Improving the Applicability of Visual Peer-to-Peer Navigation with Crowdsourcing

Erqun Dong*, Jianzhe Liang*, Zeyu Wang*, Jingao Xu*, Longfei Shangguan†, Qiang Ma* and Zheng Yang*‡
*School of Software and BNRist, Tsinghua University, Beijing, China
Email: doneq13@gmail.com, thss15_liangjz@163.com, ycdfwzy@outlook.com,
xujingao13@gmail.com, thumaq@mail.tsinghua.edu.cn, hmilyyz@gmail.com
†Microsoft Cloud&AI, Seattle, WA, USA
Email: longfei.shangguan@microsoft.com
‡Corresponding Author

***Abstract*—Visual peer-to-peer navigation is a suitable solution for indoor navigation for it relieves the labor of site-survey and eliminates infrastructure dependence. However, a major drawback hampers its application, as the peer-to-peer mode suffers from a deficiency of paths in large indoor scenarios with multifarious places-of-interest. Nevertheless, we propose one with a profound crowdsourcing scheme that addresses the drawback by merging the paths of different leaders' into a global map. To realize the idea, we further deal with entailed challenges, namely the unidirectional disadvantage, the scale ambiguity, and large computational overhead. We design a navigation strategy to solve the unidirectional problem and turn to VIO to tackle scale ambiguity. We devise a mobile-edge architecture to enable real-time navigation (30fps, 100ms end-to-end delay) and lighten the burden of smartphones (35% battery life for 2h35min) while assuring the accuracy of localization and map construction. Through experimental validations, we show that P2P navigation, previously relying on the abundance of independent paths, can enjoy a sufficiency of navigation paths with a crowdsourced global map. The experiments demonstrate a navigation success rate of 100% and spatial offset of less than 3.2m, better than existing works.**

I. Introduction

Indoor navigation has been exhibiting promising applications in scenarios like large shopping malls, warehouse management, *etc.* The conventional way of indoor navigation is to localize a user in a global map and navigate the user therein. Techniques using Wi-Fi [1], [2], [3], RFID [4], [5] and sound [6] have demonstrated efficacy. However, these works require indoor radio maps, whose site-survey procedure costs considerable effort and time; moreover, they rely on indoor infrastructure either pre-existing or dedicatedly installed. Therefore, although they are inspiring, the dependency on site-survey and infrastructure makes their application less-than-ideal.

In contrast to the conventional way, peer-to-peer (P2P) navigation is free of site-survey cost [7]. It encourages a first user, namely a leader, to record the information of a path such as sensor readings at turning points. This information is shared with a second user, namely a follower, who wants to travel from the same origin to the same destination. The follower simply needs to reproduce the recorded information to follow the path. Several works leveraging wireless signals and inertial sensors show that this P2P mode can also successfully navigate [8], [9].

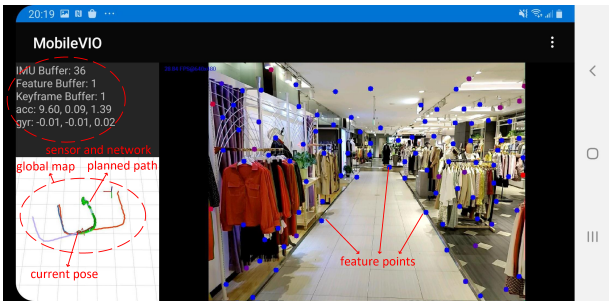


Figure 1: User Interface of the mobile side. On the left-bottom, the current pose of the follower is shown by a small red pyramid, the planned path is shown in green. On the left-top lies the sensor information and the network status. The current image the follower is looking at is on the right. Visual navigation is user-friendly because the follower can have a good understanding of his/her position and the path.

Now that the P2P mode solves the arduousness of site-survey, researchers come to eliminate the dependency on infrastructure, where computer vision methods seem rosy. Based on multi-view geometry, visual simultaneous localization and mapping (SLAM) endows navigation with a more accurate trajectory and map construction. Plus, visual navigation is more user-friendly, as images are richer in semantics than wireless signal readings so that users can easily understand the paths they are taking (See Figure 1). *Pair-navi* [10] has incorporated visual SLAM into peer-to-peer navigation. However, although *Pair-navi* works fine with motivated leaders, it has a relatively small path archive in large indoor scenarios. For example, shop owners would like to build paths from the gate of the mall to their shops, but paths between shops, much more essential to the customer experience in the shopping mall, are insufficient. This lack of paths largely vitiates the practicability of P2P navigation, but is seldom studied.

In this paper, we seek to push the limit of visual P2P navigation systems towards better applicability. Regrading path deficiency, intuitively, if we splice the paths of different

978-1-7281-9074-7/20/$31.00 ©2020 IEEE

leaders into other paths, we will obtain a more comprehensive path coverage. *FollowUs* [11] first embodied the idea of splicing paths in magnetic navigation. However, in visual navigation, this idea entails several challenges. **First**, monocular visual SLAM suffers from scale ambiguity, the meaning of which is two-fold: (1) Different trajectories have different unknown scales, making it impossible to splice them together; (2) Even within one trajectory, the scale can drift if the trajectory is very long without loop closure. Nevertheless, we still have to use monocular navigation because although some new smartphones are equipped with dual cameras, the cameras have inconsistent lens types and short baselines so that only monocular SLAM is usable. We resort to the IMU for visual-inertial odometry (VIO) to overcome scale ambiguity, as the IMU provides meter-unit measurements, thus being able to unify all trajectories within the same scale.

Second, visual paths has the unidirectional problem, which means the trajectories and map construction can only be reused in one direction, since the same spot has different scenes in opposite directions. Thus, we cannot use the spliced visual paths directly as in Pari-navi [10]. In order to overcome the unidirectional problem, we design a navigation strategy to track a user in a crowdsourced global map, in both the forward and inverse directions.

A **third** challenge is entailed in the resort to VIO. As VIO relies on high-dimensional matrix optimization, it can hardly run on smartphones in real-time, unless sacrificing accuracy, which should not be encouraged because to build a consistent global map, each local map needs to be as accurate as possible. Moreover, the battery consumption required to continuously run the optimization is a heavy burden on a smartphone. Therefore, reducing the computational overhead on mobile clients is necessary. We achieve this goal with a mobile-edge VIO architecture, carefully dividing the VIO pipeline into mobile and edge-server sides and devising data interaction to minimize both the overhead on the mobile side and the communication bandwidth while assuring accuracy and real-time response.

Based on the above ideas, we present a novel visual peer-to-peer navigation system, *GMPN* (Global Map P2P Navigation, the user interface shown in Figure 1). This paper has the following contributions:

- We make visual P2P navigation more applicable by boosting the abundance of paths with a profound crowdsourcing scheme, which transforms visual P2P navigation from the previously local-map-only, lacking-in-paths version to that with global maps and a sufficiency of navigation paths.
- We settle the unidirectional problem of visual paths in crowdsourced visual P2P navigation by an effective navigation strategy.
- We enable accurate real-time navigation on smartphones with lightened computational overhead by an edge-assisted visual-inertial odometry architecture.

Experiments demonstrate that besides comprehensive path coverage, our system can navigate with a navigation success rate of 100%. The spatial offset of our system is less than 3.2m in environments like shopping malls, supermarkets, teaching buildings, and office buildings, better than existing works. Plus, our mobile-edge architecture guarantees a lightweight burden on the smartphone (35% battery life for 2h35min) and real-time operation (30fps, as specified in [12], [13], and 100ms end-to-end delay).

The rest of this paper is organized as follows: we introduce the fundamentals of VIO in Section II; then we expatiate the navigation workflow in Section III and the mobile-edge architecture in Section IV; after that, we discuss the experiments and conclude the paper.

II. Preliminaries

Monocular VIO takes video frames and IMU readings as input, and outputs real-time poses of the camera and a local map of the 3D point landmarks. The system maintains a sliding window and solves an optimization problem to estimate the most recent camera poses and 3D landmarks within it. In the process, some keyframes will be selected from all the video frames, and collecting all the keyframes, we acquire a leader's trajectory associated with a local map, we call the combination of them a "sequence". Note that a sequence is denoted by two strings, its origin and destination name, which suggests an inborn direction.

How does the system splice sequence? Suppose there is already one sequence, which is regarded as the primitive global map. When a new sequence is made, the system will use relocalization to find out from existing sequences a keyframe which is the most similar to a keyframe in the new sequence. If relocalization succeeds, a pose transformation between the new sequence and the global map will be computed, and then we can transform the new sequence into the global map.

III. Navigation System Design

In this section, we show in detail how a global map is constructed through crowdsourcing and how we navigate a follower with our navigation strategy. Then we demonstrate the planning of new navigation paths.

A. Global Map Construction through Crowdsourcing

We consider three major situations about how sequences should be crowdsourced to form a global map, as illustrated in Figure 3. The illustrations only shows the chances for merging, regardless of the shapes of the sequences.

Situation 1. Leader 1 contributes seq1 from the Gate of the mall to shop A, which is regarded as the primitive global map. Leader 2 contributes seq2 from the Gate to shop B. In this case, since seq1 and seq2 contain a common segment (Gate$\rightarrow$C) with the identical direction, relocalization happens and seq2 is merged into seq1.

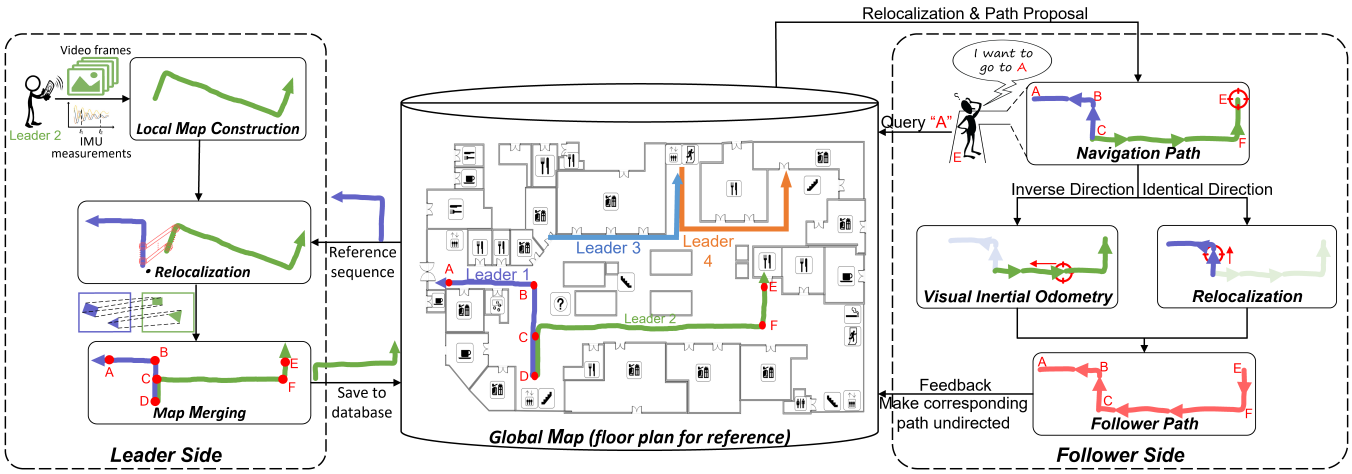


Figure 2: Workflow of a leader and a follower. Leaders and followers all interact with the global map. Leaders builds a local map and contributes it to the global map; followers use the global map to navigate, either by VIO in inverse directions or by relocalization in identical directions, and can also contribute to the global map.

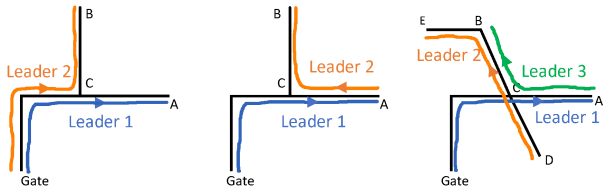


Figure 3: Three situations of the leader

Situation 2. Leader 1 remains the same. Leader 2 goes from A to B. Without other measures, even though two sequences share the same passage (AC), splicing is precluded because they are in opposite directions hence no relocalization. To splice more sequences, our system requires the leaders to face the camera towards the shopfronts at the beginning and in the end. Owing to this requirement, the paths will have co-vision at ends of the sequences and can be spliced together.

Situation 3. Leader 1 remains the same, and leader 2 contributes seq2 from shop D to shop B. Since no relocalization occurs, they can not be merged. In this case, we simply store seq2 because it may be merged into the global map through future sequences. When leader 3 contributes seq3 from A to B, it first detects relocalization with seq1 at A, and merges itself into the global map. Then, at C, seq3 relocalizes with seq2, and seq2 is merged in the global map at length.

In a nutshell, the three situations suggest that no matter what shape a path is in, it can contribute to the global map if: (1) it has relocalization with previous paths; (2) it has the same origin or destination with previous paths; (3) it can be connected with previous paths through future paths.

B. Navigation Strategy in Both Directions

For each video frame of the follower, *Pair-Navi* mainly carries out relocalization to localize the follower in the leader's trajectory. However, in *GMPN*, this strategy is not suitable because we need to support navigating in an inverse direction. If the follower's camera is facing towards the opposite direction of the sequence, no relocalization will happen. Therefore, we propose a VIO-based navigation strategy for followers.

A typical workflow of navigation. Suppose a follower is going from E to A (see Figure 2). He first selects the name of A from the known places of interest and queries it to the global map. Our system first instructs him to be relocalized in the global map. Then he travels on the path E→F→C in an inverse direction with the original path of leader 2 while our system runs a full VIO to track him and compute navigation paths accordingly. In practice, as VIO is in nature a dead-reckoning method, it gradually accumulates drifting error, however, if a relocalization is detected later, the location of the follower can be bound to the global map, resetting the drift to zero at that spot (here, a new cost term of the relocalization is added to the sliding window optimization problem).

When the follower turns right at C, his smartphone is relocalized because now the camera is facing toward the direction identical to the sequence. After this relocalization, the drifting error of this frame is corrected to zero and begins to accumulate from zero again. In a large global map, it is often the case that a follower will meet several relocalizations, so the drifting error of VIO is bounded and the system works fine. But although relocalization can correct drift, the performance of navigation in inverse direction is still crucial, which we will demonstrate in Section V.

In short, the navigation strategy is to carry out VIO the whole time, and whenever a relocalization happens, it corrects drift. After the navigation, the follower's path is fed back to the global map, just like a leader. In this way, followers help enrich the global map.

A perspective of graph theory. From a perspective of graph theory, since the sequences are innately directed,

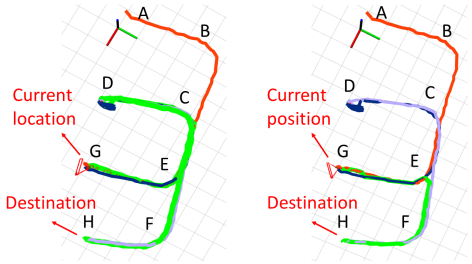


Figure 4: Path proposal methods (left) peer-to-peer path proposal; (right) active path proposal. The planned paths are shown in green lines in both cases. In peer-to-peer path proposal, a follower going from F to H needs to travel in the order "F→E→C→D→C→E→G→H", which is a huge detour; while in keyframe-level path proposal, the follower takes the nearest path "F→E→G→H".

the navigation takes place in a directed graph, where each path is supposed to be unidirectional. That being said, we manage to navigate even in an inverse direction with the slight disadvantage of VIO's drift. Furthermore, once the follower finishes walking in the inverse direction of a passage, and feeds back his sequence to the global map, this passage becomes undirected. In this way, the global map becomes more accurate for navigation, because we have more opportunities to relocalize rather than have to carry out VIO for a long time.

C. Path Planning

To plan navigation paths, we first need to find out if the destination is reachable from the follower's current position. We use a disjoint-set data structure [14] as well as a connected graph to manage the sequences. When a follower chooses a destination, *GMPN* immediately gets the destination sequence; and after relocalizing the follower in the global map, the original sequence is also known. Then *GMPN* carries out a disjoint-set search to determine the reachability of the destination sequence from the original sequence. If the destination sequence is reachable, then the destination spot is reachable, thereupon, *GMPN* can plan paths. It first searches for a sequence shifting order, *i.e.*, the series of sequences a follower should travel, then, take relocalization keyframes between sequences as shifting spots, *GMPN* puts the needed keyframes in order into a navigation path, which is illustrated in Figure 4 (left). In this way, *GMPN* plans a peer-to-peer path.

Unfortunately, peer-to-peer path planning is not always efficient. Still in the example of Figure 4 (left), if two sequences are merged at the ends, then the follower may need to detour. Therefore, we add another active path planning scheme that try to generate more efficient paths on a keyframe level. We maintain a K-D tree of the positions of all the keyframes. If the peer-to-peer path is longer than 3 times the line distance between the follower's current location and the destination keyframe, we search in the K-D tree on a keyframe level to see if there exists a shorter path. In the example of Figure 4 (right), the active path planning yields a shorter path, while the peer-to-peer path planning tells the follower to take a detour.

D. Comprehensiveness of Path Coverage by Crowdsourcing

In a pure P2P navigation system, the follower can be navigated only if a leader has walked in the path that the follower needs. In *GMPN*, this requirement is relaxed since two POIs are mutually reachable if there exists a series of intersections of sequences, regardless of the directions.

Suppose now we already have a global map, and one wishes to go to shop B from shop A. Currently, there is no leader's sequence directly connecting A and B, but there are 2 intersecting sequences, each of which have either A or B as its origin or destination. With our path planning scheme, we can plan a path from A to B with the two sequences. Furthermore, we can use more than 2 sequences to propose a path as long as the A and B are connected through a series of intersecting sequences.

IV. Mobile-Edge-Structured VIO

A. Design Insights

The main idea of our mobile-edge-structured VIO is to perform data capture and light-weight preprocessing on mobile devices while running heavy optimization on edge servers. We also carefully control the rate of data transmission to reduce computational overhead and minimize bandwidth usage.

We modify and extend *VINS-Mono* [15] for our system. It contains a vision frontend, a feature tracker which takes raw video frames as input, extracts feature points and performs optical-flow to track them. Meanwhile, the IMU preintegration module takes IMU readings and compute preintegrations between each consecutive pair of video frames. Both visual and IMU information are fed into a sliding window module, which optimizes the current pose of the smartphone as well as the 3D positions of the feature points, keeping tracking of the latest position of a leader or a follower. Finally, a sequence manager module takes charge of the global map and the current sequence, detects whether there is a relocalization of the current sequence and global map, and merges them if possible. There have been works on putting visual SLAM on the edge servers [10], [16], but putting visual-inertial odometry on the edge is different.

The edge side. Though the original *VINS-Mono* provides high accuracy for pose estimation and local map construction, it heavily depends on computation-intensive optimization, thus difficult to run on ordinary smartphones of limited computational ability and battery power. Furthermore, it also has a relatively high demand for storage, because to detect relocalization, the sequence manager first needs to construct a keyframe database. It is easy to think of curtailing the length of the sliding window and reducing the amount of

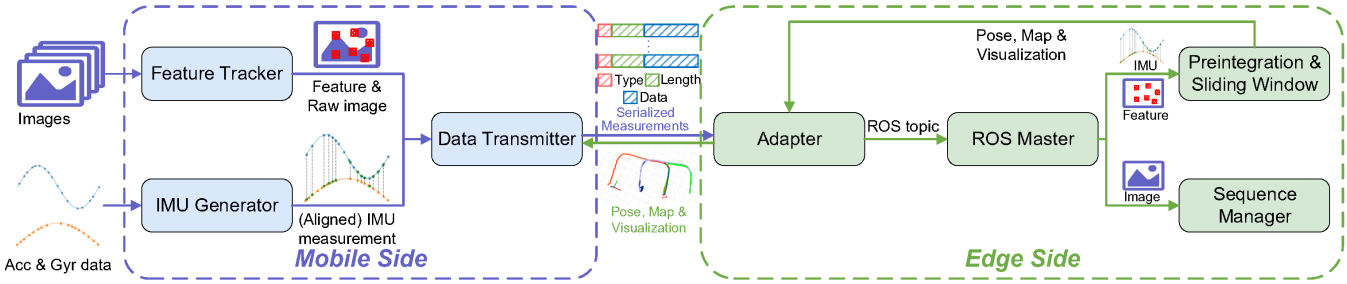


Figure 5: Architecture of edge-assisted VIO. We show the mobile side on the left and the edge side on the right. The mobile side captures images and IMU and send them to the edge side, while the edge side processes the information, manages the sequences and sends navigation information back to the mobile side.

keyframe in the database. While this may help mitigate the burden on smartphones, it can take a toll on localization and mapping accuracy, which is the worst for our system because to build a globally consistent map, all local maps need to be as accurate as possible. Thus, to truly ease the burden of smartphones without losing accuracy, we put the sliding window, sequence manager modules on the edge side.

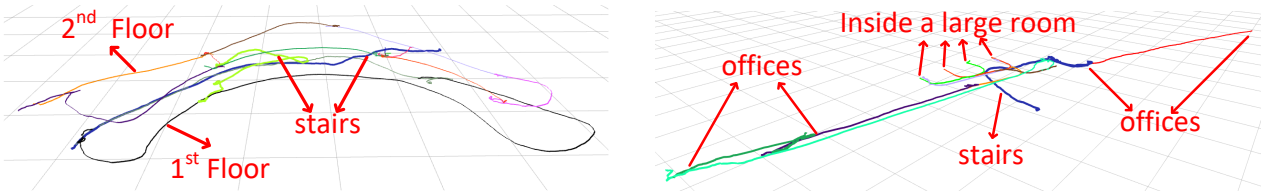
The mobile side. At first glance, a light-weight mobile client should solely collect images and IMU data and send them to an edge server. However, such a solution needs a considerable amount of bandwidth due to frequent transmission of image data. For a VIO to be able to track, the frame rate should be at least 10fps, which adds up to a prohibitive bandwidth of 6.1MB/s. Luckily, the sliding window only requires a set of feature points as visual measurements. With a camera taking 640×480-resolution images at 30fps (for better tracking and mapping accuracy under motion clutter) and each frame containing at most 100 feature points, the total bandwidth for transmitting feature points is 180KB/s, outperforming 6.1MB/s at 10fps for raw image transmission. Therefore, we extract the feature points on the mobile side and transmit them to the edge side. The whole procedure on the mobile side works in the following manner. First, we extract feature points for every frame; after that, we perform optical-flow tracking to match the points to those in the previous frame, and further perform a culling procedure using RANSAC with fundamental matrix model [17] (we will refer to this as "fundamental culling"), which winnows out the badly tracked points that plague the accuracy of VIO. Then the feature points are transmitted to the edge side. After that, we extract points again, to complement the total number of points for doing optical-flow with the next frame (we will refer to this as "complementary extracting"). Nevertheless, though features points are sufficient for the sliding window such that raw images are not needed here, we still need raw images for relocalization, so we also send raw images back to the edge-server at a low frequency, not to congest the bandwidth.

As for the last module, the IMU preintegration module, we notice that although an IMUs produce data at a rather high rate (100˜300Hz), each entry contains only several bytes of data, which means they account for only a minor portion of the total bandwidth. Also, the raw IMU readings are needed in the sliding window optimization: because the preintegration should be recomputed after each optimization iteration, so the IMU readings are repeatedly used on the edge-server. Therefore, we transmit directly back to the edge the raw IMU readings, which is comprised of accelerometer and gyroscope readings firstly aligned through an IMU generator.

Pushing the limit. With the basic architecture proposed above, we can further push the limit of the system by some more detailed tailoring strategies. We notice that pedestrians usually move slowly (about 1.3m/s), so considering image rate ($\geq$ 30fps), the parallax between consecutive frames is small. Therefore, we do not need to transmit back the feature points in a frame-by-frame manner, but can rather do it intermittently. Regarding the frames that we do not transmit back to the edge server, we simply continue the optical-flow tracking for them on the smartphone and do not need to perform fundamental culling or complementary extracting. This frugality of transmission also brings down the frequency of fundamental culling, reducing image preprocessing overhead; however, the reduction of fundamental culling can also impair tracking accuracy due to poorer feature quality. Therefore, we need to measure the influence of feature transmission rate on tracking accuracy, image processing time, and bandwidth usage, and choose a feature transmission rate in different scenarios.

B. *Implementation*

We implement our mobile side on Android platform. The mobile consists of 3 main threads, a feature tracker, an IMU generator and a data transmitter. The feature tracker extracts feature points from images and controls the transmission rate for feature points (for VIO tracking) and raw images (for relocalization). The IMU generator aligns measurements from different sensors. Since Android provides accelerometer and gyroscope measurements separately, they usually have different timestamps. For each measurement entry, the IMU generator finds its temporal neighbor and performs interpolation to generate corresponding measurements. An-



(a) (b)

Figure 6: Two global maps created in our experiment in two indoor scenarios: (a) two floors of a teaching building; (b) one floor of an office building

other problem is that some devices have different clock bases for IMU and camera (those not supported by Google ARCore certification), so we also synchronize the two clock bases by comparing them with system time for these devices. Lastly, the data transmission thread serializes each data entry into a lightweight "type-length-data" format, and sent it to the edge side through a single socket.

On the edge side, we build the system upon the ROS platform. An adapter first deserializes the data from the mobile side and sends each type of data (feature, IMU, raw image) through respective ROS topics. Then the preintegration module, the sliding window and the sequence manager receive the ROS topics and carry out their functions.

V. Experiments

In this section, we evaluate the performance of our system both in navigation and in edge-assisted communication.

We implement our system on the ROS platform. The VIO module, modified and extended from *VINS-Mono* [15], uses Ceres 1.14.0 for optimization, Eigen 3.3.4 for matrix operation, and OpenCV 3.4.2 and OpenCV Android 3.4.2 for image processing on edge-server and smartphone respectively.

Experiment settings. We asked 4 volunteers to act as leaders and followers in turn, whose heights varied from 172cm to 189cm, and whose behaviors of holding phones were varied, some holding steadily and others shakily.

The mobile phones we used were Samsung Galaxy S10+ and Huawei P10, and the edge server is Dell XPS 9560 with i7-7700HQ CPU of 2.8GHz main frequency and 16G RAM, installed with Ubuntu 16.04 operating system and ROS Kinetic.

We selected a shopping mall, a teaching building, a supermarket and an office building as our experiment venue, the area of which is about $2000m^2$, $1000m^2$, $500m^2$ and $400m^2$, respectively.

Path coverage. Examples of the final global maps are shown in Figure 6. We can see all the places are connected and mutually reachable in the global map, thus validating a far more comprehensive path coverage. Specifically, inside a large room with 4 POIs and 3 spliced sequences, the total path number is 12 compared to 3 sequences(Figure 6b); In a teaching building, where there are 8 POIs on the 2nd floor and 3 POIs on the 1st floor, our system generates 110 available paths, much larger than 13, the number of sequences. Thus, the path coverage grows considerably fast, making *GMPN* much more applicable than the pure-P2P visual navigation system *Pair-navi*.

Spatial offset. We measure the spatial offset between the checkpoint and the follower's current location when he reaches the checkpoint. The spatial offset reflects the metric error where a navigation event like turning should happen and is suitable for measuring navigation performance. It is measured directly in Rviz on the ROS platform.

We compare the spatial offset of our system with [9], [11]. Note that in our system, after each relocalization, the follower's location is forcibly aligned to the leader's trajectory owing to the optimization. When this happens at a checkpoint, the spatial offset there is zero. If our system receives more sequences, there might be more opportunities of relocalization and more spatial offsets will become zero. So we only conduct experiments on a freshly built map without much trajectory overlapping or follower's enrichment to the map. The results are shown in Figure 7.

In normal environments with relatively rich texture, *i.e.*, the shopping mall, the supermarket and the teaching building, our system outperforms others. In all cases, it has a spatial offset of less than 3.2m, and in 90% of the cases, the spatial offset is less than 1.1m. Comparing to magnetic navigation methods *FollowMe* and *FollowUs*, our system shows better accuracy. The spatial offset of *Pair-Navi* is not shown here because the whole idea of *Pair-Navi* is based on relocalization, and therefore it has zero spatial offset.

Extreme conditions We further experiment the navigation accuracy in an extreme condition, namely inverse directions. Again, we use spatial offset to measure system performance. The result is shown in Figure 8.

We build several long sequences with checkpoints, from 30m to 150m, and ask followers to walk in the inverse direction with the sequences and measure the spatial offset at the checkpoints. Since there exist some textureless scenes, the overall performance is plagued by large errors. But for 80% cases, the single way performance is comparative to *FollowMe* and *FollowUs*. Considering in a mature global map, 30m is a very large length for any passages, so the overall performance in inverse directions is acceptable.

Comparing to *Pair-Navi* that is not able to work in inverse directions, our system achieves far better applicability. Not

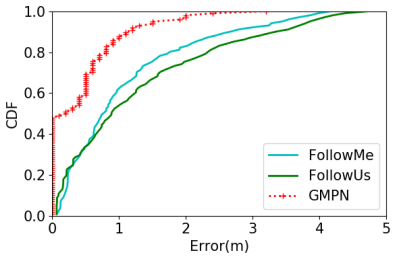


Figure 7: Spatial offset comparison between *GMPN* and some others

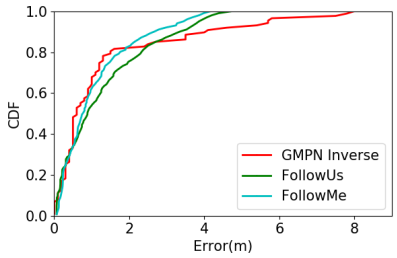


Figure 8: Spatial offset on inverse passages

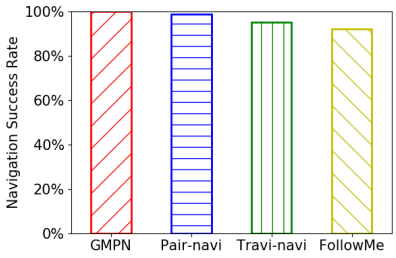


Figure 9: Navigation success rate of our system and several other systems

to mention that with our crowdsourcing strategy, the performance will become better and better, for when more and more paths are contributed to the global map, our system will be able to detect more and more relocalizations rather than accumulate drift.

Navigation success rate. We also test navigation success rate, which is defined as the rate of successful arrival at the checkpoints. We set 143 checkpoints in 40 navigation paths, and let followers go to their destinations without apprising of them the location of the destination or the checkpoints. With the help of our system, all the followers arrive at their destinations, and successfully pass 100% of the checkpoints, which is better than *Pair-navi* (98.6%), *Travi-navi* (96%) and *FollowMe* (92%). Even in textureless environments where the spatial offsets were large, the followers managed to pass the checkpoints because the textureless environments are often long aisles or staircases that do not have option paths. Here, we do need to point out some special cases in the shopping mall where the upward and downward escalators between two floors are placed in opposite sides such that the follower cannot follow the same path when he travels in the inverse direction. However, he passes the escalator and continues on the right track anyway, and finally makes it to the destination, because from the visualization of the global map, he understands that he should take the escalator on the opposite side. Therefore, these cases are counted as successful passes of the escalator checkpoints.

Battery consumption. We tested battery consumption on smartphones. We use 2.4GHz Wi-Fi for data transmission, set screen brightness comfortable to eyes, and run as less applications in the background as possible. The Samsung Galaxy S10+ ran for 2h35min, and the energy decreased from 100% to 65%. The Huawei P10 ran for 2h9min, and the energy decreased from 62% to 32%. Considering that a typical indoor navigation takes about 10min, the battery consumption on smartphones is acceptable.

Real-time Performance In our field experiment on navigation performance, we observed that the IMU buffer size (shown in top-left of Figure 1) on mobile client averages 30. Note that the Samsung Galaxy S10+ we used generates IMU data faster than 300Hz. We also found empirically that the processing delay at the edge side is neglectable due to richer computation resources. Ignoring data buffering at routers, *GMPN* is able to perform localization and mapping with end-to-end delay at $\frac{30}{300} = 100ms$, indicating it can navigate users in real-time.

VI. Related Work

Indoor P2P Navigation. Traditional indoor navigation systems navigate users in a global map with a floor plan and infrastructures (*e.g.*, Wi-Fi [3], Bluetooth, RFID [4], *etc.*), and recently some works try to solve the dependence on a pre-existing floor plan using wireless methods [18], [2] or surveillance cameras [19], [20]. On the contrary, P2P navigation, such as *Travi-Navi* [8], *FollowMe* [9] and *ppNav* [7], does not rely on a global map but rather navigates users in relevant local maps contributed by followers. *Pair-Navi* [10] exploits the power of vision, which is infrastructure-free and efficient. *FollowUs* [11] first exploits the idea of splicing trajectories, and obtained more trajectories. However, its localization method may be less-than-ideal because a follower first needs to walk for some distance to acquire a segment of magnetometer readings for segment matching, but, visual methods can perform immediate localization by visual relocalization. Plus, the comprehensiveness of path coverage, though seemingly fundamental at first glance, is hardly discussed by previous works like [11] that adopt such a scheme. In our work, we exploit the idea of merging local maps with visual approaches, and give a theoretical proof and experimental evaluation of the large path archive.

Visual-Inertial Odometry. Works on visual-inertial odometry can be classified into two categories, the loosely coupled [21] and the tightly coupled [22]. The loosely coupled is comparatively lightweight and efficient, but the accuracy may be unsatisfactory in our applications where a global consistent map relies on the accuracy of each local map. The tightly coupled have several prototypes, including EKF-based approaches [23] and optimization-based approaches [24]. With the advance of preintegration methods [24], IMU observations can be optimized together with visual observations, enabling complete systems to be more accurate, among which *VINS-Mono* [15] is one of the state-of-the-art systems. Though *VINS-Mono* [15] achieves satisfactory accuracy, it can not run in most smartphones in real-time without sacrificing accuracy, especially on Android

phones. We modify the system and extend it into a mobile-edge structure to assure real-time performance without sacrificing accuracy.

VII. Conclusion

We present a visual peer-to-peer navigation system named *GMPN* that achieves better applicability than existing peer-to-peer navigation works in a more comprehensive path coverage. It constructs a global map with a profound crowdsourcing scheme and navigates followers with an abundance of spliced paths. To realize the idea, we further deal with entailed challenges, namely the unidirectional disadvantage, the scale ambiguity, and large computational overhead. We design a navigation strategy to solve the unidirectional problem and turn to VIO to tackle scale ambiguity. The system is powered by a mobile-edge architecture of VIO. Experiments show that besides better path coverage, the navigation performance of our system is delightful. Future works may include improving the VIO system for better performance in textureless environments.

References

[1] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the IEEE INFOCOM*, 2000.

[2] C. Wu, Z. Yang, and Y. Liu, "Smartphones based crowdsourcing for indoor localization," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 444–457, 2015.

[3] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of WiFi based localization for smartphones," in *Proceedings of the ACM MobiCom*, 2012.

[4] J. Wang and D. Katabi, "Dude, where's my card? RFID positioning that works with multipath and non-line of sight," in *ACM SIGCOMM*, 2013.

[5] L. Shangguan, Z. Yang, A. X. Liu, Z. Zhou, and Y. Liu, "Stpp: Spatial-temporal phase profiling-based method for relative rfid tag localization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 596–609, 2017.

[6] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the ACM MobiCom*, 2009.

[7] Z. Yin, C. Wu, Z. Yang, and Y. Liu, "Peer-to-peer indoor navigation using smartphones," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1141–1153, 2017.

[8] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-navi: Self-deployable indoor navigation system," in *Proceedings of the ACM MobiCom*, 2014.

[9] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *Proceedings of ACM MobiCom*, 2015.

[10] E. Dong, J. Xu, C. Wu, Y. Liu, and Z. Yang, "Pair-navi: Peer-to-peer indoor navigation with mobile visual slam," in *Proceedings of the IEEE INFOCOM*, 2019, pp. 1189–1197.

[11] Y. Shu, Z. Li, B. Karlsson, Y. Lin, T. Moscibroda, and K. Shin, "Incrementally-deployable indoor navigation with automatic trace generation," in *Proceedings of the IEEE INFOCOM*, 2019, pp. 2395–2403.

[12] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proceedings of the ACM MobiCom*, 2019.

[13] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[14] B. A. Galler and M. J. Fisher, "An improved equivalence algorithm," *Communications of the ACM*, vol. 7, no. 5, pp. 301–303, 1964.

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] J. Xu, H. Cao, D. Li, K. Huang, C. Qian, L. Shangguan, and Z. Yang, "Edge assisted mobile semantic visual slam," in *Proceedings of the IEEE INFOCOM*, 6-8 July 2020.

[17] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-Markie: indoor pathway mapping made easy," in *Proceedings of USENIX NSDI*, 2013.

[19] L. Dong, J. Xu, G. Chi, D. Li, X. Zhang, J. Li, Q. Ma, and Z. Yang, "Enabling surveillance cameras to navigate," in *Proceedings of the IEEE ICCCN*, 3-6 Auguest 2020.

[20] J. Xu, H. Chen, K. Qian, E. Dong, M. Sun, C. Wu, L. Zhang, and Z. Yang, "ivr: Integrated vision and radio localization with zero human effort," in *PACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Sep 2019.

[21] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3923–3929.

[22] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization," in *Proceedings of Robotis Science and Systems (RSS)*, 2013.

[23] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3565–3572.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.