Edge Assisted Mobile Semantic Visual SLAM

Jingao Xu[1], Hao Cao[1], Danyang Li[1], Kehong Huang[1], Chen Qian[2], Longfei Shangguan[3], Zheng Yang[1]*
[1]School of Software and BNRist, Tsinghua University
[2]Dalian University of Technology
[3]Microsoft
*Corresponding author
{xujingao13, lidanyang1919, chen.cronus.qian, hmilyyz}@gmail.com
{caoh19, hkh18}@mails.tsinghua.edu.cn longfei.shangguan@microsoft.com

***Abstract*—Localization and navigation play a key role in many location-based services and have attracted numerous research efforts from both academic and industrial community. In recent years, visual SLAM has been prevailing for robots and autonomous driving cars. However, the ever-growing computation resource demanded by SLAM impedes its application to resource-constrained mobile devices. In this paper we present the design, implementation, and evaluation of *edgeSLAM*, an edge assisted real-time semantic visual SLAM service running on mobile devices. *edgeSLAM* leverages the state-of-the-art semantic segmentation algorithm to enhance localization and mapping accuracy, and speeds up the computation-intensive SLAM and semantic segmentation algorithms by computation offloading. The key innovations of *edgeSLAM* include an efficient computation offloading strategy, an opportunistic data sharing mechanism, and an adaptive task scheduling algorithm. We fully implement *edgeSLAM* on an edge server and different types of mobile devices (2 types of smartphones and a development board). Extensive experiments are conducted under 3 data sets, and the results show that *edgeSLAM* is able to run on mobile devices at 35fps frame rate and achieves a 5cm localization accuracy, outperforming existing solutions by more than 15%. We also demonstrate the usability of *edgeSLAM* through 2 case studies of pedestrian localization and robot navigation. To the best of our knowledge, *edgeSLAM* is the first real-time semantic visual SLAM for mobile devices.**

I. Introduction

Indoor localization and navigation play a key role in many location-based services. However, due to the excessive signal attenuation and multi-path propagation [1]–[4], the Global Positioning System (GPS) fails to achieve desirable accuracy and thus cannot be adopted for this purpose in most indoor scenarios. While the innovations on RF-based indoor localization techniques (*e.g.*, Wi-Fi, RFID and Bluetooth) are going full steam ahead [5]–[10], few of these solutions are mature for real-world deployment, either because of the low accuracy or high infrastructure cost.

As another promising alternative, vision-based indoor localization techniques, in particular visual simultaneous localization and mapping (visual SLAM) attracts more attentions in recent years [11], [12]. Visual SLAM utilizes a sequence of images captured by a camera and inertial measurement unit (IMU) readings to rebuild the map of ambient environment as well as estimate the current location of the camera itself in a local view. Compared with RF-based solutions, visual SLAM achieves an order of magnitude higher localization accuracy (5cm) at the minimal infrastructure cost as camera and IMU units have become the standard components of mobile devices on today's market [13].

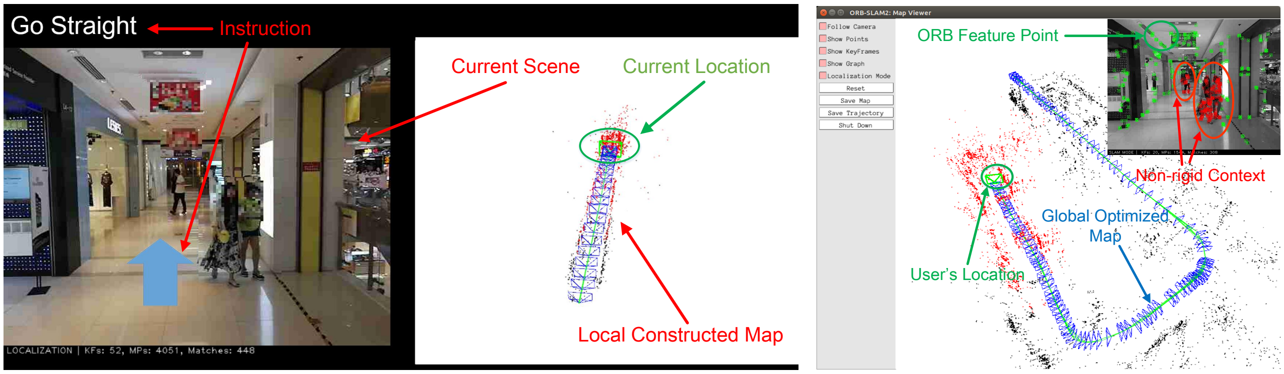
While the evolving hardware and software of mobile devices (*e.g.*, the latest Samsung Galaxy S10 smartphone is equipped with three HD cameras) guarantees the image quality and IMU sensor precision for fine-grained visual SLAM, the computational resource on mobile devices, unfortunately, is still insufficient to meet the visual SLAM's ever-growing computation demand. For example, as we experimentally demonstrated, even ORB-SLAM [12], a light-weight, functional-constrained visual SLAM algorithm, still cannot work in real-time (*i.e.*, $\leq$15 fps) on the latest smartphone (*e.g.*, Galexy S10 and Google Pixel 2). Blindly applying visual SLAM to mobile scenarios, on the other hand, may not achieve good performance because of highly dynamic envrionmental changes (e.g., customs in a shopping mall).

Recently, two new opportunities have arisen in the design of real-time Visual SLAM on mobile devices:

1) The emerging paradigm of edge computing [14], [15], as well as advanced wireless technology (5G [16] and Wi-Fi 802.11ad standard [17]), is powerful for solving computation-intensive tasks locally and in real-time. It is thus possible to speed up the visual SLAM by offloading the workload to an edge sever.
2) The evolving computer vision (CV) techniques (*e.g.*, Mask-RCNN [18]) now can recognize objects in an image with very high accuracy. It is thus possible to analyze semantic information from captured videos and improve SLAM performance.

However, realizing these intuitions is non-trivial and faces three significant challenges:

- **Task decomposition**. Simply transmitting all images back to an edge server is not feasible since it will introduces excessive bandwidth cost and transmission delays (details in Section II-B). Partitioning both visual SLAM algorithm and semantic segmentation algorithm into unit tasks, however, is also non-trivial as both their functional units are tightly coupled. As shown in Fig. 2, an improper partition may result in redundant data storage, exchange, and most importantly, system delay, which definitely in turn increases the algorithm latency.
- **Task cooperation**. The visual SLAM and semantic segmentation algorithms are generally regarded as two independent tasks, without the reuse of intermediate results. However, to achieve both low-latency and high accuracy, it is beneficial to share the intermediate results between



(a)

(b)

Fig. 1. User interface of *edgeSLAM*. (a) Mobile part: Navigation instruction and local constructed map are displayed on mobile device. [1] (b) Edge server part: Optimized global map and user's location are displayed.

these two algorithms so that redundant computations can be eliminated or minimized.

- **Task scheduling**. The computation resource on mobile device and edge server is highly unbalanced. Meanwhile the wireless link between these two parts also varies from time to time. Task scheduling strategy should be adaptive to the dynamics in computation resource and wireless link quality.

In this paper we present the design and implementation of *edgeSLAM*, a real-time **edge** assisted semantic visual **SLAM** service running on commercial mobile devices. *edgeSLAM* leverages the state-of-the-art semantic segmentation algorithm Mask-RCNN [18] to improve SLAM accuracy and speed up the SLAM and semantic segmentation algorithm by efficient computation offloading and data sharing, and adjust the offloading strategy automatically to adapt to the wireless link conditions.

To find out the optimal task decomposition strategy, we take the operation time, memory overhead and the transmission delay of each functional module into consideration and conduct extensive experiments to profile the performance of each module. We further analyze dependencies among these functional module and determine the ”hourglass position” to decompose the Visual SLAM and object detection algorithm.

To minimize the latency introduced by redundant data transmission, we leverage the fact that the scene in most consecutive frames are similar. For example, the same billboard tends to appear in multiple consecutive frames. *edgeSLAM* avoids per-frame object segmentation operation and reuses the previous result from the last object segmentation until a significant frame change is detected on mobile devices.

To accommodate dynamic link conditions, we design a probing-optimizing strategy that first probe the network conditions and then leverages such information to optimize our task scheduling mechanism.

We fully prototype *edgeSLAM's* server side on an Ubuntu edge server and the client side on three different types of mobile devices including an Nvidia Jetson TX2 development board, a Samsung Galaxy S10 and an Apple iPhone X. The interface of *edgeSLAM* is shown in Fig. 1. Comprehensive experiments are carried out under different network conditions, such as WiFi-5G, WiFi-2.4G and Cellular-4G. We also examine *edgeSLAM* on two official datasets (TUM [20] and KITTI [21]) and a self-labeled dataset from three buildings. The results demonstrate that *edgeSLAM* could achieve average 35fps frame rate with 5cm localization accuracy and 2% relative mapping error in all scenarios, which outperforms existing systems by more than 10%. Our two case studies further demonstrates that *edgeSLAM* achieves outstanding performance in pedestrian localization task with average 9.6cm accuracy and robot navigation task with 92.3% navigation success rate.

The key contributions are summarized as follows:

- We measured the operation time and memory overhead of each function module in visual SLAM and semantic segmentation, and ascertained the optimal decoupling position and disassembly method.
- We designed the system architecture to make SLAM and semantic segmentation deeply fused and determined task assignment between mobile and edge. To the best of our knowledge, this is the first time that *mobile semantic visual SLAM* can work in real-time.
- We take network condition into consideration and adopt an adaptive method to dynamically adjust system parameters.
- We implement a complete edge assisted real-time system on smartphones and extensively evaluate the performance. The results show that *edgeSLAM* achieves delightful results in all scenarios.

The rest of this paper is organized as follows. We introduce the background and motivation of our work in Section II. Followed by an overview of *edgeSLAM* in Section III. Key strategies and techniques are presented in Section IV, followed by the dynamic design of self-adaptation strategy in Section V. Related works are presented in Section VII. We finally conclude our work in Section VIII.

[1] The design of user interface for mobile part adapted from our previous work *Pair-Navi* [19].

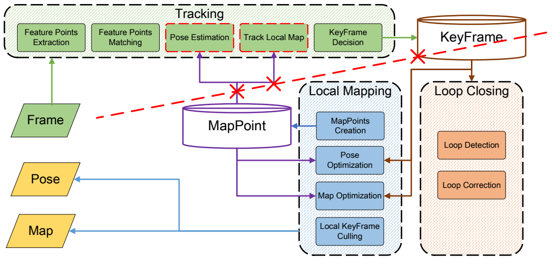


Fig. 2. Challenges of task assignment for semantic visual SLAM, where function units are tightly coupled.

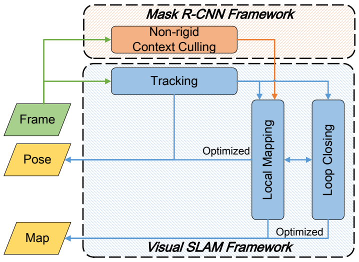


Fig. 3. System architecture of Semantic Visual SLAM. Figure referred from [19].

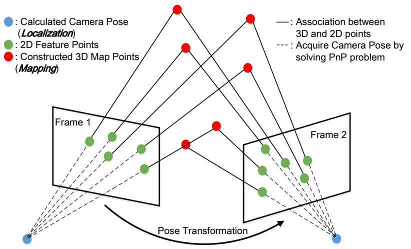
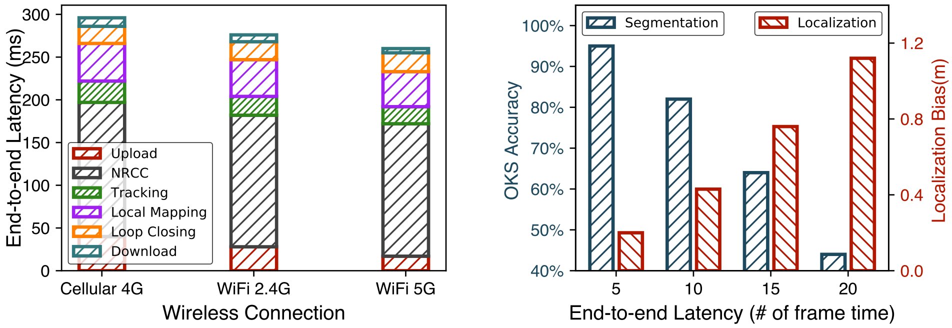

: Calculated Camera Pose (Localization)
: 2D Feature Points
: Constructed 3D Map Points (Mapping)
: Association between 3D and 2D points
: Acquire Camera Pose by solving PnP problem
Frame 1
Frame 2
Pose Transformation

Fig. 4. Illustration of Visual SLAM



(a)

(b)

Fig. 5. Accuracy and Latency Analysis. (a) Localization and segmentation accuracy with respect to different end-to-end latency. (b) End-to-end latency with different wireless connection link

II. Background

A. Semantic Visual SLAM

Semantic visual SLAM consists of four components: *tracking*, *local mapping*, *loop closing*, and *non-rigid context culling (NRCC)*, as shown in Fig. 3. We briefly introduce each module.

- **Tracking** module estimates the coarse-grained pose of the shooting camera based on the consecutive video frames. When a new video frame arrives, the tracking module will extract its 2D feature points and associate them with 3D map points in storage. As illustrated in Fig. 4, the 2D-to-3D feature points matching will give us a rough camera pose on the current frame.
- **Local mapping** module then creates a new 3D map points via triangulation between two consecutive frames [22]. An optimized camera pose can be then obtained by solving a Bundle Adjustment problem. This modules runs repeatedly as the camera takes more photos, resulting in a trajectory of the camera pose, a map of the 3D landmarks and the corresponding key frames.
- **Loop closing** module compares the features extracted from a video frame with keyframes. If a keyframe is found to be similar enough to the input video frame, the loop closing module will then fine-tune the current camera pose and optimize the map construction based on the matching result.
- **NRCC** module extracts the temporal objects (such as pedestrians) on each video frames to minimize their negative effect on both localization and mapping.

B. Latency and Accuracy Analysis

Running semantic visual SLAM on mobile devices is challenging due to its extensive computation overhead. For example, the classical visual SLAM algorithm ORB-SLAM [23] along works in a rate of only 10fpswhen running on a Samsung Galaxy S10 smartphone, far away from the real-time requirement ($\geq$ 30 fps [19], [24]). This frame rate will drop further when the semantic segmentation algorithm (*e.g.*, Mask-RCNN) runs in parallel to SLAM on mobile devices. Offloading the entire computation to the powerful edge server, on the other hand, may cause considerable latency. To better understand this transmission latency and its impact on the localization and object segmentation performance, we conduct experiments detailed below.

Latency Analysis. We model the end-to-end latency (from capturing a video frame until we obtain a camera pose) of a semantic visual SLAM solution as follows:

$$\begin{aligned} t_{\text{e2e}} &= t_{\text{upload}} + t_{\text{infer}} + t_{\text{download}} \\ t_{\text{infer}} &= \max(t_{\text{NRCC}}\,, t_{\text{T}}) + t_{\text{LM}} + t_{\text{LC}} \end{aligned} \tag{1}$$

where t_{upload} and t_{download} represents the delay for uploading an image to an edge server and downloading the result to the mobile device, respectively. t_{infer} represents the delay of running semantic visual SLAM on an edge server, which consists of the time consumed for semantic segmentation t_{NRCC}, tracking t_{T}, local mapping t_{LM}, and loop closing module t_{LC}, respectively.

We measure the latency of each functional module in various wireless link connections, and show the result in Fig. 5a. From the result we can see that offloading the entire

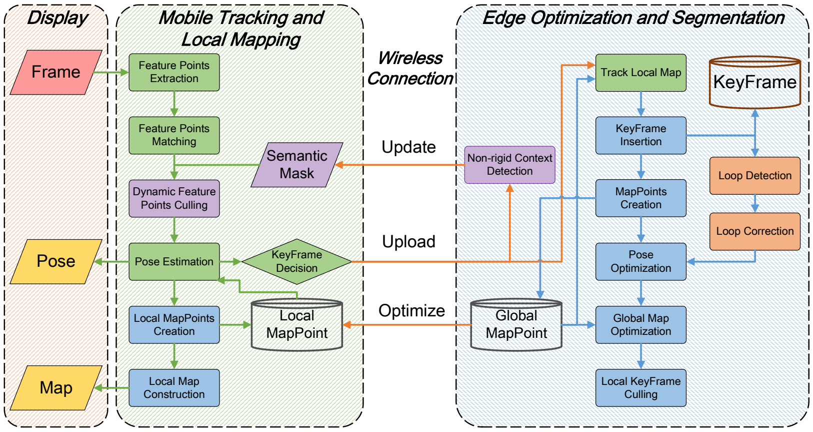


Fig. 6. *edgeSLAM* architecture

computation to an edge server introduces significant latency (280ms on Cellular 4G link, 270ms on Wi-Fi 2.4G link, and 260ms on Wi-Fi 5G link, respectively). In other words, we deserve only 4fps over all three wireless links. While SLAM and semantic segmentation can be further accelerated by leveraging more advanced edge server, the uploading and downloading latency, on the other hand, still takes 48ms, which set an upper bound of the achievable frame rate (20fps given the ignorable inference latency).

Accuracy Analysis. We further conduct experiment to understand the impact of different delays on the localization accuracy and semantic segmentation accuracy. In this trail of experiments, we record the 3-D position of the camera reported by the visual SLAM algorithm, and calculate the distance between the inference result and the ground-truth (measured by a laser ranger) using bias Euler-distance metric. We also use the Object Keypoint Similarity (OKS) [25] metric to measure the accuracy of keypoints in each group in the object keypoint segmentation task.

Fig. 5b shows the localization and semantic segmentation accuracy as a function of end-to-end delay. We observe that the localization error maintains in a very low level when the end-to-end latency is low (*i.e.*, five frames delay). The localization error then jumps to over 1m as we increase the end-to-end delay to 20 frames (about 666ms). The segmentation accuracy shows the similar trend: it decreases from over 90% accuracy to less than 15% accuracy as we increase the end-to-end delay from 5 frames to 20 frames. This result clearly demonstrates that the end-to-end latency has a significant impact on both two algorithms' performance.

III. System Architecture

To overcome these limitations, we propose a *edgeSLAM*, a semantic visual SLAM system for mobiles, achieving both accuracy and real-time simultaneously with the help of edge computation resource. To hide the latency caused by offloading the semantic SLAM task, *edgeSLAM* decouples the integrated visual SLAM process into two separate pipelines. At a high level, as shown in Fig. 6, these two parts are connected through a wireless link: *Mobile tracking and Local Mapping* part on a mobile device (smartphone for people or development board for robots) and *Edge Optimization and Segmentation* part on edge side. The former pipeline starts to simultaneously track the pose of a camera and construct a local map, while the later pipeline segments image frames at the pixel level, optimizes the rough pose, and further maintains a global map. Communications between two sides occur when keyframes are selected by mobile pipeline. Keyframes will be uploaded to edge side. Then, the edge server will send the optimized pose and map back to mobile side, as well as the segmentation result of the keyframe, which will be further used for semantic mask transfer among video clips. Once the optimization information is received by the mobile client, current camera pose and the local map will be calibrated and shown in current scene. The procedure is executed in parallel with mobile pipeline. In this way, the mobile client is capable of continuously tracking and displaying the accurate camera pose and constructed map in real-time.

In a nutshell, *edgeSLAM* is a real-time mobile semantic visual SLAM system. The elaborate design of *edgeSLAM* lies in three-fold:

- We decouple the whole visual SLAM into fine-grained modules, and re-assign these modules between mobile device and edge server, such that mobile pipeline can be executed in real-time.
- We design a *Parallel Local Tracking and Global Optimization* workflow. The time-consuming and complex optimization procedure is hidden by mobile local tracking and mapping pipeline, meanwhile ensuring the accuracy.
- We adopt a *Semantic Mask Transfer Strategy*. The pixel-level semantic information of a keyframe will be transferred to related non-keyframes, which will dramatically reduce the infer latency and make semantic segmentation task fulfilled on mobile devices.

In the following sections, we will present the details of these strategies.

IV. Real-time Mobile Semantic Visual SLAM

A. Decoupling and Task Re-assignment of Visual SLAM

In ORB-SLAM, pose estimation and tracking in *Tracking* thread, as well as optimization-related function modules in *Local Mapping* and *Loop Closing* threads contribute most of the computation latency. On mobile devices, both pose estimation and mappoints creation require more than 33.3ms,

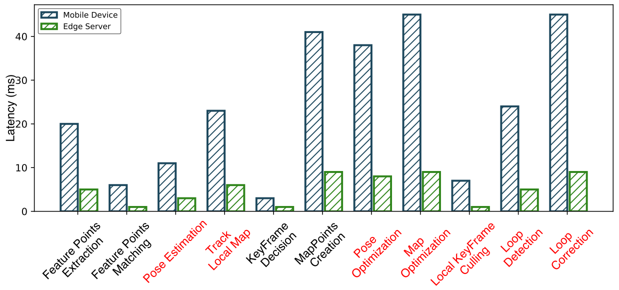


Fig. 7. Operation latency of each function unit in visual SLAM

which makes visual SLAM insuperable to run at >30fps. Moreover, they all rely on MapPoint and KeyFrame databased for global optimization. These are exemplified in Fig. 7, which breaks down the execution latency of the ORB-SLAM task.

We think the decoupling and task re-assignment method should fulfill three principles: First, mobile client can execute tasks assigned to itself in real-time, which means the total latency of these tasks is less than 33.3ms. Second, function modules that will interact with KeyFrame or MapPoint database for further optimization should be executed on edge server. Breaking the data dependency will also lead to high data transmission latency. Last but not the least, same as unmodified visual SLAM, both localization (tracking camera pose) and mapping (creating mappoints) tasks are required to be implemented on mobile devices.

The task assignment strategy in *edgeSLAM* fulfill above principles. As shown in Fig. 6, mobile client will estimate the relative rough camera pose from feature points extraction and matching modules. It will also maintain a local mappoint database, which is only used to construct the local map. In general, the volume of local mappoint database is about 10% of global mappoint database. Meanwhile, edge server will execute the time-consuming and resource-awareness optimization procedure, including optimizing pose and map, and maintaining global mappoint and keyframe databases. When receiving a keyframe, the edge side will send the optimized pose and updated mappoints' information back to the mobile side; and the mobile side will accordingly calibrate the accumulative tracking and mapping error.

B. Parallel Local Tracking and Global Optimization

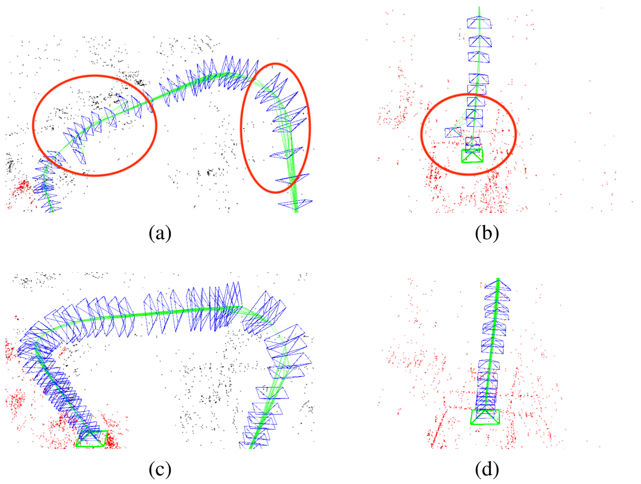


Fig. 8. Illustrations of optimization of local constructed map. A complex local constructed map (a) before optimization on mobile device, in which the camera poses of some frames in the red circle are deviated from the original path, and (c) after optimization, in which the camera poses are correct and smooth. Similarly, (b) and (d) show a simple trajectory on mobile device before and after optimization, respectively.

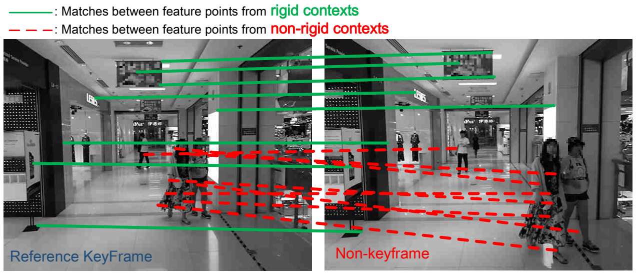

: Matches between feature points from rigid contexts
: Matches between feature points from non-rigid contexts
Reference KeyFrame
Non-keyframe

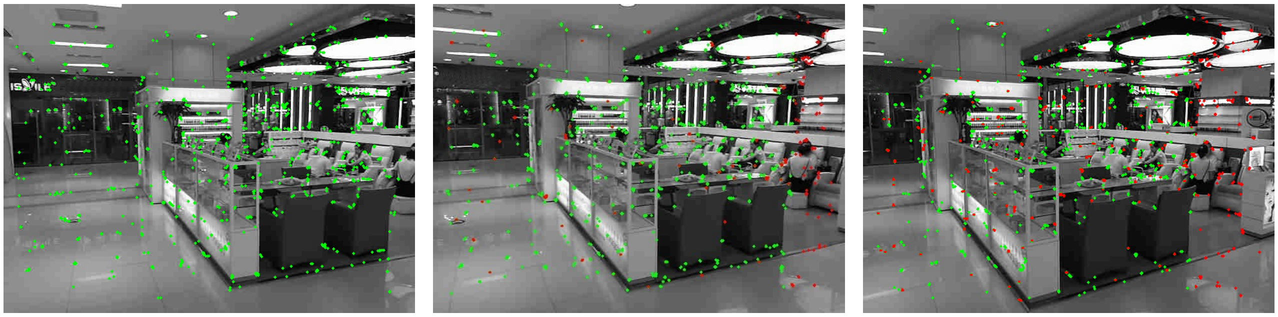
Fig. 9. Illustration of semantic mask transfer strategy

The server performs complete optimization-related modules in parallel with the *Mobile tracking* process on the mobile client. We further design a method to update local constructed map on client with server optimized map, as illustrated in Fig. 8. Transferring the enormous volume of the entire optimized globle map will increase the transmission latency as well as data serialization time dramatically. Thus, we monitor the modification of the MapPoint database updated by the keyframe and generate a map slice, which records the creation, deletion, and calibration of each mappoint by their unique ID as a primary key. Then, we send the map slice rather than entire global optimized map to client for updating. The updating procedure is also executed synchronously with mobile tracking process. By doing so, we can achieve incremental client map optimization, meanwhile decrease the transmission latency and serialization/deserialization time.

C. Semantic Mask Transfer Strategy

Same as many recent works, we adapt Mask R-CNN for NRCC. Mask R-CNN is a famous framework for instance segmentation. It aims to separate different instances in an image via a segmentation mask for each instance. However, despite high accuracy, it's time-consuming and resource-awareness. As shown in Fig. 5a, it results in more than 150ms latency to infer a video frame, which is the bottle-neck of semantic visual SLAM.

In *edgeSLAM*, we design a semantic mask transfer strategy, which can obtain the segmentation information on mobile in real-time. More specifically, as illustrated in Fig.9, only keyframe selected by mobile pipeline will be uploaded to the edge cloud and calculated semantic mask. As for non-keyframe, the pixel level semantic information will be transferred from the latest keyframe according to feature points matching. The rationale behind the transfer strategy is: First, feature points matched between two consecutive frames enjoy larger probability of belonging to the same object categories according to the matching algorithm (DBoW2) in ORB-SLAM. Second, a user (or robot) will move less than 4cm during the 33.3ms (30fps) time-window, assuming that a typical walking speed is 1m/s. Therefore the potential scene changes



(a) Frame# 360

(b) Frame# 375

(c) Frame# 390

Fig. 10. Illustration of keyframe selection strategy. (a) A reference Keyframe. (b) A non-keyframe, more than 90% of the extracted feature points are matched with (a) previous Keyframe. (c) A new keyframe, whose associating rate with (a) is less than 80%.

TABLE I
DIFFERENT CONFIGURATION TUPLES IN SELF-ADAPTATION STRATEGY

Configuration#	f_{min}	f_{max}	α	β	n_l	n_m	N (ms)
1	10	40	0.3	0.9	8	0.8	(0, 10]
2	20	50	0.3	0.9	6	0.8	(10, 15]
3	20	60	0.4	0.9	6	0.8	(15, 35]
4	30	70	0.5	0.8	4	0.7	(35, 50]
5	40	80	0.5	0.7	4	0.6	(50, 300)

in two consecutive frames will not be exceedingly significant. Only inferring keyframes whose scene have changed a lot compared with the previous keyframe, is a more effective way. Semantic information will be updated once the mobile client receives a fresh semantic mask.

V. SELF-ADAPTATION STRATEGY DESIGN

For better adaptation to diverse environments, we considered the mobile pipeline of *edgeSLAM* and found that there are several parameters available for adjustment in the keyframe selection function. By adjusting these parameters dynamically, *edgeSLAM* will achieve relatively optimal results under various computing power restrictions and network conditions.

The basis of keyframe selection is the extraction of feature points, which leverage Gaussian Pyramid to ensure the scale invariance. Different Gaussian Pyramid layers (n_l, nlevels) will lead to diverse ranges of corner point search. With more layers, the computing power consumption will also increase. Thus, the corresponding adjustments is required when the end device has limited computing power.

After extraction of feature points, we design following rules in *edgeSLAM* as principles for keyframe selection:

1) More than f_{min} (mMinFrames, 20 by default) frames have passsed from the last keyframe selection on mobile device, or the number of keyframes in the map is less than f_{min} frames.
2) Less than f_{max} (mMaxFrames, 60 by default) frames have passed from the last optimization for local constructed map.
3) The ratio of extracted feature points in current frame compared with previous keyframe should be at least α (0.4 by default).
4) The ratio of matched feature points in current frame compared with previous keyframe cannot exceed β (0.9 by default).

Condition 1 ensures a good optimization and condition 3 a good tracking. They enhance the quality of selected keyframes. Condition 2 and 4 reduce the redundancy of keyframes and ensure representativeness of them.

Considering the principles above, we can appropriately increase f_{max} and f_{min} or reduce α and β under poor network conditions.

Apart from variables we mentioned above, non maximum suppression parameter n_m in Mask R-CNN framework is also an essential parameter, which makes trade-off between segmentation accuracy and latency. We will also adjust it to adapt different requirements in diverse environments.

Similar to DeepDecision [26], our self-adaptation strategy takes environment measurements (network bandwith B and network latency L) as inputs and defines the network condition N as:

$$N = L + \frac{M}{B} \tag{2}$$

where M is the data size for each frame (1280 × 720 pixels in *edgeSLAM*). Furthermore, *edgeSLAM* outputs the optimization strategy tuple $(f_{\text{min}}, f_{\text{max}}, \alpha, \beta, n_l, n_m)$. Specifically, we set 5 configuration tuples to make *edgeSLAM* adapt to various network conditions, which can be seen in Table I.

VI. EXPERIMENTS AND EVALUATION

A. Implementation

Client. The implementation *edgeSLAM* follows the system workflow in Fig. 6. We implement the client part of *edgeSLAM* on mobile devices (Nvidia Jetson TX2, Apple iPad mini 5, iPhone X and Galaxy S10). To begin with, we continuously capture video frame by camera on devices using OpenCV [27] API and JetPack Camera API and feed it to *mobile tracking and local mapping* thread. `ORBextractor()` function will extract ORB feature points. Meanwhile, CUDA (only on the development board) will accelerate the process [28]. This process is followed by `featurePointsMatching()` function to calculate associations between extracted feature points. Furthermore, `featureCulling()` takes over the procedure, which leverages semantic mask to ensure the rigidity of environment. Keyframe decision function `isKeyFrame()` will be finally performed according to principles described in Section V. It determines whether or not the new created frame is a keyframe. Keyframe will be uploaded to edge server. Meanwhile, `poseEstimation()` and `localMapGeneration()` will estimate the coarse-grained camera pose and construct a map. Once client receives a map slice sent from edge server, `Update()` function will optimize pose and map.

Server. The server's SLAM program is developed on *ORB-SLAM* [29], and we develop the system visualization upon the visualization of *ORB-SLAM*, with some modifications via OpenCV. The server is equipped with Intel(R) Xeon(R) CPU E5-2620v4 of 2.10GHz main frequency and 256G RAM, running the Ubuntu 16.0.4 operating system. For Mask R-CNN, the GPU we use is TITAN V with cuda version 9.1.85 and cudnn-7.05. We apply our Mask R-CNN models with the ResNet-FPN-50 backbone and the network parameters are pre-trained on COCO image Dateset [30]. The Mask R-CNN code is implemented in python-3.6.5 with pytorch-0.4.0 [31].

Remote Data Interaction. The Remote Procedure Call (RPC) refers to the form of inter-process communication (IPC), however different processes have distinct address space. The RPC model helps procedures executed in local invoke remote messages and functions. It is perfectly suitable for Keyframe uploading and optimized map slice updating in *edgeSLAM*. By applying a widely-used RPC framework gRPC [32] developed by Google, we achieve efficient data transmission and remote function invocation. We also leverage the boost serialization library [33], which can be used to reconstitute an equivalent structure in another program context. Moreover, the strategy can also help to encode the structure to binary stream and embed it in Protobuf [34] package as the RPC arguments, thus, saving network bandwidth and accelerating the data transmission rate.

B. Experiment Setup

We have performed an extensive experimental validation of our system in two standard SLAM datasets: TUM [20] and KITTI [21]. 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 a exceedingly challenging dataset for monocular vision due to fast rotations and areas with lot of foliage, which make data association more difficult.

We evaluate the general localization accuracy of *edgeSLAM*, in 16 hand-held indoor sequences of the TUM RGB-Monocular benchmark. Moreover, in 5 sequences from the KITTI dataset, we evaluate the tracking accuracy of camera pose and efficiency of the constructed map optimization. We have carried out all experiments with an Nvidia Jetson TX2 development as mobile device and a desktop computer as edge server. The configuration of them has been shown above. In our experiments, we feed the image from datasets at 30 fps into mobile device. *edgeSLAM* runs in real time and processes the images exactly at the frame rate they acquired.

Furthermore, to extensively evaluate the performance of the edge-assisted design of *edgeSLAM*, we compare *edgeSLAM* with **ORB-SLAM** and **Mask-SLAM**. The former one is the original baseline of our *edgeSLAM* without edge-assisted method and NRCC. The latter is the system used in recent works [19], [35]. They execute NRCC for each video clip and fuse the semantic information with ORB-SLAM. However, neither of these comparative methods can work on mobile devices in real-time. Thus, entire task are offloaded to edge server once mobile device captures a frame.

C. Performance Comparison

1) Localization Accuracy in TUM Dataset: We first examine the localization accuracy of *edgeSLAM*. Fig. 11 depicts the performance of the proposed *edgeSLAM* as well as two other comparative systems in indoor localization scenarios. As shown, the average localization accuracy of *edgeSLAM*, ORB-SLAM, and Mask-SLAM is 0.83cm, 1.24cm and 1.97cm, respectively. *edgeSLAM* outperforms the other two approaches by more than 30%. Moreover, our *edgeSLAM* can work on the mobile device in real-time, which is the unqiue advantage among these approaches. The delightful result comes from the low end-to-end latency, which is ensured in *edgeSLAM* by edge-assisted method and task assignment strategy.

2) Pose Tracking Accuracy in KITTI Dataset: We further examine the camera pose tracking accuracy of *edgeSLAM* in KITTI camera rotation dataset. We calculate the Relative Rotation Error (RRE, an essential evaluation indicator in KITTI dataset) of the tracking result compared with the ground truth. The performance of *edgeSLAM* as well as two comparative methods are depicted in Fig. 12. The tracking accumulative bias of *edgeSLAM* is within 0.22 degree for one meter length traces, which outperforms ORB-SLAM and Mask-SLAM by 5.1% and 17.6%.

3) Mapping Precision in KITTI Dataset: Finally, we evaluate the mapping precision of *edgeSLAM* in KITTI Dataset. We calculate the Relative Translation Error (RTE, another fundamental evaluation indicator in KITTI dataset) of the constructed map compared with ground truth. In this experiment, we compared the global optimized map stored in edge server in *edgeSLAM* with maps constructed by other two systems. As shown in Fig. 13, in all five image sequences, *edgeSLAM* outperforms ORB-SLAM by more than 12% and the performance gap between *edgeSLAM* and Mask-SLAM is within 10%, which demonstrate that *edgeSLAM* achieves competitive performance.

In summary, real-time *edgeSLAM* achieves enhanced localization accuracy and competitive mapping performance with state-of-the-art offline frameworks in all evaluations. The grateful performance comes from not only the efficient design of edge assisted method, but also the adaption of semantic mask transfer strategy.

D. End-to-end Latency

Our system is able to achieve an end-to-end latency within 33.3ms inter-frame time at 30fps to ensure a smooth local-

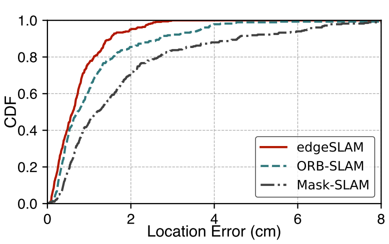


Fig. 11. Localization accuracy on TUM dataset

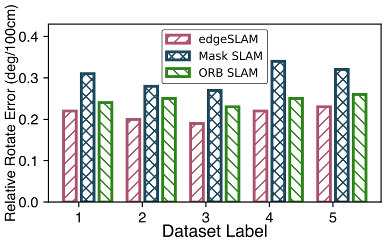


Fig. 12. Tracking accuracy on KITTI dataset

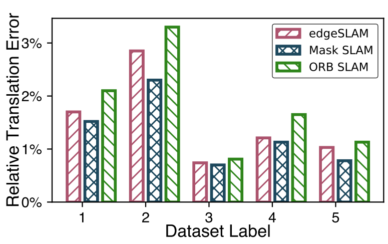


Fig. 13. Mapping accuracy on KITTI dataset

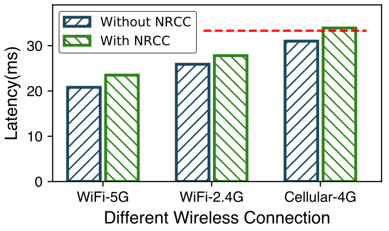


Fig. 14. End-to-end latency comparison

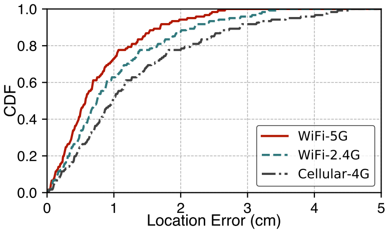


Fig. 15. Localization accuracy comparison

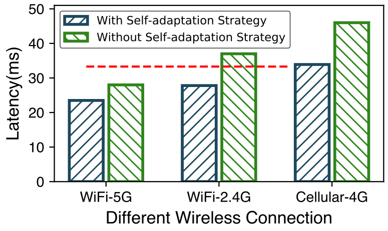


Fig. 16. Impact of self-adaptation on latency

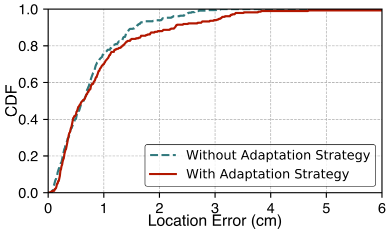


Fig. 17. Impact of self-adaptation on accuracy

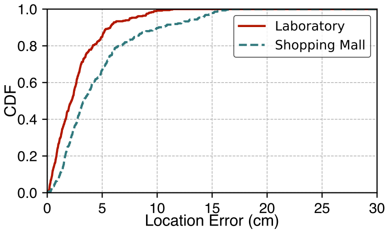


Fig. 18. Pedestrian tracking accuracy

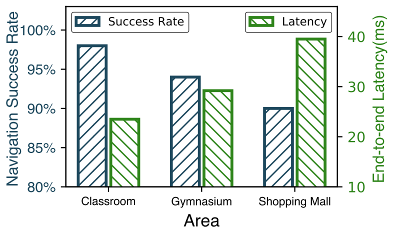


Fig. 19. Robot navigation success rate

ization and mapping experience. To validate this, we run *edgeSLAM* under differnet network connections and calculate the average end-to-end latency for each frame in Fig. 14. The red dashed line in the figure is the 33.3ms deadline for 30fps SLAM devices. We can find that our system is able to finish the entire task within 33.3ms under the majority wireless connection conditions (34.1ms under 4G wireless connection with NRCC). Moreover, the average frame latency increase merely $<$ 4ms if NRCC is involved in *edgeSLAM*, which is the bottle-neck in most systems as aforementioned.

Furthermore, we evaluate the robustness of *edgeSLAM* under different network conditions. As shown in Fig. 15, *edgeSLAM* can reach an average accuracy of 0.7cm, 0.9cm, and 1.3cm under different wireless connections, respectively. The drift of accuracy influenced by network condition is within 0.5cm.

E. Impact of Self Adaptation Strategy

To demonstrate the effectiveness of designed self-adaptation strategy. We evaluate the averge frame latency and system localization accuracy with abd without leveraging the strategy. *edgeSLAM* will upload each frame once mobile client receives the optimization result of previous uploaded frame if without self-adaptation strategy. As shown in Fig. 16, after bringing in the adaptation strategy, the average frame latency decreases more than 5ms, especially >10ms under Celluar-4G wireless connection. The result reflects that the designed self-adaptation strategy plays an important role in ensuring *edgeSLAM* runs in real-time, especially under negative network condition. Fig. 17 shows the performance of *edgeSLAM* with and without adpatation strategy. As seen, the average localization accuracy are 1.03cm and 0.93m, respectively. The precision difference is within 10%. In a nutshell, above results show that the leverage of self-adaptation startegy effectively decreases the end-to-end latency while yields similar performance.

F. Case Study

1) Pedestrian Localization and Tracking: We conducted experiments in a laboratory and 1st floor of a shopping mall. These two areas have different floor layouts, diverse wireless environments, and distinct user behavior patterns. In particular, the crowded shopping mall is the most dynamic. While there are a reasonable number of users in the laboratory most of the time.

Setup. In this case, client side of *edgeSLAM* is implemented on an iOS platform (Apple iPhone X). The server we use is a Lenovo IdeaPad-Y700 with i7-6700HQ CPU of 2.6GHz main frequency and 16G RAM, running the Ubuntu 16.0.4 operating system. For Mask R-CNN, the GPU we used is TITAN V with Cuda version 9.1.85 and cudnn-7.05.

Ground Truth Acquisition. In order to obtain the ground truth, which is the accurate location of pedestrian, we recruited a volunteer to watch surveillance videos, artificially differentiate and track pedestrian and manually mark their locations on the 2D indoor map.

Localization Result Analysis. Fig.18 shows the performance of *edgeSLAM* for pedestrian localization in different areas. As seen, *edgeSLAM* yields an average accuracy of 7.6cm in the laboratory, and 9.9cm in the shopping mall. The corresponding 95th percentile location errors in these two buildings are 9.9cm, 11.4cm, respectively. The result shows that *edgeSLAM* can locate a pedestrian at fine-grained in real-time (40fps in this case study), outperforming state-of-the-art RF-based and vision-based localization systems. Moreover, *edgeSLAM* yields similar performance (accuracy difference $<$ 20%) regardless of the environmental difference because of semantic mask transfer strategy.

2) Robot Mapping and Navigation: We conducted extensive experiments in an office building, a gymnasium and the 1st-3rd floor of a shopping mall, with area sizes of about 400m^2, 1,000m^2 and 4,000m^2, respectively. Overall, we design 17 navigation paths, including 4 short paths ($\leq 100m$), 6 medium paths ($100m - 200m$) and 7 long paths ($\geq 200m$), covering all the main pathways of the testing areas.

Setup. In this test scenario, client side of *edgeSLAM* is a robot implemented with an Nvidia Jetson TX2. The edge server has been mentioned above.

Evaluation Metrics. Similar to some existing works like *Travi-Navi* and *Pair-Navi*, we set checkpoints at turns, escalators and some landmarks on each trajectory. In total, we set 274 checkpoints for the 21 navigation paths. *Navigation success rate* is defined as the rate of successful arrival at each checkpoint.

Navigation Performance. The performance of *edgeSLAM* is depicted in Fig. 19. The average navigation success rates (with NRCC) by *edgeSLAM* in classroom building, gymnasium and shopping mall are 97%, 94% and 90%, respectively.

Additionally, the wireless connections are different in three areas. In classroom building, client are connected to edge server under WiFi-5G link, while in gymnasium and shopping mall are WiFi-2.4G and Cellular-4G, respectively. The average end-to-end latency for each frame is also shown in Fig. 19. The result demonstrates that in classroom and gymnasium, *edgeSLAM* can run in real-time ($>$ 30fps) meanwhile about 25fps in shopping mall. The rationale behind that is the network suffering severe fluctuation in crowded shopping mall.

VII. Related Work

Visual SLAM. Simultaneous Localization And Mapping (SLAM) consists of the concurrent construction of a model and the estimation of the state of the robot moving within it. SLAM has made astonishing progress over the last 30 years, which enables large-scale real-world applications. The pioneering work of monocular visual SLAM [36] adopted a filtering-based approach. Later, optimization-based methods [37] [38] came on stage and were turned out to be more accurate [39]. In recent years, *ORB-SLAM* [23], the state-of-the-art monocular visual SLAM work, used *DBoW2* [40] as the place recognition module, and *g2o* [41] as the optimization framework. Latest researches [42], [43] attempted to incorporate semantics into visual SLAM for better robustness in time-varying environments. Being able to compute the camera pose while generating the map and environment, visual SLAM is a suitable technique for robotic intelligence-based applications.

Semantic Segmentation. In recent years, Convolutional Neural Network (CNN) has been proven to achieve better performance than traditional hand-crafted feature approaches on various detection tasks. Faster R-CNN [44], R-FCN [45], YOLO [46] and SSD [47], *etc.*, are mature systems to infer object detection and classification in image or video. Furthermore, algorithms have been proposed to achieve segmentation on semantic and instance level . The prior work [48] task uses R-CNN [49] to classify region proposals, which were then refined by category-specific coarse mask predictions. FCIS [50] performs object segmentation and detection subtasks jointly and exploits the strong correlation between the two sub-tasks with shared score maps. Mask R-CNN [18], which is the state-of-the-art work for semantic segmentation, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

Edge Assisted strategy. Partially offloading computation-intensive tasks to cloud or edge cloud infrastructures is a feasible way to enable continuous vision analytics or vision based applications like VR/MR. Chen *et al.* [51] evaluate the performance of seven edge computing applications in terms of latency. DeepDecision [26] designs a framework to decide whether to offload the object detection task to the edge cloud or do local inference based on the network conditions. VideoStrom [52] and Chameleon [53] achieve higher accuracy video analytics with the same amount of computational resources on the cloud by adapting the video configurations. Liu *et al.* [24] design an edge assisted system which achieves high accuracy object detection on existing AR/MR system running at 60fps for both the object detection and human keypoint detection task. Most of these works leverage the edge assisted strategy to only focus on the relative facile object detection task. While we are riveted to semantic visual SLAM, either segmentation or localization task is insuperable on mobile devices and demonstrated more complicated than detection task.

VIII. Conclusion

In this work, we propose a *edgeSLAM*, a semantic visual SLAM system for mobile devices, achieving both accuracy and real-time simultaneously with the help of edge computation resource. The core technology of our design lies in: 1, the decomposition of computation modules of SLAM and semantic segmentation; 2, avoidance of redundent computation by reuse imtermediate results; 3, adaption of system parameters to various conditions of network bandwidth and latency. We fully implement *edgeSLAM* and extensively evaluate the performance on three datasets, under different network conditions. The results show that *edgeSLAM* achieves delightful results in all scenarios. Being truly real-time, *edgeSLAM* sheds lights on practical localization and mapping for mobile users and robots.

Acknowledgment

We sincerely thank the anonymous reviewers for their helpful comments and advices. This work is supported in part by the NSFC under grant 61832010, 61632008, 61672319, 61872081, 61632013, 61672240, Microsoft Research Asia.

References

[1] C. Wu, J. Xu, Z. Yang, N. D. Lane, and Z. Yin, "Gain without pain: Accurate wifi-based localization with fingerprint spatial gradient," in *PACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Sep 11-15 2017.

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

[3] Z. Yang, C. Wu, and Y. Liu, "Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention," in *Proceedings of the ACM MobiCom*, 2012.

[4] 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, Feb 2015.

[5] 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.

[6] J. Xu, Z. Yang, H. Chen, Y. Liu, X. Zhou, J. Li, and N. Lane, "Embracing spatial awareness for reliable wifi-based indoor location systems," in *Proceedings of the IEEE MASS*, 2018.

[7] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proceedings of the ACM MobiCom*, 2014.

[8] 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, 2016.

[9] C. Wu, Z. Yang, and C. Xiao, "Automatic radio map adaptation for indoor localization using smartphones," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 517–528, March 2018.

[10] L. Li, P. Xie, and J. Wang, "Rainbowlight: Towards low cost ambient light positioning with mobile phones," in *Proceedings of the ACM Mobicom*, 2018.

[11] H. Abdelnasser, R. Mohamed, A. Elgohary, M. F. Alzantot, H. Wang, S. Sen, R. R. Choudhury, and M. Youssef, "Semanticslam: Using environment landmarks for unsupervised indoor localization," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1770–1782, July 2016.

[12] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[14] X. Zhang, Z. Yang, Y. Liu, and S. Tang, "On reliable task assignment for spatial crowdsourcing," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 174–186, 2016.

[15] L. Cheng and J. Wang, "Vitrack: Efficient tracking on the edge for commodity video surveillance systems," in *Proceedings of the IEEE INFOCOM*, 2018.

[16] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[17] I. C. S. L. S. Committee *et al.*, "Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11^*, 2007.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE ICCV*, 2017.

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

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proceedings of the IEEE IROS*, 2012.

[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE CVPR*, 2012.

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

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

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

[25] "Object keypoint similarity (oks)," 2017, http://cocodataset.org/#keypoints-eval.

[26] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proceedings of the IEEE INFOCOM*, 2018.

[27] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[28] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[29] "Orb-slam2," 2016, https://github.com/raulmur/ORB_SLAM2.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014.

[31] "Pytorch mask r-cnn," 2017, https://github.com/multimodallearning/pytorch-mask-rcnn.

[32] "Grpc," 2017, https://grpc.io/.

[33] "Boost serialization," 2015, https://www.boost.org/doc/libs/1_70_0/libs/serialization/doc/index.html.

[34] "Protocol buffers," 2015, https://developers.google.com/protocol-buffers/.

[35] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," in *Proceedings of the IEEE ICRA*, 2018.

[36] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the IEEE ICCV*, 2003.

[37] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proceedings of the IEEE CVPR*, 2006.

[38] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the IEEE ISMAR*, 2007.

[39] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[40] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[41] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *Proceedings of the IEEE ICRA*, 2011.

[42] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *Proceedings of the IEEE ICRA*, 2017.

[43] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. Montiel, "Towards semantic slam using a monocular camera," in *Proceedings of the IEEE IROS*, 2011.

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[45] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE CVPR*, 2016.

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016.

[48] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *ECCV*. Springer, 2014.

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE CVPR*, 2014.

[50] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proceedings of the IEEE CVPR*, 2017.

[51] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proceedings of the ACM/IEEE Symposium on Edge Computing*, 2017.

[52] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proceedings of the {USENIX} NSDI*, 2017.

[53] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the ACM SIGCOMM*, 2018.