Peer-to-Peer Networking and Applications (2021) 14:3718–3736
https://doi.org/10.1007/s12083-021-01216-0

DeepNav: A scalable and plug-and-play indoor navigation system based on visual CNN

Jian Gong[1] · Ju Ren[1] · Yaoxue Zhang[2]

Received: 25 March 2021 / Accepted: 26 June 2021 / Published online: 10 July 2021

Abstract
With the proliferation of smartphones, recent years have witnessed the rapid development of smartphone-based indoor navigation systems. However, existing solutions either bear high deployment cost or cannot support large-scale navigation. A scalable and plug-and-play indoor navigation system is still highly desirable. In this paper, we propose DeepNav, a new indoor navigation system that fully uses visual CNN to realize large-scale navigation. DeepNav adopts a single-pilot deployment scheme to realize fast deployment. It divides the indoor area into dense sub-areas to simplify image-based location matching while ensuring reasonable resolution. Practical realization of DeepNav entails a set of key challenges, e.g., invalid image recognition, classification of thousands of labels and under-fitting. In order to solve these challenges, we propose invalid image filter, subgroup sigmoid layer and movable object filter, respectively, for DeepNav. Finally, we implement a prototype of DeepNav on commercial smartphones. Experimental results demonstrate that DeepNav can be quickly deployed (e.g., within an hour in a 4-storey building) with an average localization error of 2.3 meters.

Keywords Indoor navigation · Visual CNN · Scalability · Plug-and-play · Smartphone

1 Introduction

As GPS changes people's lives outdoor, indoor navigation is also becoming highly desirable. Generally, indoor navigation applications can be divided into two categories: one for conventional navigation demands in public buildings, e.g., searching for specific stores in a shopping mall, and the other for occasional events in dedicated buildings, e.g. commercial activities in office buildings and route guidance of meetings in hotels. Usually, there are no pre-deployed infrastructures for navigation in both navigation scenarios. For the former, in order to satisfy the diversity of navigation routes and the coverage of uncertain destinations, indoor navigation system is required to realize large-scale deployment and precise localization. Whereas for the latter, how to fast deploy the indoor navigation system with a limited cost becomes the key requirement.

At present, most of existing indoor localization and navigation systems are designed for conventional navigation demands. Technically, they are developed based on three kinds of techniques, including Wi-Fi fingerprints [1, 2], landmark with dead reckoning [3–8] and Simultaneous Localization And Mapping (SLAM) [9, 10]. Wi-Fi fingerprint based methods [1, 2] use the collected fingerprints of pre-deployed Wi-Fi devices to locate indoor devices by triangulation location. However, they need to acquire complete building floor maps and precise location information of Wi-Fi devices, which are not easy and sometimes impossible to obtain in realistic scenarios. In order to get rid of the dependence on floor map, the landmark based methods [3–8] are proposed by setting up landmarks in a few key locations that have absolute accuracy for indoor localization. The locations between two neighboring landmarks are estimated by the movement of the pedestrian with their gait. However, in order to accurately model the path relationships between different landmarks, they need to collect a large amount of gait data by crowdsourcing, leading to

✉ Jian Gong
gongjian@csu.edu.cn

Ju Ren
renju@csu.edu.cn

Yaoxue Zhang
zyx@csu.edu.cn

1 School of Computer Science and Engineering, Central South University, Changsha, China

2 Department of Computer Science and Technology, Tsinghua University, Beijing, China

considerable labor and time consumption. More importantly, the walking areas of different pedestrians are repeated, making the collected data very redundant. With the proliferation of smartphones, recent studies focus on leveraging the embedded cameras and enriched computing capabilities of smartphones for indoor navigation. One kind of solutions are visual SLAM-based methods [9, 10], which model a complete 3D indoor map through video keyframe matching and rigid body rotation relationships. In order to solve the closed-loop problem unique to this kind of methods, they have to collect a large number of redundant images, resulting in extensive time consumption. For occasional navigation, FollowMe [11] proposes a plug-and-play navigation method leveraging the historical route traces taken by other users. However, this method demands users to start and finish navigation at specific place, which has great limitation and could not realize large-scale deployment.

By now, there is still no mature indoor navigation solution that can well satisfy both of the conventional and occasional navigation demands. The gap between functional integrity and deployment speed motivates us to raise a question: Is there a solution that can be deployed in a very short time while realizing large-scale navigation? For the landmark based methods with dead reckoning, if we remove the dead reckoning process and set dense landmarks, can we reduce the redundancy of data collection? The challenge may still be arisen by that dense landmarks also involve massive data collection labor to guarantee precise localization. Therefore, it is important to design a method that can accurately locate the landmarks in a convenient and labor-saving data collection way.

As the one of the fastest growing branch in deep learning, Convolutional Neural Network (CNN) has shown the ability of surpassing human in object classification task during the past few years [12, 13]. Meanwhile, this computation-intensive technology has evolved to be deployable on smartphones and can achieve real-time processing [14–16]. In this paper, we propose DeepNav, a new indoor navigation system for smartphones, which adopts CNN as the basic recognizing module to recognize places. In DeepNav, we adopt the pilot-follower architecture to implement localization and navigation. All the data of navigation routes are recorded by the captured videos of the pilots' smartphones and then used for CNN training. An intuitive method for using CNN as the location classifier is to manually divide the indoor areas into multiple places, where the pilot collects a large number of images in different directions and then uses them to train the CNN after manual labeling. However, it may raise significant challenges in data collection, deviation detection and precise localization. The primary contribution of DeepNav is to provide a systematic solution for addressing these challenges to achieve fast and scalable indoor navigation.

First, due to numerous factors in practical usage scenarios (e.g., pointing to non-walking direction, camera occlusion, route deviation), the input image usually becomes invalid and may not belong to any existing label. Thus, how to accurately filter out the invalid images becomes very important. The existing image classification CNN framework can achieve this by adding a background class, which is a special class that represents no object. However, background class training requires a large number of positive samples for each model, which further increases the workload. In DeepNav, we first replace the softmax layer with a sigmoid layer, and then filter the invalid images by the proposed invalid image filter based on outlier in the sigmoid layer, to improve the training efficiency.

Second, in order to achieve high localization accuracy, we divide the indoor area into multiple fine-grained sub-places. Each sub-place denotes a place node labeled uniquely in CNN model. In CNN-based recognition, a large building often contains thousands of place nodes. The number of labels is even larger than the current largest public image classification dataset ILSVRC (containing 1000 categories). In order to execute CNN models directly on the smartphone in real time, it is inapplicable to use large networks that have state-of-the-art recognition accuracy but are too complex for mobile devices (such as ResNet-101). Additionally, due to the limitation of deployment speed, we can only collect a small number of training data, which further increases the difficulty of precise locating. In order to increase the recognition accuracy, we propose a subgroup sigmoid layer, which can narrow the classification range by clustering recognition results. We also propose a moving object filter to alleviate the occlusion from moving objects.

The main contributions of this paper are summarized as follows (Table 1).

- We propose DeepNav, a new indoor localization and navigation architecture for smartphones, which fills the gap between large-scale navigation and fast deployment. To the best of our knowledge, this is the first work to provide a systematic visual CNN based indoor navigation solution.
- We address several practical challenges due to the use of CNN: (1) A subgroup sigmoid layer is designed to alleviate the classification difficulties by massive labels and small model size. (2) An invalid image filter and a movable object filter are developed to address the invalid recognition problem and improve the system performance in dynamic environment, respectively.
- We implement a prototype of DeepNav and evaluate its performance on smartphones. Experiment results

Table 1 Motivation of DeepNav

	Deployment speed	Cost of human labor	Large-scale navigation	Navigation routes
Travi-Navi [1]	low	high	support	unlimited
ViNav [10]	low	high	support	unlimited
FollowMe [11]	fast	low	not support	limited
DeepNav	**fast**	**low**	**support**	**unlimited**

demonstrate that DeepNav can be deployed in a four-story building within an hour. Additionally, it achieves an average localization error of 2.3 meters.

2 System overview

As shown in Fig. 1, DeepNav is mainly composed of map constructing stage and navigation stage.

Map constructing stage comprises the following steps. (1) **Route recording.** A pilot records the navigation routes using the embedded camera on smartphone and upload them to the server. In order to alleviate the data redundancy problem, we design an easy-to-use data collection scheme which can finish data collect by only one pilot even in large-scale buildings. (2) **Semi-automatic data labeling.** The pilot only needs to input the timestamps of a few key frames in the videos, then all the remaining labeling works would be finished automatically. (3) **CNN training.** Each place node in CNN is labeled uniquely and the video frames are used to train the last layer of CNN. (4) **Add additional information and construct map.** The pilot labels point-of-interests (PoIs) by observing the scenes in videos and adding them to the corresponding place nodes.

The navigation stage comprises the following steps. (1) **Download maps and models**. Before starting navigation, the follower needs to download a model file and a map file corresponding to the building. Then, all the navigation functions can work offline on smartphones. (2) **Navigation on smartphones.** With the downloaded CNN model and map file, DeepNav navigates the followers by recognizing the image frames of smartphones in real time. However, since DeepNav adopts small CNN model, we propose a subgroup sigmoid layer and moving object filter to tackle the problems of low accuracy and under-fitting.

3 Map constructing stage

In this section, we present the technical details of map constructing in DeepNav. Summarily, DeepNav has the following advantages for map constructing.

– Data collection is convenient to complete. Even in a large-scale building, only one pilot is needed to finish the data collection. It can significantly save human labor without the need of crowdsourcing.

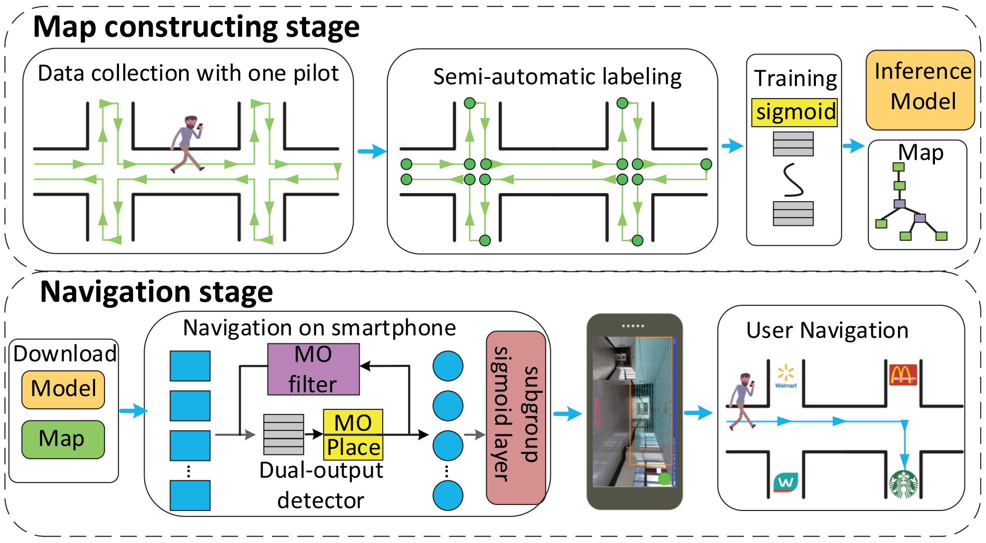
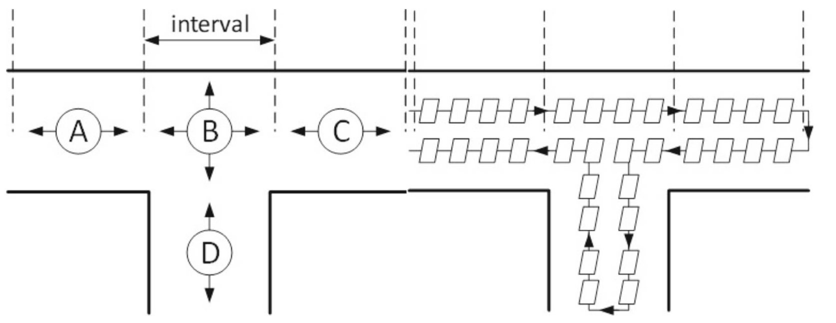


Fig. 1 Architecture of DeepNav

Fig. 2 Single pilot data collection example



(a) Place node configuration (b) Easy-to-use data collection

– Independent of dedicated infrastructures and indoor floor maps. All the data is collected by the cameras mounted on smartphones, and the PoI labeling does not need indoor floor maps.
– Data labeling is semi-automatic. Pilot only needs to label a small number of video frames, then all the remaining frames are automatically labeled by the designed method.

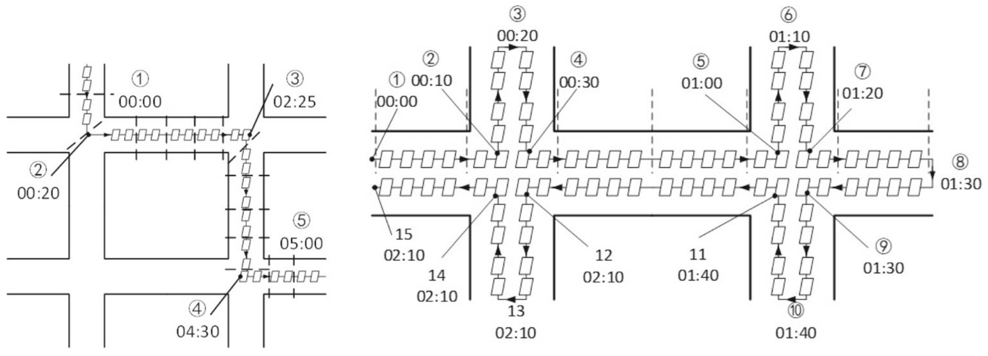
3.1 Easy-to-use image collection and labeling for dense place nodes

The main target of image collection is to realize lowest labor cost while maintaining high localization resolution. We design to make all the image collection work able to be done by one pilot using smartphone. The image collection is based on two principles. First, only images in the walking direction are collected at each place node. Thus, the pilot is able to collect images as he walks and keep his smartphone camera in front of him, which is easy to implement. Second, the images are collected only once in each direction of each place node. Thus, the pilot does not have to collect repetitive images, which saves time. In order to make these two principles feasible while guarantee that we can cover all place nodes in the building, we will do some definitions and model the map.

We use topological map to represent indoor navigation area. Each place and its corresponding direction are modeled as a place node. A place node is also the minimum space unit in the map. For a T-shape indoor area shown in Fig. 2a, there are totally 10 place nodes set in 4 different places and each of them represents a place with corresponding direction. For instance, place A contains two directions. So we have two place nodes in place A, each of which points to a unique direction. We further classify the place node into different categories according to special situations. **Cross node:** if two or more place nodes exist in the same place, e.g., the two place nodes in place A, then these place nodes are defined as cross nodes. **Spin node:** the place nodes do not belong to cross nodes but cross large turning angles or crossroads are referred to as spin nodes. For instance, the 2nd, 3th and 4th place nodes in Fig. 3a are spin nodes in this map.

Now we describe how we represent the image and place node in the maps. The distance between two adjacent place nodes is defined as interval. In Figs. 2 and 3, each small rectangle represents a collected image and the arrows represent the walking directions while collecting images. Each place node is labeled uniquely in the CNN training process. The place node with the highest recognition confidence in the sigmoid layer represents the most probable place node corresponding to an image. In the

Fig. 3 Semi-automatic data labeling example



(a) Occasional labeling (b) Large-scale labeling area centered at a place node with the width of an interval, all the images in a specific walking direction belong to this place node. For example, in Fig. 2b, there are totally 4 images collected at place node A in the right direction, then all these 4 images are labeled as place node A.

In Fig. 2b, it may seem that this area have many images to be collected. But in fact, the pilot only needs to collect one video according to the walking route instructed by the arrows in Fig. 2b. Then we can automatically extract images from the video and label them according to the time sequence. In the video, each frame can be extracted as an image. In real buildings, we can observe many similar corridor structures as in Fig. 2b. Even for a larger and more complicated area, e.g., Fig. 3b, we also only need to take one video for data collection by the walking routes instructed by the arrows. This kind of routes are not hard to think since it only need to guarantee that all the corners are traversed by once. In our experiment, an experienced pilot can complete all the data collection within 40 minutes in a 4-storey teaching building.

The semi-automatic data labeling in DeepNav is designed as follows. In occasional navigation scenes (illustrated in Fig. 3a), the route is usually simple and contains only one route from the entrance to the destination. Therefore, there is usually no cross node and only spin nodes in the route. The spin nodes in this map are marked as 2,3 and 4. When observing the video frames, the pilot could get the time of a specific frame in the video, which is called a time stamp. If we observe the time stamps of some key place nodes, such as spin nodes, they could be used for splitting the videos into multiple fragments which only contain straight routes. A fragment of a video is constructed by a sequence of successive frames. In this case, the route is divided into 4 sub-routes by the spin nodes. Each sub-route is a straight route. Then we require the pilots walking in a roughly constant speed, so the video frames in a straight route could be automatically labeled with incremental labels(e.g., 200, 201, 202) automatically in time ascending order. In a large-scale navigation scene with multiple cross nodes in each crossroad, the video is split into multiple video fragments according to the relationship of actual places. The video frames between two adjacent cross nodes are grouped into one fragment. The pilot only needs to input the time stamp of the video frame corresponding to each cross node and which cross place nodes belong to the same place. Then, the video frames between two adjacent cross nodes can be automatically labeled by predefined interval. For example, in Fig. 3b, there are totally only 15 video frames needed to be labeled by human, then the rest thousands of frames are automatically labeled.

After pilot finishing data collection and labeling, CNN will be trained to recognize different places based on the collected data. Finally, DeepNav generates a map file and a CNN model file for followers to navigate in an off-line way.

3.2 Invalid image filter

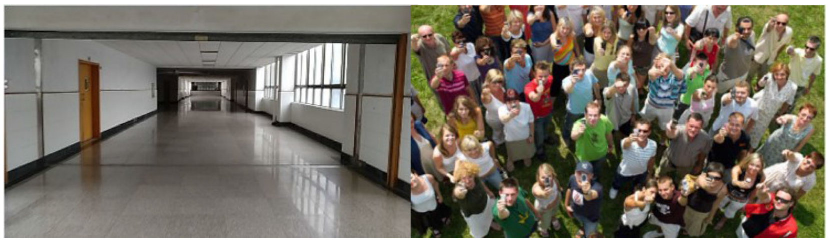
Due to practical sophistications in indoor navigation (e.g., occlusion, camera blur, invalid camera direction), the images collected by the followers usually become invalid images, which represent the images that most of the features are lost and can not be recognized. Invalid images cause degradation of localization accuracy, so it is significant to distinguish them with normal images. The existing CNN based image classification methods can distinguish the irrelevant images from normal images by adding a background class [17]. All the invalid images are sorted into background class. However, this method needs to collect massive invalid images to train the model recognizing the background class, which increases the burden of human labor and decreases deployment speed.

Additionally, when the input maps are updated, e.g., changing stores, decorating or adding new stores, the CNN model has to be updated accordingly. Due to the dependence among the outputs of the softmax layer, the whole CNN model needs to be retrained even only one element of outputs is updated. The additional training process adds unnecessary time in deployment.

In order to address the above problems, we propose the invalid image filter, as shown in Fig. 5. It replace the softmax layer with sigmoid layer in CNN in the training process by treating place classification as a binary classification task. In binary classification, each output element calculates a confidence of whether the input image belongs to a specific class. Therefore, we can decouple the dependence among all the output elements and realize both classification and judging the validness of the input images. Additionally, when adding new labels or new training data to the trained model, only the elements corresponding to the specific classes need to be retrained.

However, the binary classification introduces the problem of confusing confidence problem. Generally, the recognition confidence of navigation area outputted by the CNN model should be higher than the confidence of non-navigation area. For instance, when an image points to the ceiling, the CNN should be able to recognize it with 0 confidence because the image does not belong to any valid place. But the experimental results show an opposite trend. For example, Fig. 4a and b belong to navigation and non-navigation areas respectively. However, the average navigation confidence of Fig. 4a is much lower than that of Fig. 4b. It means that it is hard to judge whether an image belongs to navigation area according to the output confidence of sigmoid layer.

We use the term of ’negative samples’ to represent the image samples do not belong to any of the classification



(a) Recognition confidence of navigation region: 41%, 13%, 6%, 6%, 6%, 2%, 1%, 1%, 1%, 1%. (b) Recognition confidence of irrelevant region: 98%, 98%, 97%, 96%, 96%, 96%, 96%, 95%, 95%, 94%.

Fig. 4 The counter-intuitive phenomenon of top-10 recognition confidence. The confidence represents the probability of whether the images belong to existing place nodes. There are hundreds of places node confidence output and we only show the top-10

labels. The reason of this phenomenon is that the number of negative samples is much larger than the positive samples in the training process, leading to the suppression of the output confidence. For the training samples in navigation area, the sigmoid layer tends to output confidence close to 0 because the training samples are suppressed. For the non-navigation samples, the sigmoid layer has a higher probability to output high confidence because the samples are not involved in training process. We also find that when the input image is similar to the navigation area, the average recognition confidence tends to be lower even though the image does not belong to the existing labels, which further confirms the above idea.

Based on these findings, we design a new method to judge whether an image belongs to the navigation route. The process is shown in Fig. 5.

For each input image, we first process it using the CNN model and sort the recognition confidence in the sigmoid layer. Specifically, we define the sorted node ID list in which each node is composed of node ID and the corresponding recognition confidence as:

$$Node_s^i = ArgSort(L_s^i), \tag{1}$$

where L_s^i is the recognition confidence list of all the nodes of frame i. $Argsort()$ sorts them by confidence value in a descending order.

Next, since the top few recognition confidence levels are usually high and the bottom bunch of recognition confidence are usually low, they have no contribution to distinguish the validness of image. Therefore, we remove them and only reserve the middle $M_e - M_s$ recognition

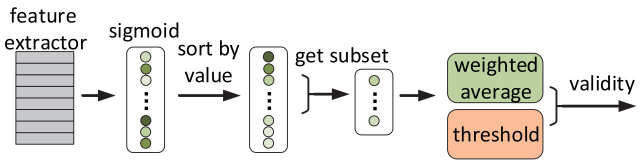


Fig. 5 Invalid image filter

confidence. Then, we define the valid recognition probability of a navigation area as

$$\bar{P}_{ib}^i = \frac{1}{M_e - M_s} * \sum_{x=M_s}^{M_e} L_s^i(Node_s^i(x)) * P_m, \tag{2}$$

where $L_s^i(x)$ denotes the recognition confidence of frame i at place node x, P_m denotes the weight for movable object detection. The setting of M_e needs to guarantee that it can distinguish the difference between navigation area and non-navigation area. Then, we remove the top M_s recognition confidence from these M_e recognition confidence to avoid confusion between recognition confidence of navigation area and non-navigation area. Finally, we calculate the average of these $M_e - M_s$ confidence to judge the validness of an image.

Finally, we use F_{ib}^i to represent whether frame i belongs to the navigation area, which is calculated as

$$F_{ib}^i = \begin{cases} 0, & \bar{P}_{ib}^i > T_{ib} \\ 1, & \bar{P}_{ib}^i \leqslant T_{ib}, \end{cases} \tag{3}$$

where T_{ib} is the probability threshold. If F_{ib}^i equals to 1, it represents that image i belongs to navigation area, and vice versa.

3.3 Solution for unbalanced training samples

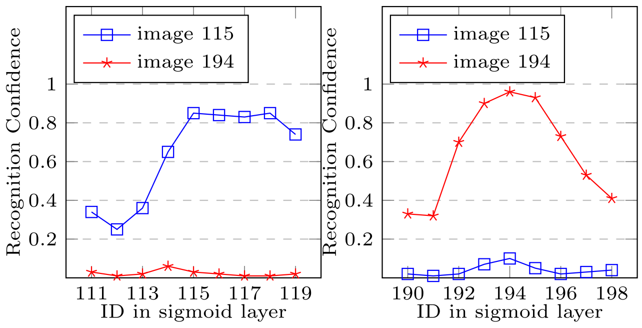
The introduction of sigmoid layer raises the problem of unbalanced training samples. For each element in sigmoid layer, only a few training images are regarded as positive samples and the rest are all regarded as negative samples. A large-scale building may contain thousands of place nodes, so the numbers of positive training samples and negative training samples are extremely unbalanced, leading to significant performance degradation and even model divergence. Lin et al. [18] proposed focal loss function to address the unbalanced training samples problem in the model used in object detection field. Inspired by this approach, we replace the original cross entropy loss function with the focal loss function [18], and then adjust the hyper parameters α and γ in the training process to alleviate the problem of unbalanced training samples. The hyper parameters α and γ control the learning preference of training samples. α should be adjust first. Usually, it could be set at about 0.5 at first and then fine-tune the value according to the extent of unbalance. Then the γ should be adjust secondly. It could be adjust at range of 0-5 according to the extent of unbalance.

4 Navigation stage

In the navigation stage, DeepNav uses the embedded camera on smartphones to navigate users based on the downloaded CNN model files. The navigation architecture is shown in Fig. 6. The navigation is divided into 3 main steps: filter movable objects, localize users and produce navigation instruction. It first eliminates the influence of pedestrian by the movable object filter. Then it estimates the preliminary localization by the dual-output detector. In order to eliminate the invalid detection and improve localization accuracy, we propose the subgroup sigmoid layer. Finally, the navigation instructions are generated from the localization result by the instruction module.

4.1 Subgroup sigmoid layer and localization

For smartphone based indoor navigation, large CNN models are unsuitable since the memory capability of smartphone is small and the lack of high-performance GPU making it not able to support real-time inference. On the other hand, CNN model with small size could not achieve high recognition accuracy due to under-fitting. By analyzing the localization data collected in realistic scenarios, we find that we could design an algorithm, denoted subgroup sigmoid layer, to narrow down the range of localization and filter out accidental recognition errors.



(a) Recognition confidence of images around node 115. (b) Recognition confidence of images around node 194.

Fig. 7 Recognition confidence correlation of neighbor nodes

4.1.1 Place node cluster

The first phenomenon we find in the experiment data is place node cluster. When we observe the recognition confidence of an image which belongs to a specific place node, we find that the neighbor nodes also have high recognition confidence.

Figure 7 shows the recognition confidence outputted by the sigmoid layer of place node 115 and 194. It can be observed that for the image belonging to node 115 (node 194), the recognition confidence of node 115 (node 194) is much higher than that of node 194 (node 115). Moreover, for the image belonging to a specific node, the recognition confidence of its neighboring nodes gradually decreases as their distance to the node increases. It indicates that the recognition confidence among neighboring place nodes has strong correlations. Therefore, we cluster the place nodes with top N recognition confidence into multiple clusters according to their relations in the map. The sum of the recognition confidence of all the place nodes belonging to the same cluster denotes a cluster confidence. If the cluster confidence of a cluster is high, the location of the image is likely to be in this cluster.

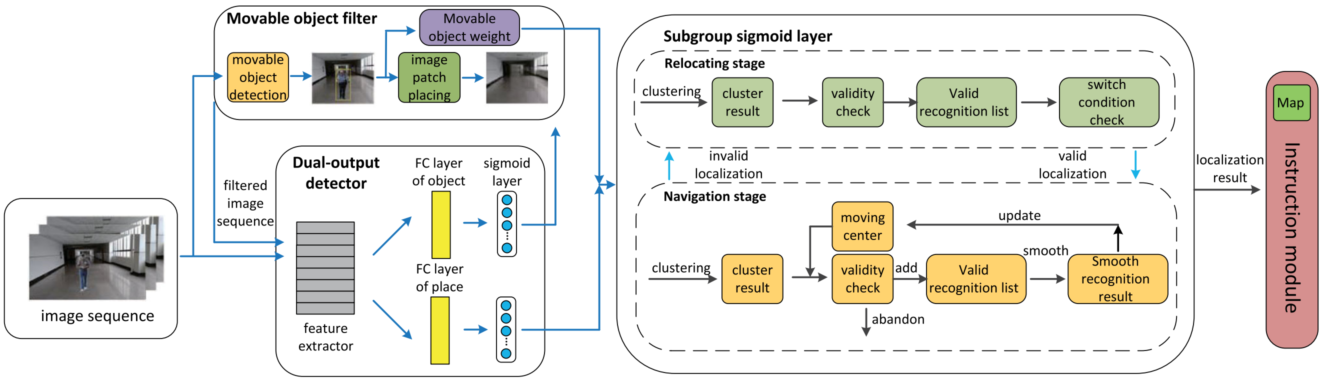


Fig. 6 Navigation architecture of DeepNav running on smartphone

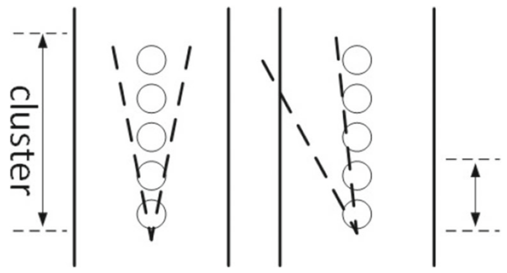


Fig. 8 Node cluster phenomenon

Although F_{ib}^{i} can effectively distinguish navigation area and non-navigation area, it is still hard to distinguish navigation direction and non-navigation direction. The reason is that the images belonging to a specific navigation area but not in the navigation direction are still similar to some place nodes. We also find that the number of place nodes increases obviously when the images are captured in the navigation direction. Meanwhile, it decreases when the images are captured in the non-navigation direction.

Figure 8 shows the comparison of two situations. It can be observed that the image captured in the navigation direction covers more successive place nodes than that captured in the non-navigation direction. Thus, the left situation can achieve a higher cluster confidence than the right situation. Therefore, we can determine whether an image is in navigation direction by the cluster confidence and the node number in cluster. If the cluster confidence is high and the cluster owns many place nodes, we believe that the image is more likely to be in the navigation direction and vice versa.

4.1.2 Noise filter

In some special cases, such as occlusion and out of focus, the recognition might be impacted temporarily, as shown in Fig. 9. We can find that in the frames where occlusion

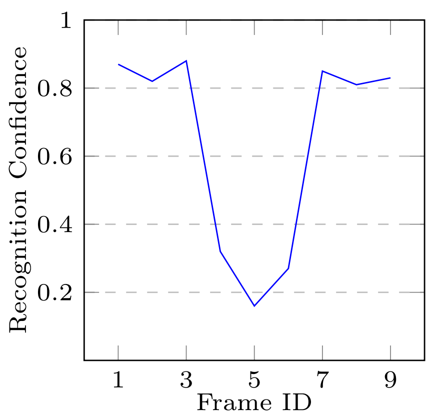


Fig. 9 Incidental recognition error caused by occlusion

appears, the recognition confidence decreases obviously. But the confidence of neighboring frames are not impacted by the occlusion. Therefore, we can correct the recognition results of occluded frames according to their adjacent recognition results.

4.1.3 Implementation details

From the node cluster phenomenon, we could replace the node confidence with cluster confidence to get more reliable recognition result, because the cluster confidence is more stable than a single node confidence. From the noise filter phenomenon, we could use recognition results from neighbor frames to correct incidental recognition error. Since these two solutions behave in different aspects, we combine them together and propose a unified solution, which denotes subgroup sigmoid layer. First, based on the recognition results from multiple successive frames, we calculate a weighted average recognition result, which is called 'moving center'. It may cause a certain lag on the user's real-time location but has a high probability of being located near the user's real-time location. Then, we set a radius for the moving center. All place nodes within the radius are sorted into the first group. Second, we put place nodes into different clusters according to their recognition confidence and sort the place nodes with high cluster confidence into the second group. Finally, we use the intersection of these two groups as the final group and take the node with the highest recognition confidence in this group as the localization result of the current frame.

Navigation in DeepNav has two main stages: navigation and relocating. When the system is initializing or cannot locate valid place for a long time, it enters relocating status. First, we define the cluster list as

$$L_c^i = getCluster(Node_s^i(0, M_c), map), \tag{4}$$

where $Node_s^i(0, M_c)$ denotes the first M_c nodes in $Node_s^i$. i denotes the frame ID. The function $getCluster()$ clusters the nodes in input node list into multiple clusters according to their relations in the map. For instance, the input node list is [546, 547, 548, 834, 835, 1435] and the nodes with adjacent IDs are also adjacent in the map, the list after clustering is [[546,547,548], [834,835], [1435]]. Correspondingly, we define the cluster confidence list as

$$L_{cc}^i = sumP(L_c^i, L_s^i), \tag{5}$$

where $sumP()$ sums the recognition confidence of the place nodes in the cluster list and adds them to the cluster confidence list according to the clustering order.

After clustering, we need to find out the cluster with maximum confidence. We can define the max confidence cluster and the corresponding cluster confidence as

$$L_{mc}^i, P_{mc}^i = getMaxCluster(L_c^i, L_{cc}^i), \tag{6}$$

where $getMaxCluster()$ is to obtain the cluster with the maximum cluster confidence and the corresponding cluster confidence in the cluster list. We use F_{id}^i to represent whether frame i is captured in the navigation direction, which is defined as

$$F_{id}^i = \begin{cases} 1, & len(L_{mc}^i) \geqslant T_{cn}, F_{ib}^i = 1 \\ 0, & otherwise \end{cases}, \tag{7}$$

where $len()$ returns the number of place nodes in the cluster. T_{cn} is a pre-defined threshold which represents the minimized place node number of a straight route.

In order to calculate the proportion of valid frames in recent N frames, we define valid frame proportion as

$$P_{vf}^i = \frac{N_v}{N}, \tag{8}$$

where N_v is the number of valid frames (i.e., $F_{id}^i = 1$) in recent N frames. Then, we can calculate a smooth recognition confidence for each place node x according to the recognition results in recent N frames.

$$\bar{L}_s^i = \frac{1}{N_v} \sum_{x=i-N}^{i} F_{id}^i * L_s^i(x) * P_m^i, \tag{9}$$

where P_m^i is the weight of moving object detection. Similarly, we can calculate max cluster $\bar{L}_{mc}^i$ from $\bar{L}_s^i$.

The conditions of switching from relocating stage to navigation stage are: 1. $len(\bar{L}_{mc}^i) > \bar{T}_{cn}$, where $\bar{T}_{cn}$ is the threshold value of cluster number. 2. $P_{vf}^i > \bar{T}_{vf}$, where $\bar{T}_{vf}$ represents the threshold value of valid frame rate. Usually, it can be set at range of 0.7-0.9, which represents that when there are over 70%-90% frames in recent N frames are valid, we regard it as valid detection. Before switching to navigation status, we need to assign value to the 'moving center', representing the valid localization information. It is assigned with the place node with the highest recognition confidence in the smooth cluster list, which is denoted as $\bar{L}_{mc}^i$.

In the navigation stage, DeepNav calculates a subgroup based on recent recognition results to narrow the localization range. The calculation of subgroup is a 2-stage process. In the first stage, the 'moving center' is updated to the node with the highest valid smooth recognition confidence in the current subgroup. The valid smooth recognition confidence is defined as

$$\bar{L}_{vs}^i = \frac{1}{N} \sum_{k=i-N}^{k} L_{vs}^i(x) * P_m^i, \tag{10}$$

where L_{vs}^j denotes the recognition confidence on recent N valid images. Only when F_{id}^i equals to 1, the recognition confidence of frame i will be added to L_{vs}^j. In the second stage, DeepNav first judges whether the recognition of current frame is valid (i.e., $F_{id}^i = 1$). If it is valid, all nodes within the distance of K from 'moving center' constitute a new subgroup. Then, the place node with the highest recognition confidence in the subgroup is taken as the localization result.

The condition of switching back to relocating status is $P_{vf}^i < \bar{T}_{vf}$, $(K \in (0, 1))$. Since the subgroup sigmoid layer utilizes the context location information and amplifies the recognition confidence of the target node according to the positional relationship among place nodes, it can greatly reduce the number of labels needed to be classified and can filter out incidental recognition errors..

4.2 Moving object filter design

The performance of image-based indoor navigation systems is highly impacted by moving objects, such as pedestrians and temporary advertising boards, since they cause significant noises on the captured images. By fully exploiting the feature extraction ability of CNN, we propose moving object (MO) filter to deal with occlusion problems. In the following, we use pedestrians as an example to analyze the impact of occlusion.

The proposed MO filter is composed of two processing stages, as shown in Fig. 10. In the first stage, an object detector recognizes the input image, locates moving objects and generates a serious of bounding boxes. In the second stage, the image patch placing module replaces the moving objects in the bounding boxes with an image patch similar to the original area. It can confuse the CNN model to perceive the existence of pedestrian and thus increase the place recognition confidence. In this way, we can alleviate the degradation of recognition accuracy caused by the occlusion of moving objects.

In some extreme cases, since the number of moving objects is extremely large causing most of the image features occluded, the confidence improvement brought by the image patch decreases consequently. Therefore, we define a weighted coefficient P_m to suppress the recognition confidence of occluded images. P_m is involved in the calculation process of subgroup sigmoid layer. P_m is defined as the proportion of the image area occupied by

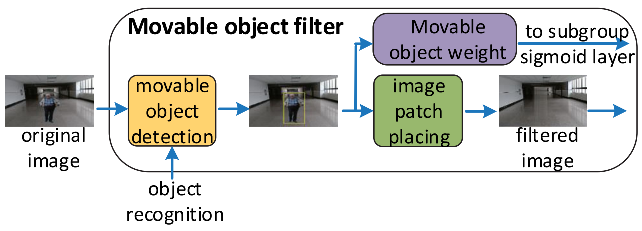


Fig. 10 Moving object filter

the bounding boxes, which is calculated in the first stage as

$$P_m = 1 - \frac{Area_{bbox}}{Area_{image}}. \tag{11}$$

When there are many moving objects, P_m is close to 0, indicating that the recognition result of the current image is very unreliable and the current recognition result would have little effect on the localization. In this way, we can filter out the recognition result suffering from severe occlusion.

In addition, DeepNav can be improved to reduce the CPU and memory usage from two perspectives. First, since the moving objects locating at the image edge have few impacts on the recognition confidence, we can perform object detection on the image center area only. Second, the MO filter is not executed for each frame. Before using the MO filter, we first leverage a MO recognizer to determine whether the image contains moving objects. The MO filter is only used when the recognizer detects that it exactly contains. Since the MO recognizer does not require locating MOs, the computation complexity can be reduced. Meanwhile, in order to share extracted features between the moving object recognizer and the place recognition module, we designed a dual output CNN to recognize both the place and the moving object, as shown in Fig. 11. On the top of the last convolutional layer in CNN model, we train two parallel fully connected layers, one for place recognition and another for moving object recognition. Since the original CNN model is trained on public dataset, it is able to recognize various objects. These two fully connected layers are trained independently. The training data of each MO class consists of two parts, the positive samples and negative samples, with a ratio of 1:1. Positive samples are extracted from the same class of samples in COCO trainval35k, and negative samples are randomly extracted from the rest of the trainval35k.

5 Evaluation

5.1 Test environment

Evaluation scenes We implement a prototype of DeepNav on Android and evaluate its performance in two representative indoor scenarios. The first is a large-scale shopping mall (denoted as SM), which contains rich image features that are helpful for place distinguishing but also contains abundant moving objects. Another scenario is a 4-storey teaching building (denoted as TB), which lacks sufficient image features and has a lot of similar sub-scenes.

Dataset In the TB scene, we collect 3 datasets. The first one is a large-scale dataset containing all possible routes in the building, which is denoted by TB-L. The second dataset is used for simulating occasional navigation and contains specific navigation routes, which is denoted by TB-O. The third dataset is collected under the same route to TB-O, but we ask a volunteer to walk in front of the camera during the data collection to occlude some parts of the image. This dataset is denoted by TB-OV. In the SM scene, we collect one dataset for simulating occasional navigation, named SM-O. Each of these datasets is divided into training sets and test sets. All these datasets are collected by a single pilot. In addition, in order to test the performance of DeepNav from multiple aspects, we also generates some other datasets from above datasets using image processing technique and they will be described in the following experiments.

Device We use a desktop computer for model training and leverage the collaboration of a desktop computer and smartphones for inference. To study the impact of smartphone performance on DeepNav, we use two kinds of smartphones with different computing capabilities in our evaluations. The device specifications are summarized in Table 2.

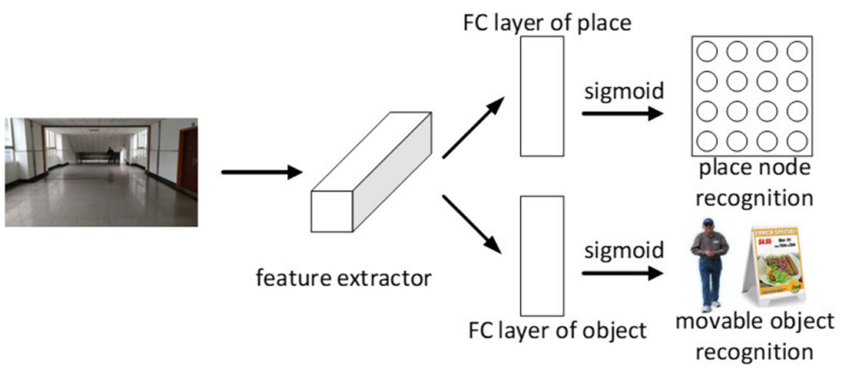


Fig. 11 Dual-output recognition module

Table 2 Device specifications in experiments

Device	Model	Configuration
Training server	Supermicro 7048GR-TR	i7-6700 3.4GHz, 16GB Ram, 128G SSD, Ubuntu 16.04
Smartphone	XiaoMi Max2	Snapdragon 625, 4GB Ram, 128GB Rom, 12 million pixels Camera, Android 7.1
Smartphone	BlackShark	Snapdragon 845, 8GB Ram, 128GB Rom, 20 million pixels Camera, Android 8.1

CNN model We have implemented DeepNav based on several popular CNN models, such as AlexNet [19], VGG-16 [20], InceptionV3 [13] and MobileNet [14]. Based on an evaluation of their performance on smartphones, we finally choose the CNN stacks from MobileNet because it has the best comprehensive performance. All the training processes are conducted on a desktop computer and we adopt TensorFlow [21] as deep learning platform for both the desktop computer and smartphones.

5.2 CNN ablation experiment

5.2.1 CNN backbone comparison

We first evaluate the impacts of different CNN architectures on the accuracy and efficiency of recognition. We adopt the existing object classification CNN as the feature extractor and replace the last fully connected layer with new fully connected layer. Then we freeze the parameters in backbone and only train the last fully connected layer. We make comparisons among 4 popular CNNs, including AlexNet [19], VGG-16 [20], InceptionV3 [13] and MobileNet [14].

We use TB-O dataset in this experiment. The training set and test set of this dataset are collected in the same route but different dates (10 months apart) to ensure the dissimilarity. The video of training set contains 8544 frames that are sampled from 284 place nodes, covering an area of around 2100 square meters. The size of all training batches is 100. The learning rate is initialized to 0.01 and divided by 10 when test error is stable. The training ends at the 50th epoch.

As shown in Table 3, the MobileNet shows the best top-1 recognition accuracy and InceptionV3 shows the best top-5 accuracy. Apart from accuracy, the inference speed and training time are also important in practical system. The comprehensive performance comparisons of the four models are shown in Table 4, where OOM means out of memory. The training is done on PC and inference is done on smartphone. It can be seen that MobileNet shows the best efficiency of top-1 accuracy under constrained training time and inference time. Therefore, we adopt it as the backbone of the recognition model. However, all these four models achieve top-1 accuracy of under 15% and top-5 accuracy of under 70%, which is far from enough to be directly deployed into practical system. To address this issue, we propose the subgroup sigmoid layer to further improve the accuracy.

5.2.2 Place node interval

In DeepNav, we adopt automatic labeling to accelerate map construction. The interval of place nodes causes direct impacts on the map resolution. We define the localization error of recognition result on frame i as

$$A_i = interval * (NodeDist(N_i, N_{gt}) + 1), \qquad (12)$$

where $NodeDist(a, b)$ denotes the distance between node a and b in the map, N_i denotes the place node recognized by the system, N_{gt} denotes the ground truth place node. The total localization error is the mean of all A_i.

To achieve low localization error, we expect small interval and high recognition accuracy. However, if we use smaller interval, the number of labels becomes larger and the number of training samples in one place node becomes smaller, which decreases the recognition accuracy.

To find out the relation between interval and localization error, we evaluate the localization error with different interval settings. We set the place node interval from 1 meter to 10 meters with an increment of 1 meter, then we train 10 CNNs of these interval settings respectively. The results are shown in Fig. 12. The localization error increases with the increasing interval. Although we can derive remarkably high recognition accuracy by using a large interval, the experiment results indicate that the increment of recognition

Table 3 Recognition accuracy of training and testing

Models	Top-1 train	Top-5 train	training time(s)	Top-1 test	Top-5 test
VGG-16	99.8%	100%	2212	10.6%	34.6%
AlexNet	99.9%	100%	626	9.8%	30.3%
InceptionV3	92.7%	100%	1071	11.7%	60.5%
MobileNet	100%	100%	306	13.9%	58.4%

Table 4 Comprehensive performance on smartphone

Device	Models	Inference time (per image)	Top-1 test /training time	Top-1 test /inference time
Max2	VGG-16	OOM	4.8e-5	/
Max2	AlexNet	755ms	1.6e-4	1.3e-4
Max2	InceptionV3	1224ms	1.1e-4	9.6e-5
Max2	MobileNet	270ms	4.5e-4	5.1e-4
Shark	VGG-16	OOM	4.8e-5	/
Shark	AlexNet	254ms	1.6e-4	3.9e-4
Shark	InceptionV3	491ms	1.1e-4	2.4e-4
Shark	MobileNet	98ms	4.5e-4	1.4e-3

accuracy cannot afford the decrement of map resolution. Therefore, the interval should be as small as possible.

It should be noticed that the top-1 recognition accuracy reduces to 13.9% while the interval is 1 meter, this is because the number of training samples of each node is only about 30 and the total node number increases to 284. If we continue decreasing the interval, the number of training samples becomes insufficient, resulting in unstable recognition and aggravation of the similarity problem. Therefore, we set interval to 1 meter finally to achieve the lowest localization error. Due to the involvement of massive image noise, the localization error reaches only 10.8 meters, so we use subgroup sigmoid layer to decrease localization error.

5.2.3 Softmax layer vs sigmoid layer with focal loss

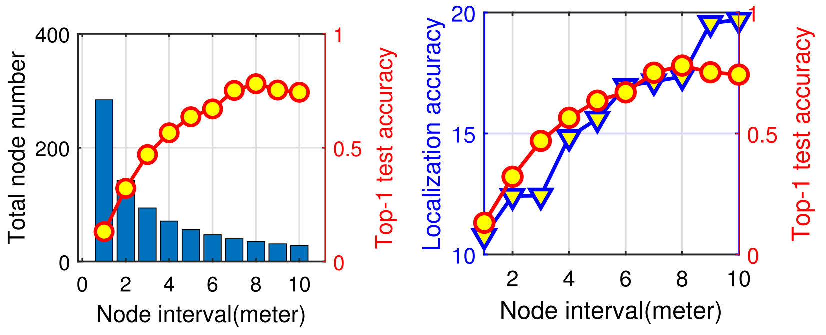
Training time comparison when adding new labels We conduct this experiment using the TB-L dataset. There are totally 1654 place nodes assigned in this dataset and the label number is 1.65 times of the ILSVRC2012 dataset. There are roughly 30 training images corresponding to each label and the number of total images is about 51000. We compare the training time of adding 2 new labels between using softmax layer and sigmoid layer. The result is shown in Table 5. Due to the dependency among the elements in the softmax layer, the whole layer needs to be retrained even when only one label is added. Therefore, the training time should be slightly longer than the initial training time. Because the elements of sigmoid layer are independent, only the newly added elements need to be trained. Therefore, the training time is much shorter. All the training epochs are 50 in this experiment.

From the experiment results, we find that although the number of newly added label is only 0.12% of the original map, the training time of softmax layer is longer than the initial training. After we replace softmax with sigmoid, the training time is decreased by 2/3, indicating that the use of sigmoid layer dramatically decreases the training time when adding new labels. We should also notice that the difference of training time is getting larger with the increment of building scale and place node number.

Comparison of recognition accuracy In this experiment, we study the improvement of the recognition accuracy and the unbalance of training samples after replacing softmax with sigmoid. All the training epochs are 50 and the learning rate is fixed to 0.01. The results are shown in Table 6.

First, we evaluate the original recognition accuracy of softmax layer. The top5 accuracy is only 58.4%. The reason is that the number of training samples is insufficient and the number of labels is too large, resulting in overfitting.

Fig. 12 Impacts of interval



(a) Impacts of interval to recognition accuracy (b) Impacts of interval to localization error

Table 5 Comparison of training time between softmax and sigmoid

Layer	Operation	Training time
softmax	initial train	1744s
softmax	adding 2 new labels	1757s
sigmoid	adding 2 new labels	598s

After replacing softmax with sigmoid, we set the two hyper-parameters, i.e., α and γ, in the focal loss function [18] to 0.5 and 0, respectively, which are equivalent to not using the focal loss function. The accuracy is lower than softmax. The reason is that the unbalanced number of positive and negative samples makes the training worse. Then, we adjust α and γ separately, and the accuracy is improved by 3.4% and 9.2%, where the two hyper-parameters are influenced mutually. We can obtain the best improvement by setting them to 0.9 and 5.0, respectively. The accuracy is increased by 8.5% compared to softmax and 9.5% compared to the original sigmoid.

5.3 Impacts of patch style

By leveraging prior knowledge and context information, we design and compare 3 kinds of patch styles. The gray patch is pure gray. The average patch averages the colors around the MOs. In the gradual patch, we adopt the linear interpolation approach.

In order to investigate the improvement of image patch with different sizes and places, we use synthetic images by synthesizing real place images and real image patches. For each original image, we place pedestrian patches, gray patches, average patches and gradual patches with 10 different sizes in 100 locations. Totally, we obtain 4000 synthetic images corresponding to each original image. Then, we choose 10 place images with distinct appearance to verify the generalization ability of MO filter. The pre-trained CNN is used to inference these images and compare the recognition confidence.

As shown in Fig. 13, the filter effect of image patches is obvious on small moving objects. Over 90% of the confidence degradation is improved. Among these three patch styles, gradual patch has the best performance. The average patch is slightly better than the gray patch, which shows that CNN is sensitive to lines and edges and we should make the patch as smooth as possible.

The mean confidence improvement proportions of gray patch, average patch and gradual patch are 31.18%, 41.29%, 76.71%, respectively, indicating that if we can precisely locate pedestrian, the MO filter would filter the MOs effectively.

Table 6 Comparison of recognition accuracy between softmax and sigmoid

Layer	α	γ	Top-5 accuracy
softmax	/	/	58.4
sigmoid	0.5	0	57.4
sigmoid	0.9	0	60.8
sigmoid	0.5	6.0	66.3
sigmoid	0.9	6.0	66.6
sigmoid	0.9	5.0	66.9

5.4 Ablation experiment

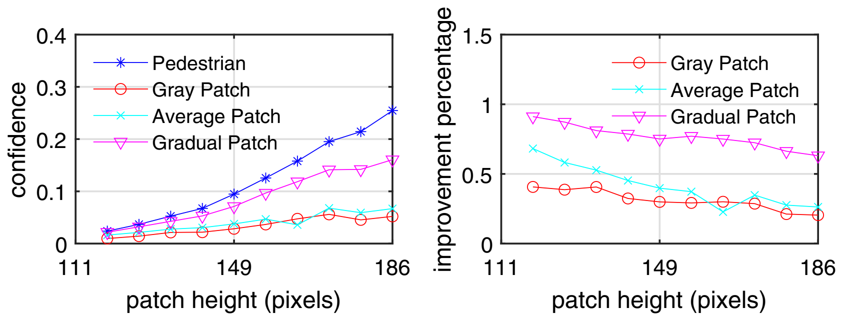
In this experiment, we evaluate the improvement of subgroup sigmoid layer and MOF on the localization error under three datasets. The first dataset is TB-O, which does not contain any movable object (NMO). The second dataset is synthesized based on TB-O and contains synthetic moving objects (SMO). The third dataset is TB-OV and contains the images with true moving objects (TMO). The experimental results on recognition accuracy and localization error are shown in Table 7.

In the NMO test, the top-1 recognition accuracy and localization of original softmax are only 0.127 and 11.32 respectively. After replacing it with subgroup sigmoid layer, the top-1 recognition accuracy is increased by 0.016 and localization error is decreased by 8.779, indicating that the subgroup sigmoid layer effectively alleviates recognition difficulties brought by massive labels and the lack of training data. In addition, we can find that MOF almost has no effect on the situation with no movable objects, which is consistent to our expectation.

In the SMO test, the recognition accuracy decreases and localization error of the original softmax increases significantly, indicating that the synthetic moving objects greatly impact the recognition accuracy of original softmax. After introducing the MOF, the localization error is decreased by 55.5%. After replacing the original softmax with subgroup sigmoid layer, the localization error is decreased by 85.0%. When combining subgroup sigmoid layer and MOF, the localization error is decreased by 94.7%. It demonstrates that both of the MOF and subgroup sigmoid layer have good performance on filtering moving objects. And their effects show in different aspects, so we can obtain the best performance when using them together.

In the TMO test, the results are similar to that in the SMO test. The localization error of using original softmax is 26.992m. After replacing it with the subgroup sigmoid layer and using MOF, the localization error is decreased by 77.9% and 27.3%, respectively. When using subgroup sigmoid layer and MOF together, we decrease the localization error by 91.4%, down to 2.3m.

Fig. 13 Improvement of different patch styles

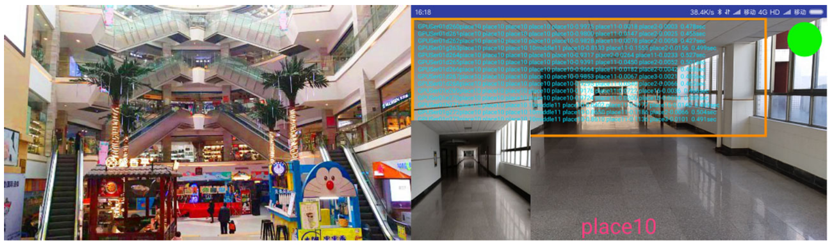


(a) Confidence improvement of different patch style (b) Improvement proportion of different patch style

Table 7 Ablation experiment results

	NMO test		SMO test		TMO test	
	top-1	loc-err	top-1	loc-err	top-1	loc-err
o-sm	0.127	11.324	0.095	47.988	0.162	26.992
sg-sm	0.143	2.545	0.150	7.202	0.175	5.976
o-sm+ MOF	0.127	11.590	0.102	21.356	0.193	19.628
sg-sm+ MOF	0.144	2.560	0.162	2.534	0.221	2.333

Fig. 14 Demonstration and the user interface of DeepNav



(a) Snapshot of the test environment (b) Snapshot of DeepNav

Table 8 Deployment time comparison in various scenes

Methods	Scene	T_c	T_l	T_u	T_t	T_{total}
DeepNav	SM-O	284s	1320s	139s	320s	2063s
DeepNav	TB-O	285s	900s	140s	306s	1631s
DeepNav	TB-L	1860s	4200s	911s	1744s	8715s
ViNav [10]	TB-L	/	/	/	/	$\sim$ 10 days

Table 9 Deployment parameters

r_v	r_u	n_f	T_t	t_l
0.245MB/s	0.5MB/s	30Hz	0.036sec	60sec

5.5 Deployment speed

We also evaluate the deployment time of DeepNav in the scenes with different complexity. The total deployment time is mainly composed of 4 parts: data collection time, data labeling time, video uploading time and training time, denoted as T_c, T_l, T_u and T_t, respectively. The total deployment time T_{total} could be roughly calculated by summing up these 4 parts. T_c roughly equals the video length. T_l, T_u and T_t could be calculated by the equations below.

$$\begin{aligned} T_l &= n_{cr} * t_l; \\ T_u &= T_{co} * r_v / r_u; \\ T_t &= T_{co} * r_f * T_t, \end{aligned}$$

where n_{cr} denotes the number of cross place nodes in the scene, r_v denotes the video data rate, r_u denotes uploading rate, r_f denotes frame rate and T_t denotes training time amortized to one frame. The parameter settings are listed in Table 9.

The first scene is a 4-storey shopping mall, as shown in Fig. 14(a). In this scene we deploy occasional navigation service, denoted by SM-O. The video length is about 5 minutes, the training frames are 8518 divided into 284 place nodes and the sub-route number is 22. The second scene is a 4-storey teaching building, where we deploy both occasional and large-scale navigation services, denoted by TB-O and TB-L, respectively. In TB-O, the video length is about 5 minutes, the training frames are 8544 divided into 285 place nodes, and the sub-route number is 10. In TB-L, the video length is 31 minutes, the training frames are about 51000 divided into 1653 place nodes, and the sub-route number is 70. In these scenes, the navigation deployment is finished by a single pilot. The deployment time of these scenes is shown in Table 8, and the deployment time proportion is shown in Fig. 15.

For occasional navigation services, the total deployment time is basically depending on the length of routes. In these two different scenes (SM-O and TB-O), the total length of each route is close to 300 meters and the total deployment time is less than an hour. For large-scale navigation services, the total deployment time increases by the increment of scale. And the data labeling time is substantially increased because of the increment of sub-routes. Although the scale is large, the total deployment time is less than 3 hours (Table 9).

We also compare the deployment time to other recent indoor navigation system. According to the description in ViNav [10], we estimate its deployment time in the same scene. In order to achieve the localization accuracy of 1 meter and resolution of 6 degrees, there are at least 60 images needed to be collected at each node and the interval between two nodes is at least 1 meter. Since the area of each floor of the test building is about 2100 square meters, there are about 504000 images needed to be collected in total. Since they use traditional image collection method, which takes picture one by one. The image collection speed is roughly 2/second. Therefore, the total collection time is estimated to be 70 hours. However, under on pilot situation, he needs rest time to share the 70 hours to 8-10 days with over 8 hours per day. Considering the post-processing process, the total deployment time is conservatively estimated to be 10 days.

5.6 Navigation time consumption

We evaluate the time consumption of navigation in the TB-O scene. We set multiple start points in the whole navigation route and arrange volunteers to start navigation from these points. The navigation route and the start points are shown in Fig. 16. The route spans from the first floor to the fourth floor and contains multiple cross place nodes.

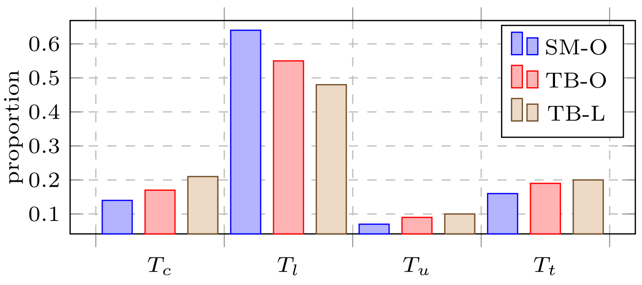


Fig. 15 Navigation time consumption proportion in different scenes

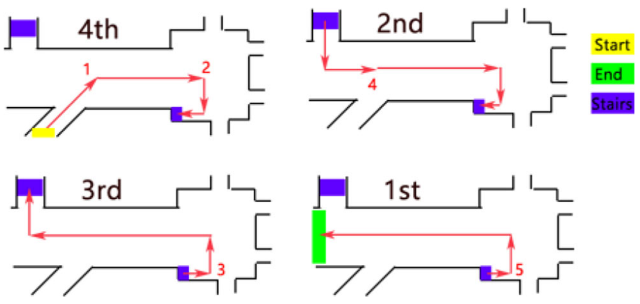


Fig. 16 The route map of the TB-O test

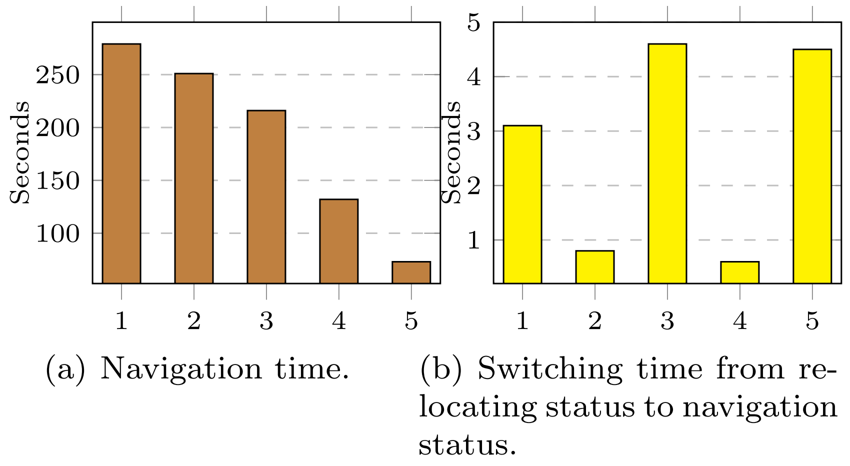


Fig. 17 Navigation time consumption starting from different places

When the volunteer is located at one of the start points, DeepNav first enters relocating status and starts locating. When it satisfies the switching condition, DeepNav enters navigation status and the volunteers follow the instructions until they arrive the end point. The snapshot of DeepNav is shown in Fig. 14b.

The experimental results are shown in Fig. 17. Although these start points are different from those collected by the pilots, all the volunteers successfully switch from relocating status to navigation status and arrive at the end point by following the navigation instructions. The time consumption of switching to navigation status is different in different start points. The reason is that DeepNav switches to navigation status only when it satisfies the switching condition. The switching is fast in the start points which locates in the straight corridor, such as start point 1 and 4, because the straight corridor contains a lot of similar place nodes and it is easier to satisfy the switching condition. The switching time of point 2 is also short, since this location contains abundant visual feature (Fig. 18).

We also compare the cumulative distribution of recognition results in the shopping mall and teaching building. This distribution represents the difficulties of locating. Although the environment of shopping mall is much more complicated than teaching building, the results show that the cumulative distribution of recognition results in shopping mall is lower than teaching building. It demonstrates that abundant visual features in shopping mall makes it easier to be located than teaching building.

5.7 Deviation detection test

In this section, we evaluate the deviation detection ability of DeepNav. In this experiment, volunteers start navigation from the origin in the above scene and deviate intentionally in the navigation process. When the system switches from navigation status to the relocating status, it successfully detects the deviation and starts relocating. As shown in Fig. 19, all the deviations are detected within 6 seconds, and 65% of the deviations are detected within 3 seconds.

6 Related works

With the popularization of smartphones and the urgent need for indoor navigation services, various kinds of localization and navigation methods for smartphones have been proposed in recent years. Wi-Fi fingerprint is one of the most widely studied technique [1, 22–25], due to its advantages of low power consumption and high localization accuracy. However, the deployment process of these methods is complicated. They require to derive the building floor maps and correspond the position of

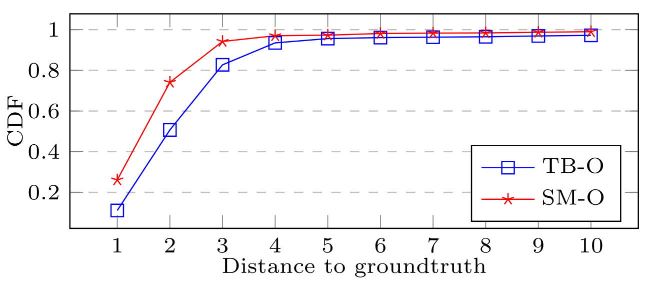


Fig. 18 Distribution of distance to groundtruth in different scenes

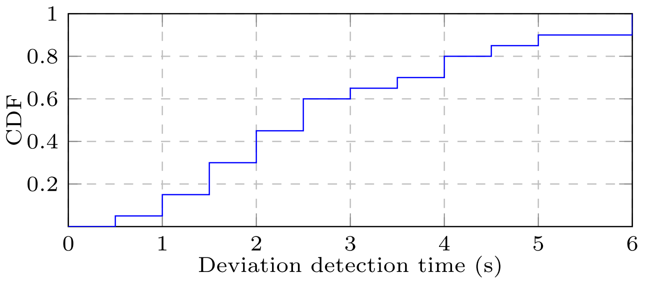


Fig. 19 Detection time of deviation

WiFi devices to the floor maps. Additionally, they need to do some post-processing algorithm to make the detection robust, making it difficult to realize fast deployment. Since DeepNav does not require floor maps and the data processing is most done automatically, it is much easier to be deployed in a fast speed.

By using vision-based technologies, SLAM is also a representative indoor localization and navigation method [9, 10], which shows good performance in feature-rich areas. However, this kind of methods relies on crowdsourcing to collect large-scale image data, resulting in low deployment speed. Another method using crowdsourcing is landmark and dead reckoning [3–8], which realizes complete localization by using the sensors on smartphones. However, these methods require a large amount of data collected by crowdsourcing, which consequently slows the deployment speed. According to ViNav [10], in a building of 2100 square meters, there are about 504000 images needed to be collected in total, which may take a couple of weeks to complete. Comparatively, DeepNav does not involve crowdsourcing and all the deployment work can be completed by only a single pilot.

Magnetic field fingerprinting is also a popular method [11, 26–28]. However, due to the characteristics of the magnetic field itself, the magnetic field fingerprint can only be used as an auxiliary localization method. The absolute localization can not be realized independently, thereby increasing the complexity of the localization system. Leveraging deep learning for indoor navigation is an emerging research trend [2, 29]. Researchers combine deep learning with classic methods, such as viewing WiFi fingerprint and magnetic fingerprint as images, and then use image recognition methods for localization. Other works using deep learning model focus on robotic autonomous navigation [30, 31], which require the support of dedicated hardware and sensors. Another research trend is to mix a variety of different methods [1, 32–37] to obtain higher localization accuracy, but they increase the complexity of the system at the same time. In contrast, DeepNav uses camera alone, which is relatively simple and relaxes the hardware requirements on the mobile device.

In addition, there are also some other methods, including RFID [38], IMU [39], Beacons [40], acoustic signal [41, 42] and even light [43]. These methods are still in the early stage of research, and they often rely on dedicated infrastructures or devices. For instance, RFID based method [38] relies on dedicated coil inductor, while beacons based method [40] has to place dedicated beacons around the navigation area and IMU based method [39] builds up a model only suitable for smartwatch. However, DeepNav does not rely on any other devices other than smartphone.

7 Conclusion

In this paper, we propose DeepNav, a fast deployable indoor navigation system that fully exploits the capability of visual CNN and works in a pilot-follower way. DeepNav enables the pilots to collect data in an easy-to-use way to solve the data redundancy problem. In order to solve the challenges caused by insufficient training data and invalid image recognition, we propose the subgroup sigmoid layer, which can effectively improve the localization accuracy in noisy environment. We also propose the movable object filter to relieve the degradation of localization accuracy caused by occlusion. We implement DeepNav on Android smartphones and evaluate its performance in a shopping mall and a teaching building. The experimental results demonstrate that DeepNav can achieve the minimum localization error of 2.3m with pedestrian occlusion. In the future, we plan to introduce image semantic segmentation in DeepNav to further improve the navigation performance.

References

1. Zheng Y, Shen G, Li L, Zhao C, Li M, Zhao F (2017) Travinavi: Self-deployable indoor navigation system. IEEE/ACM Trans Netw 25(5):2655
2. Wang J, Zhang X, Gao Q, Yue H, Wang H (2017) Device-free wireless localization and activity recognition: A deep learning approach. IEEE Trans Veh Technol PP(99):1
3. Abdelnasser H, Mohamed R, Elgohary A, Alzantot M, Wang H, Sen S, Choudhury RR, Youssef M (2016) SemanticSLAM: Using environment landmarks for unsupervised indoor localization. IEEE Trans Mob Comput 15(7):1770
4. Alzantot M, Youssef M (2012) Crowdinside: automatic construction of indoor floorplans. In: Proceedings of the 20th international conference on advances in geographic information systems. ACM, pp 99–108
5. Yang Z, Wu C, Liu Y (2012) Locating in fingerprint space: wireless indoor localization with little human intervention. In: Proceedings of the 18th annual international conference on mobile computing and networking. ACM, pp 269–280
6. Chen S, Li M, Ren K, Fu X, Qiao C (2015) Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing. In: Proceedings of the 13th ACM conference on embedded networked sensor systems. ACM, pp 59–71
7. Gao R, Zhao M, Ye T, Ye F, Wang Y, Bian K, Wang T, Li X (2014) Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In: Proceedings of the 20th annual international conference on Mobile computing and networking. ACM, pp 249–260
8. Chen S, Li M, Ren K, Qiao C (2015) Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos. In: 2015 IEEE 35th International conference on distributed computing systems. IEEE, pp 1–10
9. Mur-Artal R, Montiel JMM, Tardos JD (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot 31(5):1147
10. Dong J, Noreikis M, XIAO Y, Yla-Jaaski A (2018) ViNav: A vision-based indoor navigation system for smartphones. IEEE Trans Mob Comput

11. Shu Y, Kang GS, He T, Chen J (2015) Last-mile navigation using smartphones. In: International conference on mobile computing and networking, pp 512–524
12. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
13. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
14. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861
15. Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, et al. (2017) Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE CVPR, vol 4
16. Ma N, Zhang X, Zheng HT, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. arXiv:1807.11164
17. Wei Y, Xia W, Lin M, Huang J, Ni B, Dong J, Zhao Y, Yan S (2015) HCP: A flexible CNN framework for multi-label image classification. IEEE Trans Pattern Anal Mach Intell 38(9):1901
18. Lin TY, Goyal P, Girshick R, He K, Dollár P. (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
19. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
20. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
21. Google (2018) Tensorflow. https://www.tensorflow.org/
22. Yin Z, Wu C, Yang Z, Liu Y (2017) Peer-to-peer indoor navigation using smartphones. IEEE J Select Areas Commun PP(99):1
23. Yin Z, Wu C, Yang Z, Lane N, Liu Y (2017) ppNav: Peer-to-peer indoor navigation for smartphones. In: IEEE international conference on parallel and distributed systems, pp 104–111
24. Zhuang Y, Syed Z, Li Y, Elsheimy N (2016) Evaluation of two wifi positioning systems based on autonomous crowd sourcing on handheld devices for indoor navigation. IEEE Trans Mob Comput 15(8):1982
25. Wang X, Gao L, Mao S, Pandey S (2017) CSI-Based fingerprinting for indoor localization: a deep learning approach. IEEE Trans Veh Technol 66(1):763
26. Zhang C, Subbu KP, Luo J, Wu J (2015) GROPING: Geomagnetism and crowdsensing powered indoor navigation. IEEE Trans Mob Comput 14(2):387
27. He S, Kang GS (2017) Geomagnetism for smartphone-based indoor localization: challenges, advances, and comparisons. Acm Comput Surv 50(6):1
28. Li Z, Shu Y, Karlsson BF, Lin Y, Moscibroda T (2017) Demo: Towards flexible and scalable indoor navigation. In: International conference on mobile computing and networking, pp 495–497
29. Liu Z, Zhang L, Liu Q, Yin Y, Cheng L, Zimmermann R (2017) Fusion of magnetic and visual sensors for indoor localization: infrastructure-free and more effective. IEEE Trans Multimed 19(4):874
30. Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L, Farhadi A (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3357–3364
31. Kahn G, Villaflor A, Ding B, Abbeel P, Levine S (2018) Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In: 2018 IEEE International conference on robotics and automation (ICRA). IEEE, pp 1–8
32. Shu Y, Bo C, Shen G, Zhao C, Li L, Zhao F (2015) Magicol: Indoor localization using pervasive magnetic field and opportunistic wifi sensing. IEEE J Select Areas Commun 33(7):1443
33. Li Y, Zhuang Y, Zhang P, Lan H, Niu X, El-Sheimy N (2017) An improved inertial/wifi/magnetic fusion structure for indoor navigation. Inf Fusion 34(C):101
34. Teng X, Guo D, Zhou X, Liu Z (2015) Poster: An indoor-outdoor navigation service for subway transportation systems. In: ACM Conference on embedded networked sensor systems, pp 415–416
35. Wu FJ (2018) A sensor-assisted emergency guiding system: sensor-centric or user-centric? IEEE Trans Veh Technol 67(2):1598
36. Zhuang Y, Yang J, Qi L, Li Y, Cao Y, El-Sheimy N (2017) A pervasive integration platform of low-cost MEMS sensors and wireless signals for indoor localization. IEEE Internet Things J PP(99):1
37. Atia MM, Liu S, Nematallah H, Karamat TB, Noureldin A (2015) Integrated indoor navigation system for ground vehicles with automatic 3-d alignment and position initialization. IEEE Trans Veh Technol 64(4):1279
38. Tsirmpas C, Rompas A, Fokou O, Koutsouris D (2015) An indoor navigation system for visually impaired and elderly people based on Radio Frequency Identification (RFID). Inform Sci 320(C):288
39. Jiang Y, Li Z, Wang J (2017) PTrack: Enhancing the applicability of pedestrian tracking with wearables. In: IEEE international conference on distributed computing systems, pp 2193–2199
40. Xiang L, Tai TY, Li B, Li B (2017) Tack: learning towards contextual and ephemeral indoor localization with crowdsourcing. IEEE J Select Areas Commun PP(99):1
41. Liu K, Wu D, Li X (2016) Enhancing smartphone indoor localization via opportunistic sensing. In: IEEE International conference on sensing, communication, and networking, pp 1–9
42. Huang W, Xiong Y, Li XY, Lin H, Mao X, Yang P, Liu Y, Wang X (2015) Swadloon: Direction finding and indoor localization using acoustic signal by shaking smartphones. IEEE Trans Mob Comput 14(10):2145
43. Zhao Z, Wang J, Zhao X, Peng C, Guo Q, Wu B (2017) NaviLight: Indoor localization and navigation under arbitrary lights. In: INFOCOM 2017 - IEEE conference on computer communications IEEE

Jian Gong received the B.Sc. (2014) and M.Sc. (2017) degrees in electrical engineering, from Central South University, China. Currently, he is an associated Ph.D in computer science, University of California, San Diego. His research interests include multi-sensor fusion, deep learning, and mobile computing.

Ju Ren received the B.Sc. (2009), M.Sc. (2012), Ph.D. (2016) degrees all in computer science, from Central South University, China. Currently, he is a professor with the School of Computer Science and Engineering, Central South University, China. His research interests include Internet-of-Things, wireless communication, network computing and cloud computing. He received many best paper awards from IEEE flagship conferences, including IEEE ICC'19, HPCC'19 and IoP'18, etc. He currently serves/has served as an associate editor for IEEE Transactions on Vehicular Technology and Peer-to-Peer Networking and Applications, a guest editor for IEEE Wireless Communications, IEEE Transactions on Industrial Informatics, and IEEE Network. He also served as the TPC chair of IEEE BigDataSE'19, a poster co-chair of IEEE MASS'18, a track co-chair for IEEE/CIC ICCC'19, IEEE I-SPAN'18 and VTC'17 Fall, and TPC member for many IEEE/ACM top-tier conferences. He is a member of IEEE and ACM.

Yaoxue Zhang received his B.Sc. degree from Northwest Institute of Telecommunication Engineering, China, in 1982, and his Ph.D. degree in computer networking from Tohoku University, Japan, in 1989. He is a professor with the Department of Computer Science and Technology, Tsinghua University, China and also a professor with the School of Computer Science and Engineering, Central South University, China. His research interests include computer networking, operating systems, ubiquitous/pervasive computing, transparent computing, and big data. He has published over 200 technical papers in international journals and conferences, as well as 9 monographs and text-books. Currently, he is serving as the Editor-in-Chief of Chinese Journal of Electronics. He is a fellow of the Chinese Academy of Engineering.