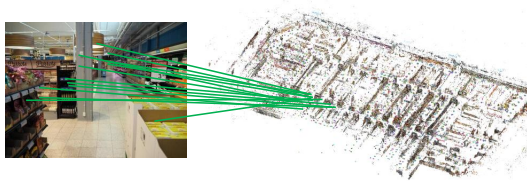


Fig. 4: Each feature point extracted from the input image has a corresponding 3D position in the SfM point cloud.

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.

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.

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

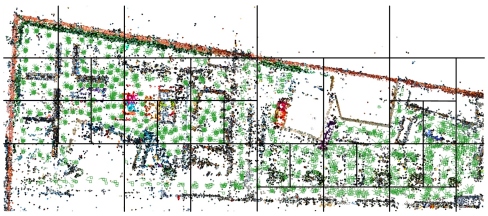


Fig. 5: An example of density-based point cloud partitioning. Each partition includes points corresponding to features extracted from no more than 100 photos. Both the width and length of each partition are larger than 5 meters. The camera positions are marked as green dots in the figure.

Algorithm 2 Pseudo code of indoor localization algorithm

Input: User's query photo: U_{photo}, Wi-Fi Fingerprints: U_{Wi-Fi}
Output: User's location: U_{loc}, User's facing direction: U_{dir}

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

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 photo. In practise, we compare reprojection errors of multiple camera pose estimations by projecting 3D points seen by a camera back to a 2D image and computing Euclidean distance between reprojections and original feature locations. We choose the pose estimation with the smallest error. The facing direction of a camera is derived from the estimated camera rotation.

4.2 Compiling Navigation Meshes

After locating the end user, we need to calculate a navigation path for her, for which a navigation mesh is fundamental. 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* firstly 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

8. http://ccwu.me/vsfm/
9. https://github.com/openMVG/openMVG

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

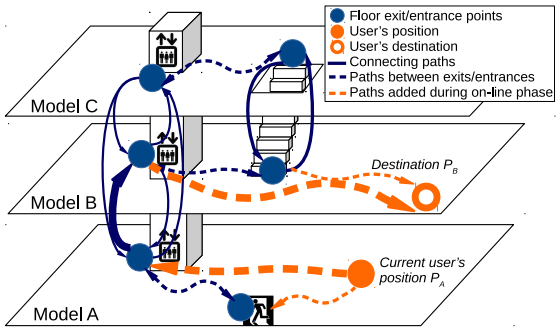


Fig. 6: Multi floor navigation graph for a 3 storey building. Blue nodes (entry and exit points) and edges (connecting and navigation paths between them) represent graph G. Orange parts are automatically added before computing a path between user's position and destination. The thicker edges represent the shortest path.

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 Duckham [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 library[11] 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.

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 10cm, while the points representing the ceiling should have the height of at least 2m.

11. https://github.com/memononen/recastnavigation

Algorithm 3 Pseudo code of calculating cross-floor navigation paths

Input: Current location: $P_A(x_s, y_s)$; 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: $entryPoints_B = getFloorEntryPoints(mesh_B, DB)$
5: // Add edges from the start position to the floor exits
6: **for** $P_x \in exitPoints_A$ **do**
7: $\quad addEdge(G, edge(P_x, P_A))$
8: // Add edges from the floor entries to the destination
9: **for** $P_e \in entryPoints_B$ **do**
10: $\quad addEdge(G, edge(P_e, P_B))$
11: // Run Dijkstra's algorithm to find the shortest path through G
12: $path = FindShortestPathWithDijkstra(G, P_A, P_B)$

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 X_i and ends at an entrance point of another floor E_j (blue solid lines in Figure 6). In other words, it defines a floor change between those particular points. Any X_i may correspond to multiple connecting paths and different entry points E_j, 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 E_i and exit X_j 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 E_j and X_i 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 Figure 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 $P_A(x_s, y_s)$ on one floor (model A), to a destination $P_B(x_d, y_d)$ located on another floor (model B), a navigation path is calculated according to Algorithm 3. The algorithm firstly utilizes A* algorithm to find paths from P_A to all exit points in model A (line 2) and from all entry points in model B (line 4) to P_B. The paths are then added to G (line 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 P_A and P_B (line 11) and send it to the user. Our algorithm supports navigating between multiple floors and arbitrary floor changing points even in cases when one has to take several elevators, escalators or stairs to reach a desired destination.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

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 PoIs, 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 m^2, 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 m^2. 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 Fig. 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.

$$E_D = \sum_{i=1}^{N} [length(s_i) E_d(s_i)] / \sum_{i=1}^{N} length(s_i) \quad (1)$$

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 outline of the CS building with an overall distance error of 0.502 meters.

Based on the updated 3D model, *ViNav* compiles a navigation mesh including 215 obstacles. The navigation mesh is visualized in Fig. 8b. We calculate the average location error of obstacles according to Eq. (2). Denote the coordinates of an obstacle in the mesh by (x_i, y_i), and its actual coordinates by (X_i, Y_i), the average location error of obstacles, denoted by $E_{obstacles}$, can be calculated as

$$E_{obstacles} = \sum_{i=1}^{N} \sqrt{(x_i - X_i)^2 + (y_i - Y_i)^2} / N \quad (2)$$

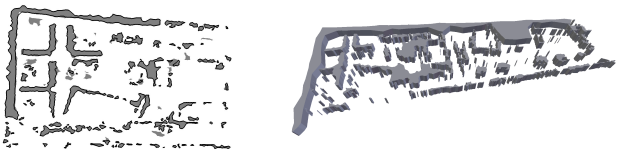
where N is the number of obstacles. As a result, the average location error is 0.78 meters, which indicates a reasonably high accuracy of the navigation mesh.

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

(a) (b) (c)

Fig. 7: Comparison of the generated 3D models with the actual layout of the CS building. (a) Official floor plan. The layout of shelves in the library has changed since the plan was released. (b) A 2D view of the initial 3D model. The blue lines marked on the figure indicate the walking traces of the volunteers. (c) A 2D view of the updated 3D model.



(a) (b)

Fig. 8: Navigation meshes generated from 3D models. (a) Navigation mesh of the library area. The grey solid polygons represent the one compiled from the initial 3D model only, while the black hollow polygons correspond to the one utilizing also the user trajectories. (b) The navigation mesh of the CS building updated in March, 2015.

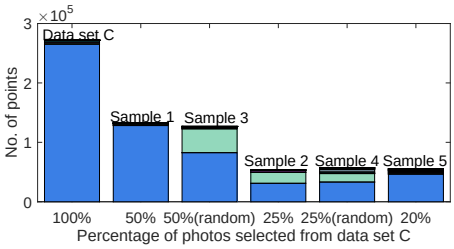
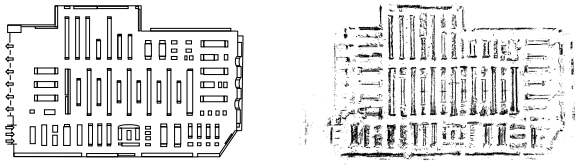


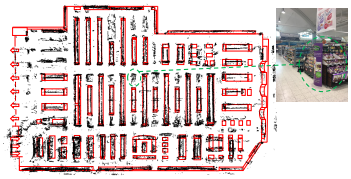
Fig. 9: Comparison of the number of 3D points included in the 3D models built from different samples. The different color in each bar represents the number of points included in each fragment of the 3D model in question.

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% of the photos. The set of sampled photos is referred to as **Sample 1**. Similarly, we evenly sampled 25% 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** an **Sample 4** by randomly selecting 50% and 25% of photos from *data set C*, respectively, regardless of location or facing direction. For comparison, we manually selected 20% 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.



(a) Floor plan of the supermarket. (b) A 2D view of the reconstructed point cloud.



(c) A comparison between the floor plan and the point cloud. The point cloud reflects the ground truth better than the original floor plan, which lacks several newly added shelves.

Fig. 10: 3D modelling result of the supermarket.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

Staircase name	No. of measurements	Position error (m)
Floor 1		
Staircase to 2nd floor	5	1.3622
Staircase to 2nd floor	4	2.5137
Elevator	15	4.2298
Floor 2		
Staircase to 1st floor	6	1.8135
Staircase to 3rd floor	13	2.6886
Staircase to 1st floor	18	1.3317
Staircase to 3rd floor	8	0.8875
Elevator	3	0.5212
Floor 3		
Staircase to 2nd floor	12	2.1184
Staircase to 2nd floor	6	4.6336
Elevator	8	1.3539

TABLE 1: Discovered connecting paths (stairs and elevators) in a 3 storey building. The second column shows how many detected positions were used in calculating a connecting path endpoint. The last one shows an Euclidean distance error between an estimated endpoint and a ground truth.

distance error of the outline is 0.68m 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.

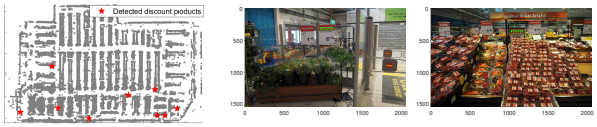
5.3 Locating stairs and elevators

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.

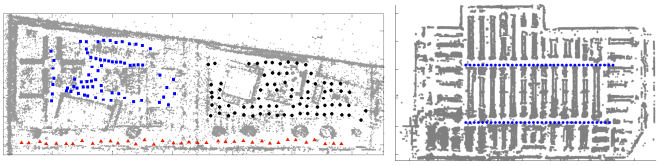
5.4 Accuracy of Mapping Points of Interest

We applied one state-of-the-art OCR tool, Google cloud vision[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[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.



(a) (b) (c)

Fig. 11: Examples of POI mapping. The red stars in (a) refer to the positions of the sale signs in the supermarket. (b) and (c) show example images from which the sale signs are recognized by text and marked with green rectangles.



(a) (b)

Fig. 12: Measurements points in two environments. (a) 185 measurement points in CS building. They were distributed in the cafe (black circle), corridor (red triangle) and library (blue square). (b) 71 measurements in the supermarket.

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.

5.5 Performance of 3D-model-based Indoor Localization

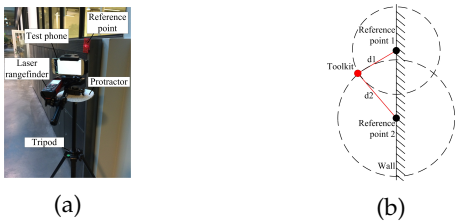
We collected a new test data set in CS building for evaluating the performance of indoor localization. The data set includes 2,220 photos taken from 185 measurement points in the CS building. As shown in Fig. 12a, the measurement points are distributed evenly across the walkable area on the ground floor.

We used a self-built toolkit, as shown in Fig. 13a, to obtain the ground truths corresponding to the data set. At each measurement point, we took photos and collected Wi-Fi fingerprints using the Android phone placed on the tripod. After taking one photo, the camera was rotated horizontally by 30 degrees before taking another one. More specifically, before taking the first photo at each measurement point,

12. https://cloud.google.com/vision

13. The supermarket provided us a list of products selling in it.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing



(a) (b)

Fig. 13: Evaluation toolkit. The left-hand figure shows a self-built toolkit that consists of a laser rangefinder, a protractor, an Android phone running *ViNav* mobile app, and a tripod as a stand. The right-hand figure demonstrates how trilateration is applied to determine the location of the toolkit based on measurements of the distance to each reference point.

Area	No. Measurement Points	No.of Photos	Hit rate		
			Image	**Wi-Fi**	*ViNav*
Cafe	78	936	91%	100%	91%
Corridor	37	444	100%	100%	100%
Library	70	840	98.6%	100%	97.1%
Total	185	2,220	95.7%	100%	95.1%

TABLE 2: Comparison of hit rate among **image**, **Wi-Fi** and *ViNav* in the CS building.

the camera was facing the same direction defined as the baseline direction. After that, the camera was rotated by 30 degrees to take the next photo. We repeated the process until the camera was facing the baseline direction again. At each measurement point, we collected 12 photos. Additionally, we used a laser rangefinder to measure the distance to each reference point. Given the measurements from the distance and locations of the reference points, we calculated the actual location of the camera based on trilateration, as illustrated in Fig. 13b. For each photo, the actual location and the facing direction of the camera was recorded and used as the ground truth for accuracy analysis.

We also collected a new test data set in the supermarket. We collected the data along two main corridors as described in Fig. 12b. We used the grid on the floor to measure the ground truth[14]. The length of one grid is 0.5m, we chose a measurement point every 1.5m. In total, we collected images at 71 measurement points. At each measurement point, we took images facing to 4 different directions. The angle between 2 images is 90 degree. 284 images were collected in the end.

Two SfM tools were applied in the two scenarios. In CS building, we used VisualSFM, which tries to mach the query image with the images registered in the 3D model. In supermarket, we adopted openMVG, which utilizes a fast approximate nearest neighbour L2 matching technique [24] to speed up the matching process.

We measured the performance of indoor localization with four metrics, namely, hit rate, location error, direction error, and response delay. **Hit rate** refers to the percentage of measurement points that can be located. When there is a hit, the accuracy of the result is measured with **location error**, and **direction error** if the result includes also user's facing direction. The location error calculates the distance

14. We were collecting the data during opening hours, using laser range finder will disturb customers. Thus we used a different way to measure the ground truth.

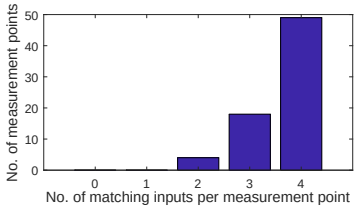


Fig. 14: 69% of the measurement points in the supermarket have 4 matching inputs, while 25.4% and 5.6% of them have 3 and 2 matching inputs respectively.

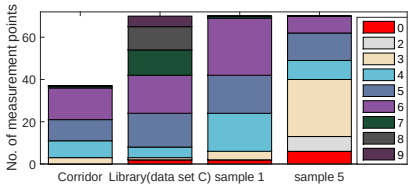


Fig. 15: The numbers of measurement points where the indicated amount of matching inputs are identified at each point in the CS building. Sample 1 and 5 are subsets of data set C in the library. Each color represents the number of matching inputs at a certain measurement point. For example, there are 15 (out of 37) measurement points in the corridor. Each point has 6 matching inputs.

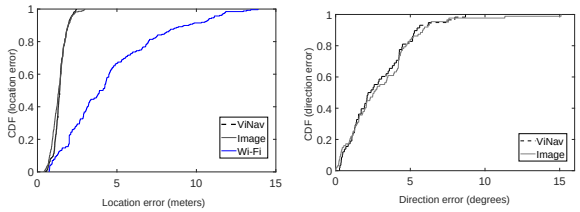
between the estimated location of the measurement point and the ground truth, while the direction error measures the angle offset from the actual facing direction. Additionally, **response delay** measures how much time it takes for the server to return the result. Note that the response delay of indoor 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing



(a) (b)

Fig. 16: Comparison of (a) location error and (b) direction error between *ViNav*, **Image**, and **Wi-Fi** in the CS building. Note that **Wi-Fi** does not return user's facing direction.

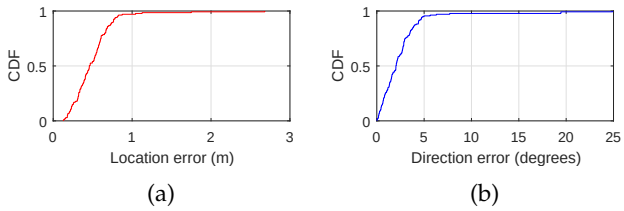
In Table 2 we compare the hit rate among *ViNav*, **Wi-Fi**, and **Image** in the CS building. In general, the hit rate is reasonably high in all the three cases. Compared with **Wi-Fi**, the two image-based schemes (i.e., *ViNav* and **Image**) fail to locate around 5% of the measurement points. This 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 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% 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 Fig. 16a and 16b, the accuracy of *ViNav* is very close to



(a) (b)

Fig. 17: (a) Location error and (b) direction error of *ViNav* in the supermarket.

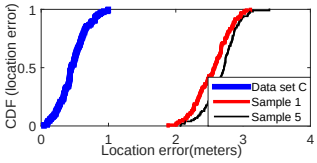


Fig. 18: Comparison of location error between scenarios where the 3D models were built from data set C, Sample 1, and Sample 5, respectively.

that of **Image**. In both cases, for at least 90% 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% of the test cases and less than 10 meters in 90% 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% 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

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

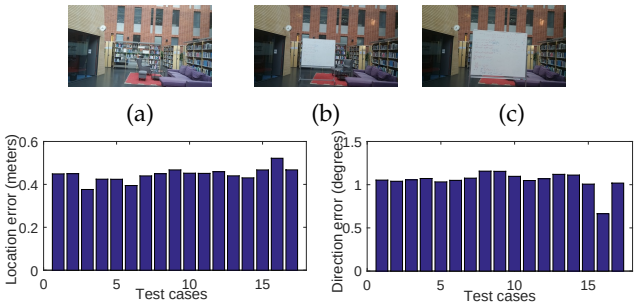


Fig. 19: Location and facing direction errors in the cases where a whiteboard was brought into the scene and then moved towards the camera.

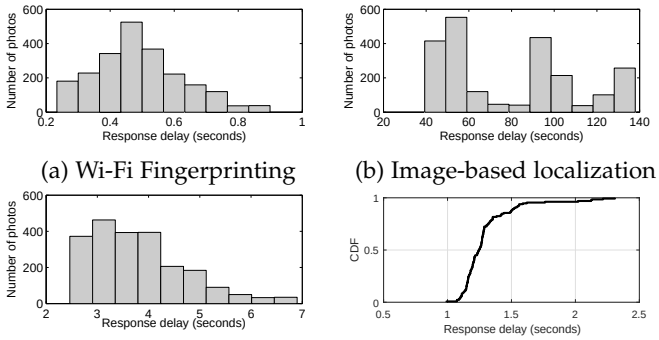


Fig. 20: Comparison of response delay between indoor localization algorithms in the CS building (a)(b)(c) and supermarket (d).

with the density of the point cloud. Nevertheless, the density does not increase if less photos are used for building the point cloud.

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 Fig. 19a – 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 Fig. 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, 2.6GHz), 64GB 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% 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.

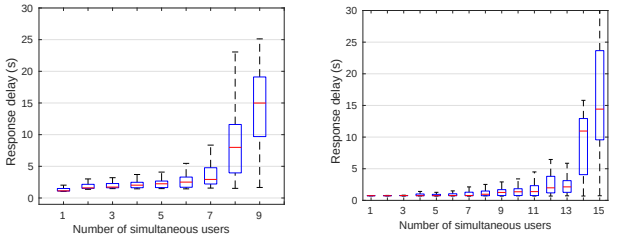


Fig. 21: *ViNav* response delays for different numbers of simultaneous users. The marking inside each box shows the median, bottom and top edges of the box indicate the 25th and 75th percentiles, the whiskers extend to the most extreme data points.

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% 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% of the matching inputs are finished within 1.5 seconds and more than 95% of them are finished within 2 seconds. The response delay is shorter, as OpenMVG supports pre-loading 3D models at the beginning[16], 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[17] p2.xlarge instance. Figure 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 Figure 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 129.62KB (s = 60.75), of which the captured image accounts for more than 95%. Therefore, a server with a typical 100Mbps 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

16. VisualSFM does not support pre-loading models since it is a closed source software.

17. Amazon Web Services (AWS) https://aws.amazon.com/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

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 wardriving. 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 SLAM (Simultaneous Localization and Mapping) 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 backpacker-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 150K 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2018.2857772, IEEE Transactions on Mobile Computing

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.

REFERENCES

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *ACM Commun.*, 54(10):105–112, 2011.

[2] P. Bahl and V. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *Proc. of INFOCOM 2000*, pages 775–784, 2000.

[3] L. Chen, L. Pei, H. Kuusniemi, Y. Chen, T. Kroger, and R. Chen. Bayesian fusion for indoor positioning using bluetooth fingerprints. *Wireless Personal Commun.*, 70(4):1735–1745, 2013.

[4] J. C. Chow. *Multi-Sensor Integration for Indoor 3D Reconstruction*. PhD thesis, University of Calgary, 2014.

[5] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In *Proc. of MobiSys'11*, pages 141–154, New York, NY, USA, 2011.

[6] J. Dong, Y. Xiao, Z. Ou, and A. Ylä-Jääski. Utilizing internet photos for indoor mapping and localization – opportunities and challenges. In *Proc. of SmartCity'15*, pages 1–6, Hong Kong, 2015.

[7] M. Duckham, L. Kulik, M. Worboys, and A. Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recogn.*, 41(10):3224–3236, 2008.

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[9] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *Proc. of ECCV'10*, pages 368–381, Berlin, Heidelberg, 2010.

[10] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Proc. of ICCV'09*, Kyoto, Japan, 2009.

[11] R. Gao, Y. Tian, F. Ye, G. Luo, K. Bian, Y. Wang, T. Wang, and X. Li. Sextant: Towards ubiquitous indoor localization service by photo-taking of the environment. *IEEE Trans. Mobile Comput.*, 15(2):460–474, 2016.

[12] R. Gao, M. Zhao, T. Ye, F. Ye, G. Luo, Y. Wang, K. Bian, T. Wang, and X. Li. Multi-story indoor floor plan reconstruction via mobile crowdsensing. *IEEE Trans. Mobile Comput.*, 15(6):1427–1442, 2016.

[13] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proc. of MobiCom'14*, pages 249–260, New York, NY, USA, 2014.

[14] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Commun. Surveys Tuts*, 11:13–32, 2009.

[15] J.-Y. Huang, S.-H. Lee, and C.-H. Tsai. A fast image matching technique for the panoramic-based localization. In *Proc. of ICIS 2016*, pages 1–6, Dublin, Ireland.

[16] M. N. Husen and S. Lee. Indoor human localization with orientation using wifi fingerprinting. In *Proc. of ICUIMC'14*, pages 109:1–109:6, New York, NY, USA, 2014.

[17] M. R. Lehto and S. J. Landry. *Introduction to human factors and ergonomics for engineers*. Crc Press, 2012.

[18] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proc. of UbiComp'12*, pages 421–430, New York, NY, USA, 2012.

[19] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Trans. Syst., Man, Cybern.*, 37(6):1067–1080, 2007.

[20] Z. Liu, L. Zhang, Q. Liu, Y. Yin, L. Cheng, and R. Zimmermann. Fusion of magnetic and visual sensors for indoor localization: Infrastructure-free and more effective. *IEEE Trans. Multimedia*, 19(4):874–888, 2017.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[22] G. Lu and C. Kambhamettu. Image-based indoor localization system based on 3d sfm model. In *Proc. of SPIE*, Intelligent Robots and Computer Vision, 2014.

[23] R. Martin-Brualla, Y. He, B. C. Russell, and S. M. Seitz. The 3d jigsaw puzzle: Mapping large indoor spaces. In *Proc. of ECCV'14*, pages 1–16, Zurich, Switzerland, 2014. Springer.

[24] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[25] K. Muralidharan, A. J. Khan, A. Misra, R. K. Balan, and S. Agarwal. Barometric phone sensors: More hype than hope! In *Proc. of HotMobile '14*, pages 12:1–12:6, New York, NY, USA, 2014. ACM.

[26] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of ISMAR'11*, pages 127–136, Basel, Switzerland, 2011.

[27] L. T. Nguyen and J. Zhang. Wi-fi fingerprinting through active learning using smartphones. In *Proc. of UbiComp'13 Adjunct*, pages 969–976, New York, NY, USA, 2013.

[28] B. E. Okorn, X. Xiong, B. Akinci, and D. Huber. Toward automated modeling of floor plans. In *Proc. of 3DPVT'10*, Paris, France, 2010.

[29] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proc. of Mobicom'12*, pages 293–304, New York, NY, USA, 2012.

[30] N. Roy, H. Wang, and R. Roy Choudhury. I am a smartphone and i can tell my user's walking direction. In *Proc. of MobiSys'14*, pages 329–342, New York, NY, USA, 2014.

[31] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. of ICCV'11*, pages 667–674, Barcelona, Spain, 2011.

[32] R. Smith. An overview of the tesseract ocr engine. In *Proc. of ICDAR 2007*, volume 2, pages 629–633. IEEE, 2007.

[33] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. Seitz. Scene reconstruction and visualization from community photo collections. *Proc. of the IEEE*, 98(8):1370–1390, 2010.

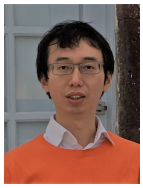
[34] X. Tong and D. A. Evans. A statistical approach to automatic ocr error correction in context. In *Proc. of WVLC-4*, pages 88–100, 1996.

[35] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proc. of MobiSys'12*, pages 197–210, NY, USA, 2012.

[36] C. Wu. Towards linear-time incremental structure from motion. In *Proc. of 3DV'13*, pages 127–134, Seattle, USA, 2013.

[37] Q. Ye and D. Doermann. Text detection and recognition in imagery: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(7):1480–1500, 2015.

[38] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao. Travinavi: Self-deployable indoor navigation system. *zheng2017travi*, 25(5):2655–2669, 2017.



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.

Marius Noreikis is a Ph.D student at Department of Communications and Networking, Aalto University. He received his M.Sc in computer science from Aalto University and Royal Institute of Technology (KTH) in 2014. His research interests include mobile computing, scalable cloud computing and computer vision.

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

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