Smartphone-Based Indoor Visual Navigation with Leader-Follower Mode

JINGAO XU, ERQUN DONG, and QIANG MA, School of Software and BNRist, Tsinghua University, People's Republic of China
CHENSHU WU, Department of Electrical & Computer Engineering, University of Maryland, College Park, USA
ZHENG YANG, School of Software and BNRist, Tsinghua University, People's Republic of China

Existing indoor navigation solutions usually require pre-deployed comprehensive location services with precise indoor maps and, more importantly, all rely on dedicatedly installed or existing infrastructure. In this article, we present Pair-Navi, an infrastructure-free indoor navigation system that circumvents all these requirements by reusing a previous traveler's (i.e., leader) trace experience to navigate future users (i.e., followers) in a Peer-to-Peer mode. Our system leverages the advances of visual **simultaneous localization and mapping** (**SLAM**) on commercial smartphones. Visual SLAM systems, however, are vulnerable to environmental dynamics in the precision and robustness and involve intensive computation that prohibits real-time applications. To combat environmental changes, we propose to cull non-rigid contexts and keep only the static and rigid contents in use. To enable real-time navigation on mobiles, we decouple and reorganize the highly coupled SLAM modules for leaders and followers. We implement Pair-Navi on commodity smartphones and validate its performance in three diverse buildings and two standard datasets (TUM and KITTI). Our results show that Pair-Navi achieves an immediate navigation success rate of 98.6%, which maintains as 83.4% even after 2 weeks since the leaders' traces were collected, outperforming the state-of-the-art solutions by >50%. Being truly infrastructure-free, Pair-Navi sheds lights on practical indoor navigations for mobile users.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing;**

Additional Key Words and Phrases: Indoor navigation, computer vision, visual SLAM

A preliminary version of this article appeared in International Conference on Computer Communications (IEEE INFOCOM 2019).
This work is supported in part by the National Key R&D Program of China under grant no. 2018AAA0101200, and the NSFC under grants no. 61832010, no. 61632008, no. 61872081, no. 61632013, and no. 61972131.
Authors' addresses: J. Xu, E. Dong, Q. Ma, and Z. Yang (corresponding author), School of Software and BNRist, Tsinghua University, 30 Shuangqing Road, Haidian District, Beijing, China, 100084; emails: xujingao13@gmail.com, doneq13@gmail.com, thumq@mail.tsinghua.edu.cn, hmilyyz@gmail.com. C. Wu, Deparment of Electrical & Computer Engineering, University of Maryland, College Park, Washington DC, MD 20742; email: wucs32@gmail.com.

1550-4859/2021/05-ART18 $15.00
https://doi.org/10.1145/3448417

ACM Reference format:
Jingao Xu, Erqun Dong, Qiang Ma, Chenshu Wu, and Zheng Yang. 2021. Smartphone-Based Indoor Visual Navigation with Leader-Follower Mode. *ACM Trans. Sen. Netw.* 17, 2, Article 18 (May 2021), 22 pages.
https://doi.org/10.1145/3448417

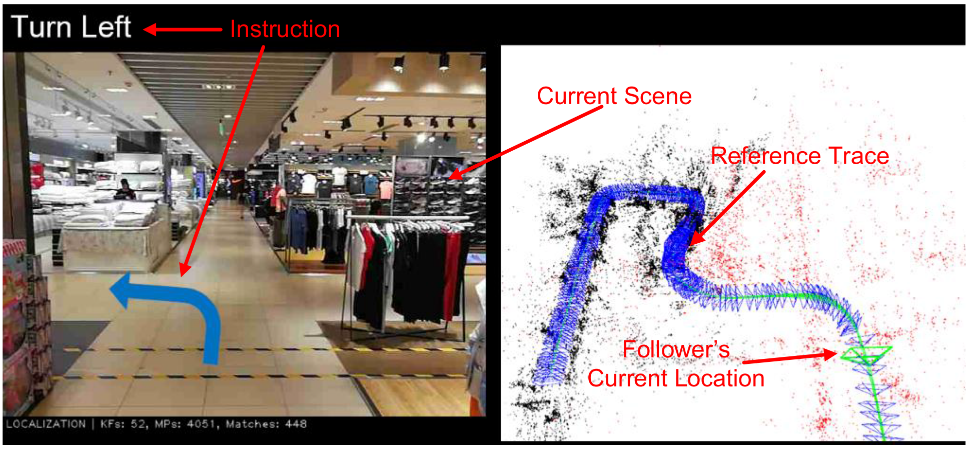
1 INTRODUCTION

During the past decades, technologies using Wi-Fi [33, 60–62, 66, 71], RFID [49, 58], sound [4], visible lights [34], and so forth, have been proposed to shape a range of location-based services. Therein, indoor navigation with a smartphone acts as a killer application [30, 56]. All conventional navigation techniques, however, require particular infrastructure, either pre-existing or dedicatedly installed, to be appropriately set up in advance in the area-of-interests. Recently, an alternative **Peer-to-Peer** (**P2P**) navigation is proposed to circumvent the pre-installation of indoor localization services [14, 69]. In this mode, a previous traveler, named *leader*, records the trace information (e.g., turnings and certain ambient properties) and shares it through the Internet to a later *follower*, who needs to travel to the same destination. A typical example would be a self-deployed navigation service to direct a customer to a shop, which enables a shop owner to record such trace information from the entrance of a large mall to his/her own shop and offer them to potential visitors as guidance, without resorting to any pre-deployed location systems provided by third parties.

Several pioneer works have demonstrated such a leader-follower mode for P2P navigation [51, 69, 70]. These works mainly leverage ambient Wi-Fi signals, in addition to inertial sensor measurements and/or images captured by smartphone [15, 64, 67], as trace properties to synchronize leaders' and followers' traces. Although these works are inspiring, the pre-installation of Wi-Fi and the error introduced by the dynamically changing nature of Wi-Fi signal and the inertial sensors make them less than ideal for practical usage.

Recently, two arising trends may overcome the above limitations and underpin a practical solution to indoor navigation. First, **simultaneous localization and mapping** (**SLAM**) technology has been rapidly developed. For example, visual SLAM has been enabled with a single camera [13, 27, 36], making it feasible on commodity smartphones that usually have only one camera on the back side. Second, vision capability has become a standard and continues growing more powerful on mobile devices, allowing advanced vision tasks on mobiles.

In this work, we investigate visual SLAM with the power of mobile vision and present Pair-Navi, a P2P indoor navigation system that requires no pre-existing or dedicatedly installed infrastructure, pre-deployed localization service, or indoor digital maps. Visual SLAM utilizes one or more cameras to explore an unknown environment by continuously locating the camera itself in the environment and meanwhile constructing a map of the environment [36]. Our approach is built upon monocular visual SLAM with a single camera commonly equipped on commodity smartphones. A leader of Pair-Navi simply walks through a path recording a video clip along the route and shares the trajectory video via a cloud server for potential upcoming followers. Pair-Navi consumes the video for SLAM and constructs the trajectory that the leader has traveled. When a follower appears, he/she will be provided with a leader's trajectory as reference. On the follower side, Pair-Navi also captures real-time video frames and precisely locates the follower's relative location to the reference trajectory, accordingly navigating the follower by timely promoting walking hints. As shown in Figure 1, both the current scene and the reference trace will be displayed to the follower. In addition, Pair-Navi handles navigation deviation, which is necessary but neglected by existing P2P navigation approaches. Without the need to instrument the building-of-interests, Pair-Navi works in any scenes as long as a camera-equipped smartphone is available.



LOCALIZATION | KFs: 52, MPs: 4051, Matches: 448

Fig. 1. Follower's navigation interface of Pair-Navi.

However, translating visual SLAM into a robust navigation system entails various challenges:

— *Environmental Non-Rigidity*. Most popular SLAM solutions assume rigid and static indoor environments, where the surrounding scenes are not supposed to change both when the SLAM is running and after the map is constructed [7, 27, 36]. However, the real world is time-varying due to considerable dynamics, e.g., pedestrians, furniture changes, advertising screens, the inherent object deformation, and lighting condition variations, rendering the constructed trajectories inaccurate and difficult to follow. As shown in Figure 2, such environmental non-rigidity will cause errors in video frame matching and thus significantly degrading visual SLAM. Although some SLAM approaches attempt to reason about minimal non-rigidity with restrictive applicability [36, 38], it still remains challenging to employ visual SLAM for mobile indoor navigation in vibrant scenarios full of dynamics, such as busy shopping malls, large airports, and so forth.

— *Real-time*. Visual SLAM technologies typically require intense computation for several core tasks including visual odometry and optimization, making them difficult to run in real-time on commodity mobiles. A practical navigation application, however, should locate the user precisely, render the navigation path, and provide user-friendly instructions, all in real-time. Applying visual SLAM to real-time mobile navigation is a non-trivial task that calls for significant efforts in system design and implementation.

To combat environmental non-rigidity, we propose to extract and subtract the dynamic foregrounds, e.g., pedestrians and other changing contents involved in the trajectory video, and keep only the rigid parts for SLAM. The key observations are twofold: (1) Video frames of typical indoor environments usually provide abundant features for SLAM, allowing room to sift out non-rigid contexts while keeping as good or even better performance since the remaining features are mainly from those rigid and reliable objects. (2) Recent progress in computer vision, especially with the application of deep learning, make it feasible for efficient and effective detection and segmentation of non-rigid dynamic contexts inside a video [22, 25, 45]. Based on these two insights, we employ **Mask Region-based CNN** (**Mask R-CNN**) [25] to identify non-rigid objects and cull them from the video feature set used for SLAM. By doing so, Pair-Navi eliminates the impacts of the environmental dynamics, thus retaining robust trajectories for leaders as well as precise locations for followers.

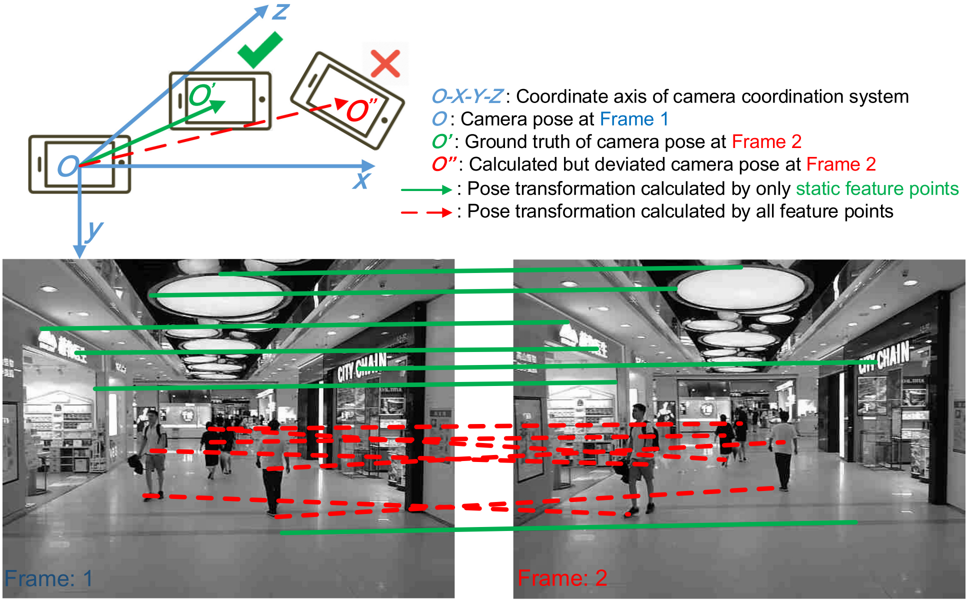

z
x
y
O
O'
O''
O-X-Y-Z: Coordinate axis of camera coordination system
O: Camera pose at Frame 1
O': Ground truth of camera pose at Frame 2
O'': Calculated but deviated camera pose at Frame 2
: Pose transformation calculated by only static feature points
: Pose transformation calculated by all feature points
Frame: 1
Frame: 2

Fig. 2. Illustration of feature point matching of two consecutive frames. Feature points from rigid contexts are matched correctly (solid green lines), while most of feature points from non-rigid contexts are mismatched (dashed red lines), resulting in errors in camera pose calculation.

To enable real-time navigation on smartphones, Pair-Navi decouples the originally coupled SLAM modules and reassembles merely the necessary modules. For a follower, rather than employing a complete SLAM system, we only conduct relocalization to synchronize his/her relative walking progress to a leader's trajectory. Furthermore, we employ a synchronization strategy for the basic visual navigation module and non-rigid context culling module. By doing so, Pair-Navi achieves real-time navigation on a mobile. In contrast, the latest works [6, 8, 48] that attempted to incorporate semantics for robust SLAM fail to operate in real-time.

We implement our system on the **Robot Operating System** (**ROS**) platform [2] on the server and on ROS-Android [1] on the phone side. Comprehensive experiments are carried out in three buildings with various conditions over 2 weeks. The results demonstrate that Pair-Navi achieves a remarkable navigation success rate of 98.6%. Even after 2 weeks since the construction of the leaders' trajectory, the rate maintains 83.4%, outperforming the state-of-the-art *Travi-Navi* [70] by 50.9% and *FollowMe* [51] by 80.4%. We further evaluate the efficiency of **Non-Rigid Context Culling** (**NRCC**) in two official datasets: TUM [54] and KITTI [20]. Both the localization and the mapping accuracy are increased by more than 25%. Being truly infrastructure-free, Pair-Navi takes an important step toward practical indoor navigation for mobile users.

In summary, the core contributions are as follows.

- We present the first user-friendly vision-based P2P navigation system, which neither requires to instrument a building nor relies on pre-deployed localization service with indoor maps.
- We employ non-rigid context culling by using Mask-RCNN to overcome indoor environmental dynamics for visual SLAM, which significantly improves the robustness and precision of navigation. Extensive experiments on two standard datasets demonstrate the leverage of NRCC improves the localization accuracy and mapping precision.

— We implement a complete real-time system on commodity smartphones and extensively evaluate the performance. The results show Pair-Navi achieves delightful results and outperforms all existing solutions.

The rest of article is organized as follows. We present an overview in Section 2 and introduce visual navigation in non-rigid environment in Section 3. Real-time design and implementation is provided in Section 4, followed by experiments in Section 5. We review related works in Section 6 and conclude in Section 7.

2 OVERVIEW

2.1 Peer-to-Peer Navigation

Different from conventional navigation systems that rely on pre-deployed localization services, Pair-Navi works in an easy-to-deploy P2P navigation mode. P2P navigation also circumvents the need of indoor digital maps, which are sometimes difficult to obtain and process. There are two key roles in a P2P navigation system, i.e., a leader and a follower. The basic idea is to reuse the experience from earlier travelers who become leaders. Anyone walked through a path can serve as a leader for that particular path by contributing corresponding trace information, i.e., certain trace data (e.g., Wi-Fi signal series, geomagnetic series, and IMU sensor measurements [51, 69, 70], or video clips in our case) together with the automatically extracted walking hints (e.g., heading, turning, climbing). The trace information is later requested by and sent to a follower for his/her reference. The navigation for the follower is then achieved by synchronizing his/her relative location to a leader's reference trajectory. Note that a user can participate as either a leader or a follower, depending on specific scenarios.

P2P navigation is, in particular, useful as a fast- and easy-to-deploy service for ordinary users who demand to provide small-scale navigation. For example, a shop owner can provide a self-owned navigation service to guide potential customers to his/her own shop, and a conference organizer can direct attendees to the conference location with little effort. Among many other similar scenarios, P2P navigation, which is self-deployable and almost zero-effort, acts as a promising alternative to traditional centralized localization and navigation systems.

2.2 System Overview

Pair-Navi enables this kind of leader-follower navigation by leveraging mobile vision capabilities. The system architecture is illustrated in Figure 3. For both leaders and followers, they walk naturally in the course and hold their smartphones to shoot videos along the trace. Every video frame captured by his/her smartphone camera is sent to a cloud server via network for further processing. Although we leverage SLAM technology, our system does not involve all modules for leader and follower. For a leader, we feed the video clips into three SLAM modules, i.e., *Initialization*, *Visual Odometry*, and *Trajectory Construction*, to simultaneously form a trajectory (a sequence of 3D camera poses) and construct a map (3D map points and key frames). When the trace is completed, the trajectory and map data, in addition to leader-labeled starting and destination places, will be stored on the server.

In case a follower arrives, he/she first chooses the destination and will be provided with a reference trace leading to the same destination, contributed by some leader. When the follower walks, our system will immediately locate his/her relative location to the reference trace by *relocalization* based on the currently captured video frame. If relocalization succeeds, the follower will be relatively located and accordingly instructed with timely walking hints that come with the leader's reference trace. However, if relocalization fails, which indicates the follower may deviate from the given path in the following steps, a *deviation detection* module will be triggered to launch auxiliary

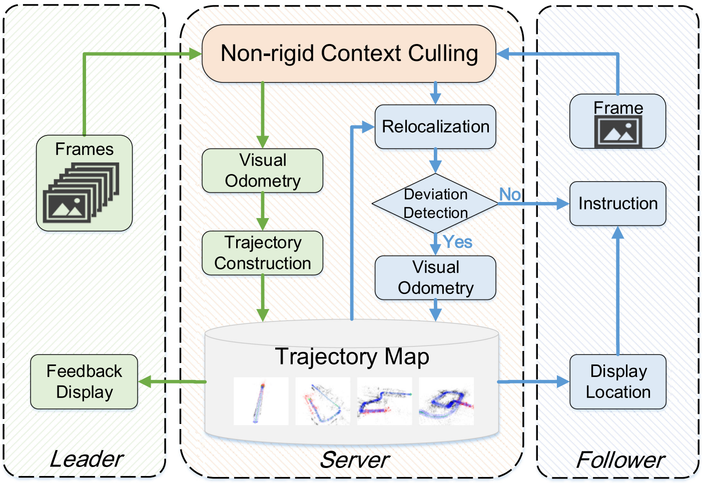


Fig. 3. Pair-Navi architecture.

visual odometry to track the follower's camera poses independently. The results will also be fed back and displayed on the follower's phone, together with the reference trace, so that the follower can get himself back on the correct path for further navigation.

A key and unique component in Pair-Navi is the NRCC, which aims to extract and subtract dynamic contents in the video clips to combat time-varying environments. To ensure precise reference trajectory generation and robust follower localization, NRCC is applied to both leader's and follower's videos.

3 VISUAL NAVIGATION IN NON-RIGID ENVIRONMENT

In this section, we describe how Pair-Navi utilizes visual SLAM for navigation and how it addresses non-rigidity in the environment.

3.1 Visual SLAM for Navigation

In Pair-Navi, we use monocular visual SLAM that works with a single camera since most smartphones have only one camera on the back side. We decouple the tightly coupled modules of a monocular visual SLAM system, and reorganize the required modules into leader and follower applications to simplify for real-time and meanwhile ensure accuracy. In particular, we introduce three key modules, initialization, visual odometry, and relocalization, as follows.

Initialization. When the system initially launches, our system takes an initialization step using epipolar geometry [24, 36] to locate the camera in an initial map with 3D map points as landmarks of the environment. To begin with, the camera captures two very early video frames, from which we extract and match certain feature points that describe the video frames (see Figure 5(a)). In Pair-Navi, we leverage ORB feature point [47], which has comparative matching accuracy with the state-of-the-art SIFT [39] and SURF [5], yet is far more efficient in computation.

Then the relative camera motion between the first two frames, denoted as ${}^{c_1}_{c_2}T$, where T represents a transformation matrix and c_i denotes certain coordinate systems,[1] is computed by solving

[1] This convention is brought up by [10], where the left superscript of a variable means the reference coordinate system and the left subscript means the objective coordinate system. By multiplication, we can cancel the left subscript of a variable together with the left superscipt of the next variable. For example, ${}^{A}t = {}^{A}_{B}T \cdot {}^{B}t$ means a vector t described in B coordinate system multiplied by transformation ${}^{A}_{B}T$ equals the description of t in A coordinate system.

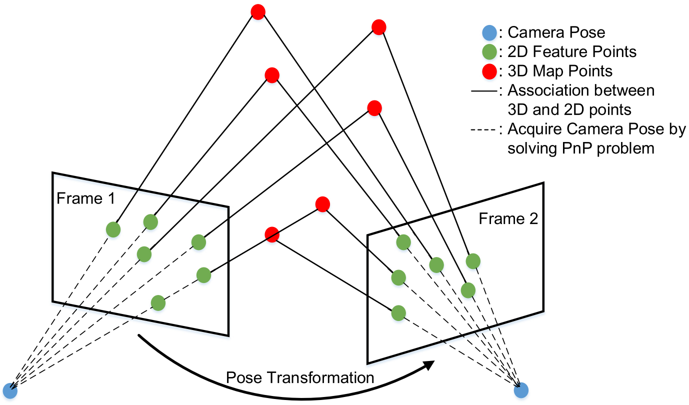
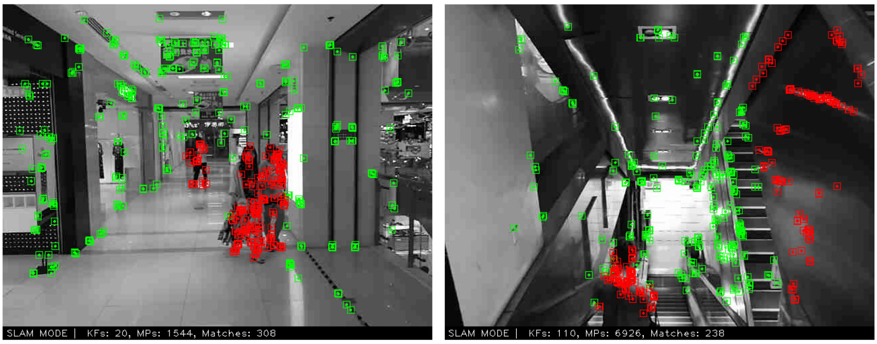

●: Camera Pose
●: 2D Feature Points
●: 3D Map Points
——: Association between 3D and 2D points
----: Acquire Camera Pose by solving PnP problem
Frame 1
Frame 2
Pose Transformation

Fig. 4. Illustration of visual odometry.


SLAM MODE | KFs: 20, MPs: 1544, Matches: 308
SLAM MODE | KFs: 110, MPs: 6926, Matches: 238

(a) (b)

Fig. 5. Examples of non-rigid contexts: feature points from non-rigid contexts, pedestrians in (a) and mirror reflections in (b), are recognized and marked as red, while the remaining feature points from rigid and static objects are green.

the so-called epipolar constraint equation [24]. When the camera motion between the early two frames is acquired, the camera pose of the first frame is set to be the "world coordinate system" w, and the camera pose of the second frame becomes ${}^{w}_{c_2}T$. Based on camera motion between them, the depths of the feature points are also calculated by triangulation [24]. After triangulation, we obtain the 3D coordinates (2D pixel coordinate plus depth) of every feature point in the camera coordinate system. Since the first frame is set to be the world coordinate, we actually get the 3D coordinates of every feature point in the world coordinate. Therefore, an initial sparse 3D map of the environment, i.e., a sparse set of 3D map points, is accordingly built up.

Visual Odometry. After initialization, the **visual odometry** (**VO**) module takes charge of the system, to continuously track the camera pose from consecutive video frames. Specifically, when a new video frame arrives, its 2D ORB feature points are extracted and associated to already-created 3D map points by feature matching.

Figure 4 shows an example of the extracted feature points associated to 3D map points. From the association of 2D feature points and 3D map points, we can acquire the camera pose of this frame by solving a so-called **Perspective-n-Point** (**PnP**) problem [29], which determines the position and orientation of a camera from a set of correspondences between 3D points and 2D pixel points. As shown in Figure 4, given the camera poses of two frames, new 3D map points can be generated by calculating the 3D coordinates via triangulation among the two frames [24]. Repeatedly, as more new frames come, a trajectory of the camera poses and a map of the 3D landmarks and corresponding keyframes[2] are built incrementally. In Pair-Navi, VO is the core module for a leader to construct a trajectory map.

Relocalization. In order to reuse a previously built trajectory map, relocalization module comes in handy, which is the central component in the follower program. It compares a video frame with the keyframes in the map, and finds out the most similar keyframe based on feature point matching. This step is also called visual place recognition, of which the state-of-the-art is Bags of Visual Words [18]. After the most similar keyframe is found, the feature points in the current follower frame are associated to the feature points in the selected keyframe. On this basis, a PnP problem [29] is solved in the same manner as VO module to get the camera pose, thus relocalizing the camera in the map. In Pair-Navi, the follower program mainly employs relocalization to achieve efficient relative localization (Section 4).

Note that classical monocular visual SLAM still involves complicated steps like loop closing detection, global optimization, and so forth [36]. In our system, however, we merely apply the necessary modules for leaders and followers, respectively, to avoid intensive computation, as detailed in Section 4.

3.2 NRCC

3.2.1 Limitations of Visual SLAM in Non-Rigid Environments. While visual SLAM technology underpins a promising solution to infrastructure-free navigation, it is vulnerable to non-rigid indoor environments with significant dynamic changes over time. Specifically, the limitations are twofold.

Low-Precision Trajectory. As is described in the above section, to calculate the camera poses, we need to match feature points first. Therefore, correct feature point matches influence the accuracy of the constructed trajectory. In the presence of erroneous matches, the generated trajectory will deviate from the ground truth. Figure 2 illustrates an example of matching two consecutive video frames from a typical shopping mall. As seen, feature points extracted from those dynamic contexts (e.g., pedestrians) will lead to considerable erroneous matching, as indicated by red lines in Figure 2. As a consequence, if we calculate camera poses from the whole set of feature point matches without screening, the constructed trajectory will deviate from the truth, depraving further navigation. To obtain precise trajectory, we need to intelligently recognize the non-rigid contexts and sift out their corresponding feature points.

Vulnerable Relocalization. Apart from degrading trajectory precision, non-rigid contexts further harm relocalization robustness. In practice, the environment observed by a leader and later a follower may change significantly, leading to feature point mismatches and thus large relocalization errors or even relocalization failures. In one situation, if there are only a small number of matching outliers, the feature points may be matched to wrong 3D map points, resulting in errors in camera pose computation. In another situation, if a large portion of feature points fail to match, a wrong keyframe will be chosen and relocalization fails.

Therefore, to ensure accurate trajectory construction for leader as well as successful relocalization for follower navigation, we propose the NRCC module that takes out features points from non-rigid contexts and only exploits the remaining reliable feature points, mainly from rigid and

[2] Keyframes are a subset of all frames to eliminate redundancy.

Table 1. Object Classification for Indoor Scenario

Non-Rigid Context objects	Rigid Context objects
person, hair drier, toothbrush, cat, keyboard, phone, bottle apple, cup, backpack, umbrella, handbag, tie, suitcase, dog frisbee, book, clock, skis, snowboard, sports ball, kite bat, glove, skateboard, surfboard, mouse, remote, glass fork, knife, spoon, bowl, banana, tennis racket, scissors teddy bear, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, laptop	couch, potted plant*, dining table tv, microwave, oven, toaster refrigerator, vase*, bed, toilet, sink

static areas, for frame matching. Our key observation is that there are abundant feature points for frame matching, allowing room to cull part of them without degrading visual SLAM performance.

3.2.2 NRCC via Mask R-CNN. In Pair-Navi, we adapt Mask R-CNN [25] for NRCC. Mask R-CNN is a recent framework for instance segmentation. It aims to separate different instances in an image via a segmentation mask for each instance. We use the Mask R-CNN network pre-trained on the COCO dataset [32], and select the object categories that are suitable for indoor scenario.

Since we aim at distinguishing rigid and non-rigid context, we divide all the object categories into two sets: rigid context objects and non-rigid context objects as shown in Table 1. If an object belongs to the rigid context set, it means the location, pose, and shape of the object will not change, and whenever the leader or follower come to the same place, they will observe the object in the same situation. On the contrary, an object is dynamic if it belongs to the non-rigid context set. What is worth mentioning is that the classification of the two sets is flexible. Some objects (e.g., vase, potted plant) can belong to either the rigid context object set or the other, depending on the time interval between the leader's trajectory being constructed and the follower's navigation. For example, for vases, if the time interval is shorter than 7 days, they will be regarded as rigid context objects. Yet if the interval is longer than 7 days, they will be treated as non-rigid context.

When the server receives a video frame from the camera, we extract its feature points and use the Mask R-CNN framework to detect the non-rigid contexts, then we filter out the feature points that lie in the masks of dynamic instances. As shown in Figure 5, the feature points filtered out, which are marked by red color, are mainly from people, and the map is generated only using the feature points belonging to static environment, which are marked by green. Moreover, we also use the method proposed in YOLO [44] to detect mirrors and smooth surfaces in video frames (as shown in Figure 5(b)). We believe the illumination change in places like academic buildings may be drastic, rendering the feature points lying in mirrors volatile. Therefore, they are also culled to increase system robustness. After this preprocessing of the video frame, although the user may be facing a non-rigid environment, the feature points lying in the masks of dynamic instances will not be involved in trajectory construction (for leaders) or relocalization (for followers). As shown in Figure 6, removing non-rigid contexts helps improve precision and avoid potential relocalization failures.

4 REAL-TIME NAVIGATION

In this section, we present the design of Pair-Navi to enable real-time navigation.

4.1 Leader Trajectory Map Construction

To construct a map of navigation trajectories, a leader simply holds the smartphone in front of his/her body to shoot video frames while walking along his path. Upon finishing, the leader

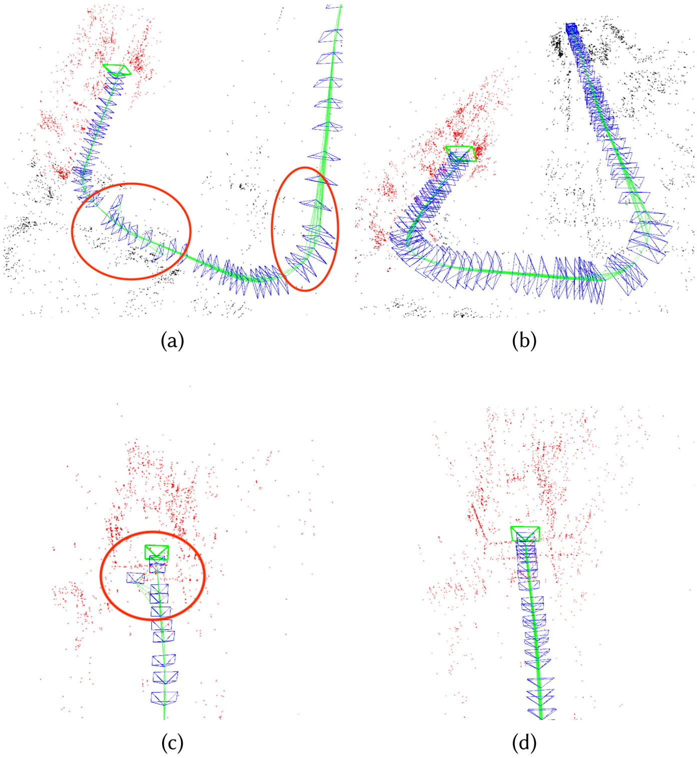


Fig. 6. The effect of NRCC on trajectory construction: A complex trajectory computed (a) without NRCC, in which the camera poses of some frames in the red circle are deviated from the original path, and (b) with NRCC, in which the camera poses are correct and smooth. Similarly, (c) and (d) show a simple trajectory without and with NRCC, respectively.

uploads to the server the video labeled with the origin and the destination. The server then runs visual SLAM, including initialization and visual odometry, with Mask R-CNN for each frame to cull out the non-rigid contexts, and then the map is saved for followers.

4.2 Follower Real-Time Navigation

To begin with, a follower chooses the destination given by the server, then the smartphone runs as a ROS node, capturing video frames and sending them to the server. The server, as another ROS node, loads the chosen trajectory and map, and starts running: it relocalizes every follower video frame, visualizes the camera pose in the leader's trajectory map, and calculates the navigation instruction. On receiving all the returned messages, the follower can see the navigation instruction, together with an visualization of the current camera pose and leader's trajectory map.

If the relocalization observes inadequate quantity of 3D map points, the auxiliary VO is launched from this frame, and will take the place of the relocalization to show camera pose if it truly fails. At this time, the follower can still see the camera pose in the leader's trajectory and spare little effort to return to the track. Once the relocalization is successful again, the auxiliary VO is shut down, and the follower goes back to the normal case, in which only relocalization is executed.

Our system has the following three designs that ensure it can run in real-time (by "real-time," we mean at least 10 fps, the typical value of persistence of vision):

Relocalization. We decouple a typical visual SLAM system, adaptively combining the relocalization module and VO module into our follower navigation program. In this design, we avoid the heavy overhead to acquire the follower's current position with a full visual SLAM system since the beginning of the follower's navigation. Instead, we get the utmost of the leaders' efforts, only running relocalization for follower, which is more efficient than a full visual SLAM system yet produces as accurate results as it. As will be demonstrated in Section 5, our design considerably reduces running latency compared to a conventional SLAM system.

Mask Synchronization Strategy. Non-rigid context culling is for both leader and follower's video frames, but in different ways. For leaders, we generate one mask for each frame, because the leader's map construction can be done offline. Yet for followers, the calculation of Mask R-CNN is relatively slow, at an average frame rate of about 5 fps. So we take a tradeoff strategy to synchronize Mask R-CNN with visual SLAM. We aggregate every two frames (corresponding to 0.2 s due to 10 fps), and generate one mask for them both.

The rationale behind this is that a user will move for only about 20 cm during the 0.2 s time-window, assuming a typical walking speed of 1 m/s. Therefore the potential scene changes in two consecutive frames will not be too significant, which may slightly affect trajectory construction accuracy for leaders, but do not necessarily influence relocalization for followers. In summary, to obtain a highly precise and reliable trajectory for followers to use, we need to cull non-rigid contexts for each frame; yet to save the the computational resources for real-time follower navigation, we can confidently reuse the masks for several consecutive frames.

The Fringe Benefit of NRCC. Additionally, NRCC brings a fringe benefit to the real-time performance. Since the number of feature points is reduced, further computation, including feature point matching, calculating bags of visual words, PnP, and so forth, is simplified by the proportion of rigid feature points in all the feature points. The quantitative benefit of this simplification is shown in Figure 15.

4.3 Follower Deviation Handling

Existing P2P navigation systems such as *Travi-Navi* and *FollowMe* have no function of deviation handling, and thus if follower deviation happens, the tracking will lose and the navigation will fail, with no recovery approach. In contrast, we add an auxiliary VO module to track the deviated trajectory, which acts as a fail-safe measure to guarantee the success of navigation.

In our scenario, deviation happens when the follower walks off the instructed course, or even when camera pose slightly deviates. In these cases, the follower's relocalization module observes inadequate 3D map points in a frame, which is when the system launches the auxiliary VO. The auxiliary VO, extracting new feature points and generating new 3D map points, runs in parallel with the relocalization to retain continuous tracking and show the camera pose in the trajectory map, as shown in Figure 7. With the camera pose shown in the trajectory map, the follower can easily go back on the right track. When it comes to an extreme situation in that the relocalization truly fails, the VO will come on the stage and guide the follower to go back to the course by showing his pose in the leader's trajectory map. Once the relocalization observes a healthy quantity of 3D map points, the auxiliary VO will shut down and hand over the navigation to the relocalization module.

Note that to further keep our system running in real-time with this auxiliary VO, we strictly select a feature point quantity threshold for triggering it, so that redundant calculation is eliminated. In this way, we make the system tolerate navigation deviations in harsh environments while still maintaining real-time performance.

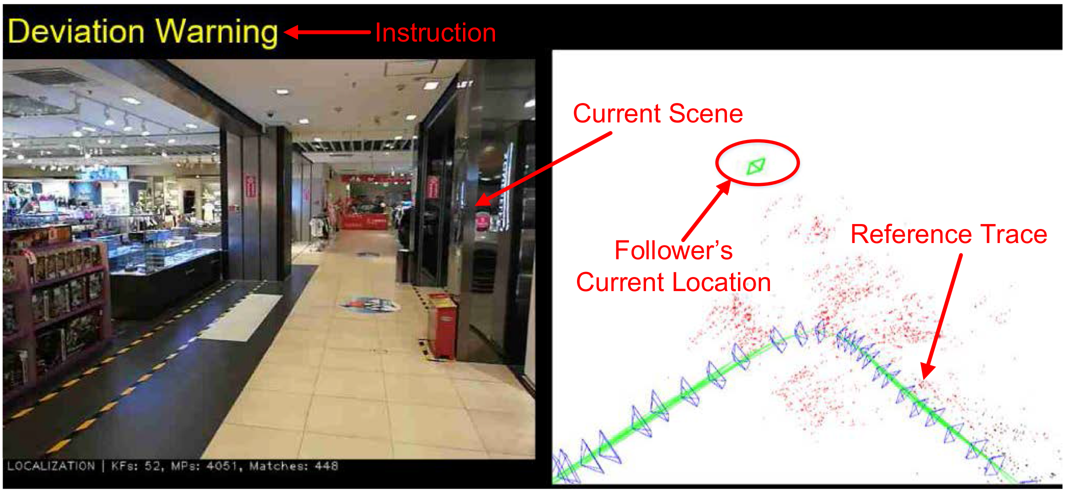

Deviation Warning
Instruction
Current Scene
Follower's Current Location
Reference Trace
LOCALIZATION | KFs: 52, MPs: 4051, Matches: 448

Fig. 7. Illustration of auxiliary VO when deviation happens.

4.4 Navigation Instruction Calculation

The calculation of navigation instruction takes the relocalized camera pose of the current follower frame with reference to the world coordinate system, ${}^{w}_{c}T$, as an input (the left superscript w denotes "world," which is set to be the first frame of the leader's map, and the left subscript c denotes "current"). In this way, every follower frame is localized and assigned a pose in the leader's trajectory map. Next, the server needs to compute navigation instruction through the follower's current camera pose. Since in the leader's trajectory, there is one pose ${}^{w}_{r}T$ (r denotes "relocalization") that is closest to follower's current camera pose, we choose it as a representative, and average the translation[3] of the next 10 poses in the leader's trajectory as ${}^{w}t_{next}$. The reason why we select 10 following poses is that the average time interval of 10 keyframes is approximately 1.5 seconds by our statistic and the average step period of human is around 0.4–0.6 seconds, therefore, this 10-keyframe interval suggests approximately in which direction the follower should head to in the next two or three steps.

So far, we have the pose of the current follower frame ${}^{w}_{c}T$, and the average translation of the next 10 poses ${}^{w}t_{next}$. By converting t_{next} to homogeneous coordinate (adding a fourth dimension with value 1, such that t_{next} becomes $[x\ y\ z\ 1]^T$), we can compute

$$ {}^{c}t_{next} ={}^{c}_{w} T \cdot {}^{w} t_{next}, \tag{2} $$

where ${}^{c}t_{next}$ is the homogeneous coordinate of t_{next} in the current follower frame. The coordinate axis direction is already shown in Figure 2.

Finally, we can generate the navigation instruction with the angle between ${}^{c}t_{next}$ and the imaging plane of the camera

$$ \alpha = \arctan \frac{z}{x}, \tag{3} $$

[3] Here the translation is a 3D vector denoting the position of a pose. A pose can be denoted by a transformation matrix

$$ T = \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1} $$

where R is 3×3 rotation matrix, t is the translation vector.

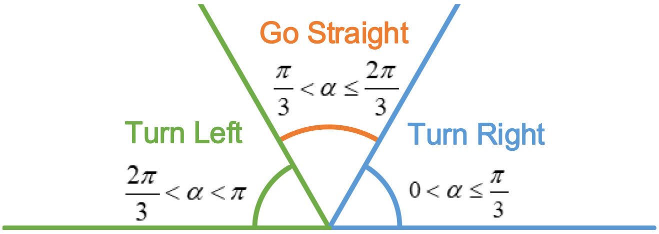

Go Straight
$\frac{\pi}{3} < \alpha \leq \frac{2\pi}{3}$
Turn Left
$\frac{2\pi}{3} < \alpha < \pi$
Turn Right
$0 < \alpha \leq \frac{\pi}{3}$

Fig. 8. Illustration of the relationship between navigation instruction and α.

where we assume the follower is holding the phone in a steady level so that the y coordinate of ${}^{c}t_{next}$ has little effect on navigation.

As shown in Figure 8, by our design, if $\frac{\pi}{3} < \alpha < \frac{2\pi}{3}$, the system will give out “Go Straight” instruction; when $0 < \alpha < \frac{\pi}{3}$, “Turn Right”; and when $\frac{2\pi}{3} < \alpha < \pi$, “Turn Left.” Note that we assume the leader will not go backwards when constructing the trajectory map, and thus theoretically z is always positive, so no “Go Backward” instruction is necessary. If z is truly negative, which is due to flawed relocalization, we will give “Deviation Warning” instruction. If a deviation is later detected, the auxiliary VO will be triggered to continue the navigation.

4.5 Implementation

We implement our system on the ROS Kinetic [2] platform. The user program is developed on the ROS-Android [1] platform. The server’s SLAM program is developed on *ORB-SLAM* [36] on ROS, and we develop the system visualization upon the visualization of *ORB-SLAM*, with some modifications via OpenCV. We resize all the frames to 640 × 480, which is relatively high-resolution and still not too large to prolong the process time of each frame. The phone-server communication resorts to ROS topic publication and subscription, which guarantees all the imformation is integral because ROS data transfer is based on a TCP protocol.

We applied our Mask R-CNN models with the ResNet-FPN-50 backbone and the network parameters are pre-trained on the COCO image daaset. The Mask R-CNN code is implemented in Python-3.6.5 with PyTorch-0.4.0.

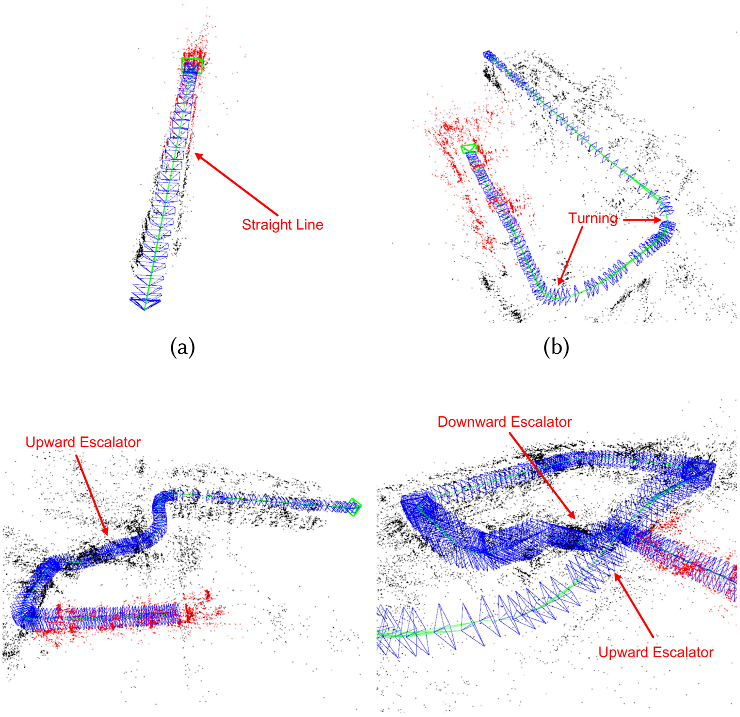
5 EXPERIMENTS AND EVALUATION

5.1 Experiment Settings and Methodology

Experiment Venues. We conducted extensive experiments in an office building, a gymnasium, and the firts through fourth floors of a shopping mall, with area sizes of about 400 m^2, 1,000 m^2, and 6,000 m^2, respectively. The three testing environments have diverse conditions. In particular, the crowded shopping mall is the most dynamic. The office building has the most drastic illumination oscillation during a day. The gymnasium has the medium crowdedness among the three environments.

Data Collection. Overall, we design 21 navigation paths, including 6 short paths (≤100 m), 7 medium paths (100–200 m), and 8 long paths (≥200 m), covering all the main pathways of the testing areas. Figure 9 shows four trajectories of different lengths constructed from the three areas. The reference trajectories for these paths are constructed by three different leaders. The total length of leaders’ reference trajectories is about 3.1 km with 33,452 video frames, among which 6,781 keyframes are selected, and the followers’ total walking distance is around 15 km.

Devices. We tested Pair-Navi on a variety of Android mobile devices, including Huawei P10, Nexus 6p, Nexus 7, and Lenovo Phab2 pro. Since the main device discrepancies are camera



(a) (b)

(c) (d)

Fig. 9. Four typical real trajectories in our experiment: (a) a straight line; (b) a U-turn; (c) a complex trajectory going up an escalator; (d) a complex trajectory going up one escalator and then down another.

intrinsics (i.e., focal length, lens center, and distortion), we calibrate camera intrinsics of the smartphones and accordingly rectify video frames. The server, Lenovo IdeaPad-Y700 with i7-6700HQ CPU of 2.6 GHz main frequency and 8 G RAM, runs the Ubuntu 16.0.4 operating system and ROS Kinetic. For Mask R-CNN, the GPU we used is TITAN V with CUDA version 9.1.85 and cuDNN-7.05.

Users. We recruited four volunteer followers to walk along different routes naturally as they usually do. The follower behaviors are diverse in camera holding gestures and heights. For example, one user prefers to hold the camera with two hands, while others tend to use their right hands. Thus, the cameras suffer various extents of shake when the users are walking, which may cause different feature point matches. User study is conducted on three particular days: the same day as the trajectories were constructed, 1 week later, and 2 weeks later.

Comparison. To evaluate the performance of Pair-Navi, we implemented *Travi-Navi* [70] and *FollowMe* [51], two start-of-the-art P2P navigation systems for comparison.

Evaluation Metrics. Similar to some existing works like *Travi-Navi* and *FollowMe*, we set checkpoints at turns, escalators, and some landmarks on each trajectory. In total, we set 274 checkpoints for the 21 navigation paths. The followers were not informed of navigation routes, the final destination, or the checkpoint locations. *Navigation success rate* is defined as the rate of successful arrival at each checkpoint in *Travi-Navi* and *FollowMe*. Thanks to the employment of deviation handling, this rate is always 100% in Pair-Navi, which means the followers arrived at the destinations successfully in all cases. So instead, we use a more strict definition of navigation success rate as $1 - p$ (where p is the rate of auxiliary VO launches) for Pair-Navi, while keeping the original defination of navigation success rate unchanged for *Travi-Navi* and *FollowMe*.

5.2 Overall Performance

5.2.1 Performance Comparison. The performances of Pair-Navi as well as the two state-of-the-art approaches to compare are depicted in Figure 10. We find that Pair-Navi achieves the best performance among all three of them, no matter how long the time interval is. The average navigation success rates by Pair-Navi in the same day of the trajectory's construction, after 1 week and after 2 weeks are 98.6%, 93.2%, and 83.4%, respectively. Compared with the immediate performance (tested in the same day), the navigation success rates after 2 weeks decline to 14.1%, 49.3%, and 59.3% in Pair-Navi, *Travi-Navi*, and *FollowMe*, respectively. In contrast to *Travi-Navi* and *FollowMe*, Pair-Navi attains high navigation accuracy after a 2-week interval, and outperforms *Travi-Navi* by 50.9% and *FollowMe* by 80.4%.

The reason for this performance gain is twofold:

(1) Innate metric advantages of vision-based methods. On the one hand, radio-frequency-based and magnetic-field-based systems suffer from metric error of localization typically averaging 3–5 m [43, 50] due to intrinsic defects such as fluctuation of signal strength, change of indoor multipath environment, and the inaccuracy of pedestrian dead-reckoning. On the other hand, vision-based methods tackle metric error rationally, which consists of three parts: rolling shutter cameras in most smartphones skewing the image, which is negligible; the calculation of epipolar geometry and PnP, which have closed form solution and is accurate; and feature point mismatch, which is largely tackled by ORB feature. As is reported in [36], typical localization error in indoor senarios (TUM RGB-D dataset) in visual SLAM systems is within 5 cm, much smaller than that in radio-frequency-based and magnetic-field-based systems.

With an error of 3–5 m, a person can miss turning points in indoor environments every now and then, but it is not the case with an error of 5 cm.

(2) Robust design of NRCC. Apart from the innate metric advantages of vision-based systems, benefiting from the design of NRCC, the system robustness is remarkably enhanced. Radio-frequency-based and magnetic-field-based methods are plagued by ambiguity of localization when several signal patterns of different locations are close to each other, under which cases the tracking of phones can easily be lost. With the design of NRCC, our system obtains good relocalization robustness.

Furthermore, as for long-term performance, NRCC attacks environment change by ruling out the changed while maintaining the unchanged contexts in the trajectory map. In this way, even after 2 weeks, our system can handle time-varying environment and palliate the performance drop even in the long term. On the contrary, the deterioration of trajectory information in radio-frequency-based and magnetic-field-based systems is not dealt with properly. In Section 5.3, we will further delve into the effectiveness of NRCC.

5.2.2 Performance Under Different Conditions. We invite four volunteers to examine the robustness and practicability of Pair-Navi in different areas and at different times. As shown in Figure 11, Pair-Navi achieves an average navigation success rate of more than 85% for each user and the decrease of success rate for each one is less than 15% after 2 weeks. Furthermore, Figure 12 shows that Pair-Navi yields similar performance regardless of the different crowdedness levels and illumination conditions at different areas. Pair-Navi achieves a consistently delightful success rate of more than 90%, 85%, and 80% in the office building, gymnasium, and large shopping mall under a time interval within 2 weeks.

To further demonstrate the applicability of Pair-Navi, we note that the trajectories in the office building are mainly constructed by User-2 and User-3, and in the gymnasium and shopping mall by User-1 and User-2. The heights of the four users are different, and camera holding gestures are variant. Figure 13 reflects the robustness of Pair-Navi for different leaders and followers.

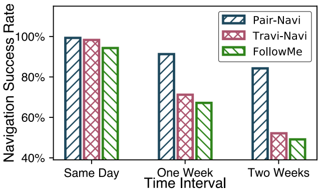


Fig. 10. Different methods.

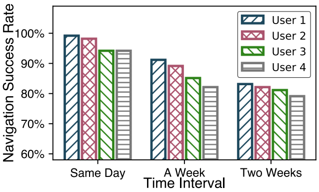


Fig. 11. Different users and time interval.

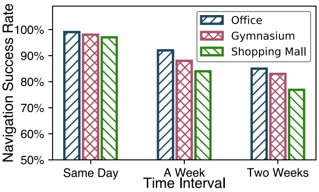


Fig. 12. Different areas and time interval.

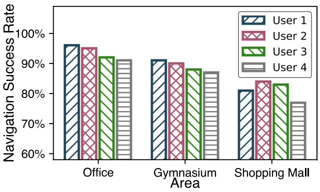


Fig. 13. Different users and areas.

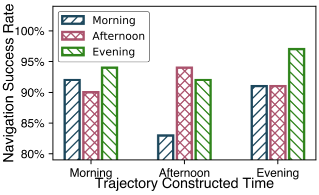


Fig. 14. Different time during a day.

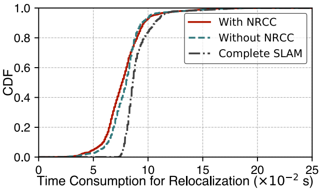


Fig. 15. System latency.

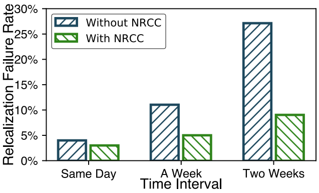


Fig. 16. Impact of NRCC over different time interval.

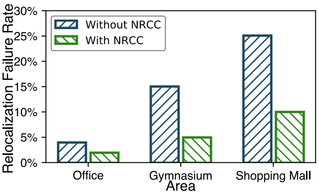


Fig. 17. Impact of NRCC in different areas.

Navigation success rates for all users in different areas are more than 80% and the success rate gaps between different users are within 5%.

5.2.3 Impact of Illumination. We further tested our system with different illumination conditions in the office building, which undergos the most drastic illumination oscillation among the three areas. We first asked a leader to walk four pathways in the morning, afternoon, and evening during a day. Then we asked volunteers to walk the same pathways correspondingly and calculate the navigation success rate of each test case. As shown in Figure 14, whenever followers walk the pathways, the navigation success rates are more than 80% and more than 90% if followers walk at evening. Generally, in the office building, we usually turn on all of the lights at evening, the majority at morning, but rarely at afternoon, which lead to the same video frame captured at evening enjoys the most drastic light and shadow oscillation. Therefore, the video frame has more ORB feature points than captured at morning and afternoon [47] and the relocalization success rate of the frame will increase.

5.2.4 System Latency. We recorded the time consumption of all frames in all followers' navigation experiments. As shown in Figure 15, Pair-Navi reduces the average relocalization time for one frame to 76 ms. Compared with a complete SLAM system, the average delay is reduced by 17.4%, moreover, the percentage of frames using less than 100 ms for relocalization increase from 81.6% to 91%. Surprisingly, we also observe that NRCC even slightly reduces the system latency by 4 ms per frame on average. This is because NRCC shrinks the number of valid features points involved in relocalization. The average mask process time for each frame (resized as 640 × 480) is

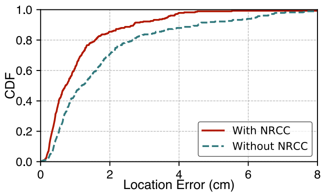


Fig. 18. Localization accuracy on TUM dataset.

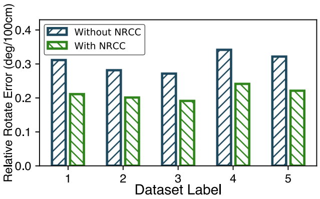


Fig. 19. Tracking accuracy on KITTI dataset.

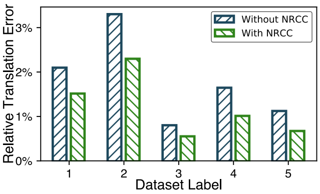


Fig. 20. Mapping accuracy on KITTI dataset.

0.18 s in our system. In other words, a mask takes in charge of the NRCC of two frames that are captured in 0.2 s. In a nutshell, Pair-Navi accomplishes relocalization and navigation within the system sampling time of 0.1 s and runs fluently in real-time, with partial computation off-loaded to a cloud server. As our future work, we plan to optimize to a complete standalone system on smartphones based on model compressing.

5.2.5 Energy Consumption. We record the energy consumption of the follower's navigation app on the smartphones. On the Huawei P10, the program ran 41 minutes and 48 seconds and consumed 203.12 mAh, while the battery level dropped from 100% to 69%. On the Lenovo Phab2pro, the program ran 44 minutes and 36 seconds and consumed 348 mAh, while the battery level dropped from 100% to 85%. Since indoor pathways from one place to another are usually less than 15-min walking distance, we consider the energy consumption of Pair-Navi is acceptable.

5.3 Impact of NRCC

To demonstrate the effect of non-rigid context culling, we first compare the relocalization failure rate with and without non-rigid context culling. Specifically, we save all the video frames captured by a follower's camera and record whether each frame can be relocalized (matched to a keyframe in the trajectory map with enough feature point matches). This part of the experiment is conducted on all 21 navigation trajectories mentioned above and under different time intervals. Furthermore, we have performed an extra extensive experimental validation of NRCC on two standard benchmarks for visual SLAM: TUM [54] and KITTI [20] datasets. We mainly focus on two targets: *Pose Tracking Accuracy* and *Mapping Accuracy*, and we compare the precision of ORB-SLAM with and without NRCC.

5.3.1 Performance in Navigation Scenario. As shown in Figure 16, the relocalization failure rate increases from 4% to 27% without NRCC in the same day, 1 week later, and 2 weeks later, while it keeps low at 3%, 5%, and 9%, respectively, with NRCC. In other words, the use of NRCC significantly decreases the relocalization failures by more than 65% when the time interval exceeds 2 weeks.

Furthermore, Figure 17 shows the relocalization robustness at different areas. We conducted the experiment under the time interval of 2 weeks. The relocalization failure rate at the office building, the gymnasium, and the shopping mall is 2%, 5%, and 10% with NRCC, compared to 4%, 15%, and 25% when without NRCC. On average, NRCC declines relocalization failures by 57%. Especially in the gymnasium, the failure rate is decreased by 67%, which reflects remarkable improvement on robustness.

5.3.2 Performance in Standard Datasets. The TUM benchmark is an excellent dataset to evaluate the accuracy of camera localization as it provides several sequences with accurate ground truth obtained with an external motion capture system. The odometry benchmark from the KITTI dataset contains 11 sequences from a car driven around a residential area with accurate ground truth from GPS and a Velodyne laser scanner. This is an exceedingly challenging dataset for monocular vision due to fast rotations and areas with lots of dynamic objects, which make data association more difficult.

We evaluate the general localization accuracy in 16 hand-held sequences of the TUM RGB-Moncocular benchmark. Moreover, in five sequences from the KITTI dataset, we evaluate the tracking accuracy of camera pose and precision of the constructed map.

Localization Accuracy in TUM Dataset. Figure 18 depicts the impact of the proposed NRCC in indoor localization scenarios. As shown, the average localization of ORB-SLAM with and without NRCC is 1.24 cm and 1.97 cm, respectively. The use of NRCC significantly increases the localization precision by more than 35%.

Pose Tracking Accuracy in KITTI Dataset. We further examine the impact of NRCC on camera pose tracking task in the KITTI dataset. We calculate the **Relative Rotation Error** (**RRE**, an essential evaluation indicator in the KITTI dataset) of the tracking result with and without NRCC. As shown in Figure 19, the tracking accumulative bias is within 0.25° for 1-m-length traces with NRCC, compared to 0.34°/m when without NRCC. On average, NRCC declines RRE by 25.6%.

Mapping Precision in KITTI Dataset. Finally, we evaluate the impact of NRCC on mapping accuracy in the KITTI dataset. **Relative Translation Error** (**RTE**, another fundamental evaluation indicator in the KITTI dataset) is used to demonstrate the mapping accuracy. As shown in Figure 20, in all five frame sequences, NRCC declines RTE by 10%.

In summary, the leverage of NRCC achieves enhanced localization accuracy and mapping precision compared with original ORB-SLAM in all evaluations. The delightful performance gain is owed to the robust design of NRCC, which attacks environment change by ruling out the changed while maintaining the unchanged contexts in the trajectory map.

6 RELATED WORKS

Indoor P2P Navigation. Traditional indoor navigation solutions require a global map of the indoor floor plan in infrastructure-ready indoor environments (e.g., Wi-Fi [42, 55, 59, 65, 66, 68], Bluetooth, RFID [49]), and navigation instructions are provided based upon the absolute position in the global map. Recently, P2P navigation appears as another solution to indoor navigation, which does not rely on a complete global map and absolute localization in the map [9, 46, 51, 52, 69, 70]. In [9], an electronic escort system was proposed by using crowd encounter information and dead-reckoning techniques. The most relevant works *Travi-Navi* [70], *FollowMe* [51], and *ppNav* [69] all employ trace-driven navigation on smartphones. *Travi-Navi* synthesized Wi-Fi and inertial measurement to bootstrap navigation services without indoor floorplan. *FollowMe* exploited magnetic sensing and dead-reckoning to achieve lastmile navigation for smartphone users. *ppNav* utilized the ubiquitous Wi-Fi fingerprints in a novel diagrammed form and extracted both radio and visual features of the diagram to track relative locations. *edgeSLAM* [63] leverages the power of edge server to enable mobile devices to run visual SLAM in real-time. In contrast, Pair-Navi exploits the power of vision, which is infrastructure-free and demonstrated to be more efficient, precise, and further beneficial to various vision-based applications, such as indoor 3D-reconstruction, store sign identification, and so forth.

Monocular Visual SLAM. The pioneer work of visual SLAM [13] adopted a filtering-based approach. Later, optimization-based methods [27, 35] came on stage and were demonstrated to be more accurate [53] than a filtering-based approach. *ORB-SLAM* [36], the state-of-the-art monocular visual SLAM work, used *DBoW2* [18] as the place recognition module, and *g2o* [28] as the optimization framework. The above-mentioned monocular visual SLAM systems lie in the genre of feature point method, whose counterpart, though not comparatively well-studied than the other, is the genre of direct method [17, 19], which focuses mainly on pixel gradient rather than feature point. Latest researches on monocular visual SLAM have at least two trends. Some works attempted to incorporate semantics into monocular visual SLAM [6, 8] for better robustness in time-varying environments. Others are dedicated to solving the scale uncertainty of monocular SLAM, introducing IMU into monocular visual SLAM [16, 37].

Being able to compute the camera pose while generating the map and environment, visual SLAM is a suitable technique for indoor navigation.

Instance Segmentation. The **Region-based CNN** (**R-CNN**) approach [22] leveraged candidate object regions [26] and evaluated convolutional neural networks for each **Region of Interest** (**RoI**) for object detection. Driven by the effectiveness of R-CNN, many approaches to instance segmentation are based on segment proposals. Earlier methods [23] resorted to bottom-up segments [3, 57]. DeepMask [40] and following works [41] learn to propose segment candidates, which are then classified by Fast R-CNN. In these methods, segmentation precedes recognition, which is slow and less accurate.

Most recently, Li et al. [31] combined the segment proposal system in [11] and object detection system in [12] to simultaneously address object classes, boxes, and masks, making the system fast. Furthermore, R-CNN and [31] were extended to allow RoI extraction on feature maps using RoIPool [21] and then advanced to Faster R-CNN [45] by learning the attention mechanism with a **Region Proposal Network** (**RPN**). On this basis, Mask R-CNN [25] was proposed to use mask predictions for classification and is the state-of-the-art in instance segmentation. More specifically, Mask R-CNN followed the idea of Fast R-CNN [21] that applies bounding-box classification and regression in parallel; it proposed an RoIAlign layer that removes the harsh quantization of RoIPool to properly align the extracted features with the input. In addition, it adopted a two-stage procedure like Faster R-CNN, with a first stage of RPN and a second stage of outputting a binary mask for each RoI.

7 CONCLUSION

In this article, we present Pair-Navi, a robust and real-time P2P navigation system based on visual SLAM, requiring no pre-installed infrastructure or pre-deployed localization services. We decouple conventional visual SLAM into independent modules and reassemble the necessary components into a P2P navigation system. To bolster system performance in time-varying environments, we incorporate Mask R-CNN to dynamically cull non-rigid environmental changes. We implement Pair-Navi on commodity smartphones and conduct experiments in multiple buildings over 2 weeks. Experiment results show that our system outperforms existing solutions in navigation success rate and robustness. We believe Pair-Navi takes a promising step toward practical P2P navigation. Our future works target at fusing crowdsourced trajectories to generalize navigation routes to a larger scale and make a global consistent map.

REFERENCES

[1] ROS Android. 2017. Retrieved from http://wiki.ros.org/android.

[2] ROS Kinetic Kame. 2016. Retrieved from http://wiki.ros.org/kinetic.

[3] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T. Barron, Ferran Marques, and Jitendra Malik. 2014. Multiscale combinatorial grouping. In *Proceedings of IEEE CVPR.*

[4] M. Azizyan, I. Constandache, and R. Roy Choudhury. 2009. Surroundsense: Mobile phone localization via ambience fingerprinting. In *Proceedings of ACM MobiCom.*

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *European Conference on Computer Vision.* Springer, 404–417.

[6] Sean L. Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J. Pappas. 2017. Probabilistic data association for semantic slam. In *Proceedings of IEEE ICRA.*

[7] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics* 32, 6 (2016), 1309–1332.

[8] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D. Tardós, and J. M. M. Montiel. 2011. Towards semantic SLAM using a monocular camera. In *Proceedings of IEEE IROS*.

[9] Ionut Constandache, Xuan Bao, Martin Azizyan, and Romit Roy Choudhury. 2010. Did you see Bob?: Human localization using mobile phones. In *Proceedings of ACM Mobicom*.

[10] John J. Craig. 2005. *Introduction to Robotics: Mechanics and Control*. Vol. 3. Pearson/Prentice Hall, Upper Saddle River, NJ.

[11] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. 2016. Instance-sensitive fully convolutional networks. In *Proceedings of ECCV*. Springer.

[12] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*.

[13] Andrew J. Davison. 2003. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of IEEE ICCV*.

[14] Erqun Dong, Jingao Xu, Chenshu Wu, Yunhao Liu, and Zheng Yang. 2019. Pair-Navi: Peer-to-peer indoor navigation with mobile visual SLAM. In *Proceedings of the IEEE INFOCOM*.

[15] Liang Dong, Jingao Xu, Guoxuan Chi, Danyang Li, Xinglin Zhang, Jianbo Li, Qiang Ma, and Zheng Yang. 2020. Enabling surveillance cameras to navigate. In *Proceedings of the IEEE ICCCN*.

[16] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. 2017. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics* 33, 1 (2017), 1–21.

[17] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. 2014. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA'14)*. IEEE, 15–22.

[18] Dorian Gálvez-López and Juan D. Tardos. 2012. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics* 28, 5 (2012), 1188–1197.

[19] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. 2018. LDSO: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'18)*. IEEE, 2198–2204.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE CVPR*.

[21] Ross Girshick. 2015. Fast R-CNN. In *Proceedings of IEEE ICCV*.

[22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE CVPR*.

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of IEEE CVPR*.

[24] Richard Hartley and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of IEEE ICCV*.

[26] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. 2016. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 4 (2016), 814–830.

[27] Georg Klein and David Murray. 2007. Parallel tracking and mapping for small AR workspaces. In *Proceedings of IEEE ISMAR*.

[28] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. G^2O: A general framework for graph optimization. In *Proceedings of IEEE ICRA*.

[29] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. 2009. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision* 81, 2 (2009), 155.

[30] Danyang Li, Yumeng Lu, Jingao Xu, Qiang Ma, and Zhuo Liu. 2019. iPAC: Integrate pedestrian dead reckoning and computer vision for indoor localization and tracking. *IEEE Access* 7 (2019), 183514–183523.

[31] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. 2017. Fully convolutional instance-aware semantic segmentation. In *Proceedings of IEEE CVPR*.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of ECCV*. Springer.

[33] Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. 2012. Push the limit of WiFi based localization for smartphones. In *Proceedings of ACM MobiCom*.

[34] Song Liu and Tian He. 2017. SmartLight: Light-weight 3D indoor localization using a single LED lamp. In *Proceedings of ACM Sensys*.

[35] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. 2006. Real time localization and 3D reconstruction. In *Proceedings of IEEE CVPR*.

[36] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. 2015. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671

[37] Raúl Mur-Artal and Juan D. Tardós. 2017. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters* 2, 2 (2017), 796–803.

[38] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of IEEE CVPR*.

[39] Pauline C. Ng and Steven Henikoff. 2003. SIFT: Predicting amino acid changes that affect protein function. *Nucleic Acids Research* 31, 13 (2003), 3812–3814.

[40] Pedro O. Pinheiro, Ronan Collobert, and Piotr Dollár. 2015. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*.

[41] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. 2016. Learning to refine object segments. In *Proceedings of ECCV*. Springer.

[42] Kun Qian, Chenshu Wu, Yi Zhang, Guidong Zhang, Zheng Yang, and Yunhao Liu. 2018. Widar2.0: Passive human tracking with a single Wi-Fi link. In *Proceedings of ACM MobiSys*.

[43] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. 2012. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of ACM MobiCom*.

[44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of IEEE CVPR*.

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. 91–99.

[46] Timothy H. Riehle, Shane M. Anderson, Patrick A. Lichter, Nicholas A. Giudice, Suneel I. Sheikh, Robert J. Knuesel, Daniel T. Kollmann, and Daniel S. Hedin. 2012. Indoor magnetic navigation for the blind. In *Proceedings of IEEE EMBC*.

[47] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of IEEE ICCV*.

[48] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly, and Andrew J. Davison. 2013. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of IEEE CVPR*.

[49] Longfei Shangguan, Zheng Yang, Alex X. Liu, Zimu Zhou, and Yunhao Liu. 2017. STPP: Spatial-temporal phase profiling-based method for relative RFID tag localization. *IEEE/ACM Transactions on Networking* 25, 1 (2017), 596–609.

[50] Y. Shu, Y. Huang, J. Zhang, P. Coué, P. Cheng, J. Chen, and K. G. Shin. 2016. Gradient-based fingerprinting for indoor localization and tracking. *IEEE Transactions on Industrial Electronics* 63, 4 (April 2016), 2424–2433.

[51] Yuanchao Shu, Kang G. Shin, Tian He, and Jiming Chen. 2015. Last-mile navigation using smartphones. In *Proceedings of ACM MobiCom*.

[52] William Storms, Jeremiah Shockley, and John Raquet. 2010. Magnetic field navigation in an indoor environment. In *Proceedings of IEEE UPINLBS*.

[53] Hauke Strasdat, José M. M. Montiel, and Andrew J. Davison. 2012. Visual SLAM: Why filter? *Image and Vision Computing* 30, 2 (2012), 65–77.

[54] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the IEEE IROS*.

[55] Xiaohua Tian, Ruofei Shen, Duowen Liu, Yutian Wen, and Xinbing Wang. 2016. Performance analysis of RSS fingerprinting based indoor localization. *IEEE Transactions on Mobile Computing* 16, 10 (2016), 2847–2861.

[56] Xiaohua Tian, Mei Wang, Wenxin Li, Binyao Jiang, Dong Xu, Xinbing Wang, and Jun Xu. 2017. Improve accuracy of fingerprinting localization with temporal correlation of the RSS. *IEEE Transactions on Mobile Computing* 17, 1 (2017), 113–126.

[57] Jasper R. R. Uijlings, Koen E. A. Van De Sande, Theo Gevers, and Arnold W. M. Smeulders. 2013. Selective search for object recognition. *International Journal of Computer Vision* 104, 2 (2013), 154–171.

[58] Jue Wang and Dina Katabi. 2013. Dude, where's my card? RFID positioning that works with multipath and non-line of sight. In *ACM SIGCOMM*.

[59] Chenshu Wu, Jingao Xu, Zheng Yang, Nicholas D. Lane, and Zuwei Yin. 2017. Gain without pain: Accurate WiFi-based localization with fingerprint spatial gradient. In *PACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*.

[60] Chenshu Wu, Zheng Yang, and Yunhao Liu. 2015. Smartphones based crowdsourcing for indoor localization. *IEEE Transactions on Mobile Computing* 14, 2 (2015), 444–457.

[61] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. 2012. WILL: Wireless indoor localization without site survey. *IEEE Transactions on Parallel and Distributed Systems* 24, 4 (2012), 839–848.

[62] Chenshu Wu, Zheng Yang, and Chaowei Xiao. 2018. Automatic radio map adaptation for indoor localization using smartphones. *IEEE Transactions on Mobile Computing* 17, 3 (2018), 517–528.

[63] Jingao Xu, Hao Cao, Danyang Li, Kehong Huang, Chen Qian, Longfei Shangguan, and Zheng Yang. 2020. Edge assisted mobile semantic visual SLAM. In *Proceedings of the IEEE INFOCOM*.

[64] Jingao Xu, Hengjie Chen, Kun Qian, Erqun Dong, Min Sun, Chenshu Wu, Li Zhang, and Zheng Yang. 2019. iVR: Integrated vision and radio localization with zero human effort. In *PACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*.

[65] Jingao Xu, Zheng Yang, Hengjie Chen, Yunhao Liu, Xianchun Zhou, Jinbo Li, and Nicholas Lane. 2018. Embracing spatial awareness for reliable WiFi-based indoor location systems. In *Proceedings of the IEEE MASS*.

[66] Zheng Yang, Chenshu Wu, and Yunhao Liu. 2012. Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Proceedings of ACM MobiCom*.

[67] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. 2015. Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys* 47, 3 (April 2015), Article 54, 34 pages.

[68] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. *ACM Computing Surveys (CSUR)* 46, 2 (2013), 25.

[69] Zuwei Yin, Chenshu Wu, Zheng Yang, and Yunhao Liu. 2017. Peer-to-peer indoor navigation using smartphones. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1141–1153.

[70] Yuanqing Zheng, Guobin Shen, Liqun Li, Chunshui Zhao, Mo Li, and Feng Zhao. 2014. Travi-Navi: Self-deployable indoor navigation system. In *Proceedings of ACM MobiCom*.

[71] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. 2019. Widar3.0: Zero-effort cross-domain gesture recognition with Wi-Fi. In *ACM International Conference on Mobile Systems, Applications, and Services*.

Received September 2020; revised November 2020; accepted January 2021