IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2023, accepted 26 June 2023, date of publication 28 June 2023, date of current version 12 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3290901

RESEARCH ARTICLE

Comparison of Analyze-Then-Compress Methods in Edge-Assisted Visual SLAM

JOHANNES HOFER[1], (Graduate Student Member, IEEE), PETER SOSSALLA[1,2], (Graduate Student Member, IEEE), CHRISTIAN L. VIELHAUS [1], JUSTUS RISCHKE[1], MARTIN REISSLEIN[3], (Fellow, IEEE), AND FRANK H. P. FITZEK[1,4], (Senior Member, IEEE)

[1]Deutsche Telekom Chair, 5G Laboratory Germany, Technische Universität Dresden, 01062 Dresden, Germany
[2]Audi AG, 85057 Ingolstadt, Germany
[3]School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA
[4]Centre for Tactile Internet with Human-in-the-Loop (CeTI), Technische Universität Dresden, 01062 Dresden, Germany

Corresponding author: Martin Reisslein (reisslein@asu.edu)

This work was supported in part by the Federal Ministry of Education and Research of Germany in the Program of "Souverän. Digital. Vernetzt." Joint Project 6G-life, project identification number: 16KISK001K, in part by the Federal Ministry of Education and Research of Germany in the Program Software Campus R-SLAM under Grant 01IS17044, and in part by the German Research Foundation (DFG) as part of Germany's Excellence Strategy EXC 2050/1 – Project ID390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technical University of Dresden.

ABSTRACT In edge-assisted visual Simultaneous Localization and Mapping (SLAM), mobile devices offload computationally intensive tasks to an edge cloud. The mobile device should transfer its input data to the edge cloud in a resource-efficient way, especially in visual SLAM systems with their very large image frame input data. Most state-of-the-art systems use a conventional Analyze-Then-Compress (ATC) approach that pre-analyzes the captured frames on the mobile device (incurring substantial processing latencies), and transmits only the (several times smaller) pre-analysis results, namely only the feature representation of so-called key frames (and the corresponding tracking results), to the edge. We examine two novel transmission methods (ATC workflows) for "functional-split" edge-assisted visual SLAM: Feature-Representation-Every-Frame (FREF) transmits the feature representation for every captured frame to the edge, and the edge performs the key frame creation processing (including the tracking); Feature-Representation-Only-Key Frames (FROKF) transmits only the feature representation of key frames without the tracking results to the edge, and the edge performs the tracking. We evaluate these ATC methods in terms of the required network throughput (bandwidth), as well as the latencies for transmission and processing, as well as the resulting end-to-end latency (from frame capture at the mobile device to key frame and tracking results becoming available at the edge cloud) via testbed measurements for various computing hardware platforms. Compared to the existing conventional approach, our newly introduced FREF method can reduce the end-to-end latency down to one quarter; while our FROKF method can reduce the required throughput down to one-fifth. Additionally, feature compression can reduce the required throughput; within the FROKF method, down to one twentieth compared to the conventional method without feature compression, albeit at the expense of significantly increased processing latency for the compression.

INDEX TERMS Edge computing, feature representation, key frame, low latency communication, network throughput, visual simultaneous localization and mapping (vSLAM).

I. INTRODUCTION

Simultaneous Location and Mapping (SLAM) uses sensor data, such as Light Detection and Ranging (LiDAR) [1], radio signals [2], or images [3], [4], [5], [6], to generate a map of the environment and determine the current position of a device. SLAM is often used in mobile devices that require spatial sensing capabilities. For Automated Guided Vehicles (AGV), SLAM is attractive for local navigation due to low installation complexity and installation cost, while achieving

The associate editor coordinating the review of this manuscript and approving it for publication was P. Venkata Krishna.

high levels of flexibility, accuracy, and reliability [7]. SLAM systems are emerging in Industry 4.0 to make production processes cheaper and safer with the help of Mobile Industrial Robots (MIR) [8]. Visual SLAM systems, which utilize optical camera data, are attractive due to their sensor configuration simplicity, miniaturized size, and low cost [9], [10], [11], [12], [13], [14], [15], [16]. The disadvantage is that they typically require extensive computing resources.

A. MOTIVATION FOR EDGE-ASSISTED VISUAL SLAM

To save weight, reduce energy consumption, and keep the form factor of mobile devices small, recent research has explored the offloading of the computationally intensive visual SLAM processing tasks to an edge cloud [17], [18], [19]. However, due to bandwidth limitations and realtime requirements, it is not preferable to offload the entire visual SLAM process. Instead, only certain modules are offloaded in a ''functional-split'' manner, while the basic functionality remains on the mobile device. The offloaded tasks typically include optimizing the map in local and global bundle adjustments and finding and removing unnecessary information from the map.

In order for certain tasks to be offloaded, the mobile device has to send its sensor data to the edge cloud. The type of sensor data plays a decisive role. LiDAR data is usually significantly smaller than conventional optical camera data. For this reason, the LiDAR data can be transmitted raw as in RecSLAM [20]. The transmission of raw optical camera data requires a lot of bandwidth. Thus, edge-assisted visual SLAM systems usually compress or analyze the data on the mobile device. Two approaches can be distinguished. In Compress-Then-Analyze (CTA) architectures [21], [22], the image or video data is reduced in size using well-known compression methods, such as Joint Photographic Experts Group (JPEG) or H.264. The compressed data can be efficiently transmitted to the edge cloud in order to subsequently perform the analysis. In Analyze-Then-Compress (ATC) architectures, the image data is pre-analyzed on the mobile device by extracting features and compressing the features into feature descriptors.

B. ORIGINAL CONTRIBUTION AND STRUCTURE

This study quantitatively examines possible ATC workflows for edge-assisted visual SLAM architectures. The main focus is on the respective throughputs and end-to-end latencies for the transmission of the pre-analyzed input data. After proving general background on visual SLAM in Section II and reviewing related work on edge-assisted visual SLAM in Section III, the core of this article makes the following main contributions:

- Section IV specifies the conventional ATC workflow of transmission of the pre-analyzed input data (key frames with tracking results) from the mobile device to the edge as a comparison benchmark. Also, Section IV introduces two novel functional-split ATC workflows: Feature-Representation-Every-Frame (FREF) which transmits the feature representation for every captured frame from the mobile device to the edge cloud; and Feature-Representation-Only-Key Frames (FROKF) which transmits the feature representation only for key frames from mobile device to the edge cloud. Neither FREF nor FROKF transmits the tracking results from the mobile device to the edge; instead, the edge computes the tracking.
- Section V quantitatively compares the newly introduced FREF and FROKF methods with the conventional transmission method in terms of throughput, processing latency, and resulting end-to-end latency from the capture time instant of a frame on the mobile device to the time instant when the key frame (with tracking results) becomes available in the edge. Section V includes experimental testbed evaluations for four different hardware platforms and three different visual SLAM test sequences. Also, the influence of additional feature compression methods is examined.

II. BACKGROUND

A. EDGE/CLOUD-ASSISTED VISUAL SLAM SYSTEMS

Offloading visual SLAM tasks reduces the computations that the mobile device has to perform [23], [24], [25], reducing energy consumption and eliminating the need for powerful computing hardware on the mobile device. Also, offloading visual SLAM allows for the centralized storage and management of map information on a centralized server. This can be useful in scenarios where multiple mobile devices are operating in the same environment and need to share map information. However, it is important to keep in mind that realtime localization of the mobile device is a mandatory requirement for many use cases; otherwise, security risks may arise.

Offloading the entire visual SLAM system to a server results in a reduction in realtime localization performance, as the round-trip communication time between the mobile device and the cloud or edge cloud can introduce significant latency. For this reason, most state-of-the-art edge-assisted visual SLAM systems perform localization locally on the mobile device. The map generation, maintenance, and optimization have no strict realtime requirements (rather, these tasks enhance accuracy and reliability) and can therefore be offloaded.

Depending on the employed offloading strategy, there are different requirements for the network. When all SLAM tasks are offloaded, it is extremely important that the network latencies are as low as possible. This can be ensured by using the edge cloud, where cloud computing resources are placed at the edge of the network [26], [27], [28], [29], [30]. However, even for partially offloaded SLAM systems, high latencies reduce the reliability [31]. Hence, a nearby edge cloud (with short round-trip communication latency) is generally preferable to conventional cloud computing for offloading visual SLAM.

B. FEATURE DETECTORS AND DESCRIPTORS

Direct visual SLAM systems utilize the information contained in each pixel of the input image [32]. Indirect feature-based visual SLAM systems extract features from the collected input image data. These features are usually points or regions in the image that are unique and easy to identify, such as corners. The mobile device can then use these features to determine its position and orientation relative to the map. Many different feature detection algorithms exist, e.g., Oriented FAST and rotated BRIEF (ORB) [33], FAST [34], SIFT [35], and SURF [36], which differ in their computational complexity and robustness.

To be able to determine the current pose from the features found, one must compare the features with each other. For this purpose, each feature receives a descriptor. A feature descriptor, e.g., BRIEF [37], is a mathematical representation of the appearance of a local image region around a key point (e.g., a corner or edge). The feature descriptor is used to compare and match features between different images or image regions based on the similarity of their descriptor vectors. The key point itself consists of spatial information describing the position in the image, the orientation, and a scale factor, whereby the *OpenCV2* key point class has a total size of 224 bits.

Depending on the SLAM system, certain properties can be omitted or quantized to reduce the amount of data. The size of the feature descriptor depends on the particular feature descriptor implementation. Additionally, different quality levels can be used. In the case of BRIEF, the descriptor consists of a binary vector with a size ranging from 128 bits to 512 bits. The more bits are used per descriptor, the better they can be distinguished from each other. In ORB SLAM2 [38], 256 bits were chosen to keep the memory footprint small while achieving good comparability. Using all unquantized properties of a key point and the 256 bit BRIEF descriptor, a total of 480 bits is required per feature. For instance, for 1000 features per frame and 30 frames per second (FPS) input camera data, the transfer of features would result in a data rate of 14.4 Mbps. The transmission of the feature representation requires roughly only one-fifth of the data rate of the uncompressed transmission of 640×480 pixel grayscale images.

C. FEATURE COMPRESSION

As explained in Section II-B, the data size of an ORB feature results from its binary feature descriptor vector and the corresponding key point. This data size can be reduced by additional compression [39], which generally employs either intra-descriptor coding or inter-frame coding. Intra-descriptor coding exploits the statistical dependencies of the descriptor elements of a single feature. On the other hand, in inter-frame coding, feature descriptors are encoded using a reference feature set, which is usually obtained from the features of the previous frame. In [40], a binary feature intra-descriptor coding scheme was presented, which exploits the dependency between the feature descriptor and the corresponding visual word. In visual SLAM systems, the concept of visual words is used to simplify computationally intensive comparison processes. The concept of visual words is often used in relocalization and loop-closing tasks, where incoming images have to be compared with the complete map. Usually, a bag of words (BoW) is assigned to each image. The BoW contains all visual words found in an image, which are derived by comparing the individual feature descriptors with the visual vocabulary.

The intra-descriptor coding scheme [40] (which is used in [41], [42], whereby we use the [41] implementation for our feature-compression measurements in Section V-E) only sends the index of the found visual word, plus an entropy encoded residual vector per feature. In [41], an edge-assisted visual SLAM system is presented, in which intra-descriptor or inter-frame coding can be applied individually per feature. For inter-frame coding, the number of reference frames can be set. For features that do not change much between frames, the inter-frame skipping mode can be used. In this case, only the ID of the feature that has already been transferred is transmitted. The key point coding scheme was replaced with a lossless scale pyramid-based coding scheme.

In general, the advantage of reducing the amount of data to be transmitted by encoding comes at the cost of an increased computational load and a loss of information. Especially the increased computational load on the mobile device in edge-assisted visual SLAM systems can lead to significant processing latencies.

III. RELATED WORK

A. OVERVIEW

This section reviews the related work on edge-assisted visual SLAM, as summarized in Table 1. We note that to the best of our knowledge, all existing studies on edge-assisted visual SLAM are orthogonal to our study. More specifically, we are the first to examine the fundamental workflow structure (sequencing) of the creation of the key frames and the tracking on the mobile device vs. on the edge in function-split edge-assisted visual SLAM systems that split the visual SLAM functions between the mobile device and the edge cloud (but do not offload the entire visual SLAM processing to the edge cloud).

Generally, there are several ways to resolve the conflict between the low computing power of the mobile device and the high computing complexity of visual SLAM systems. One possibility is to reduce the computational complexity of the SLAM system without losing accuracy and reliability. Visual SLAM systems are very demanding, mainly due to the large number of map points that are continuously inserted into the map during the exploration phase. The longer the runtime and the larger the environment to be explored, the more map points are included in the generated map, and the more demanding tasks (e.g., local and global bundle adjustments) become. One approach to address this challenge

TABLE 1. Overview of data transmission types in existing edge-assisted visual SLAM studies: While early studies considered Compress-Then-Analyze (CTA) methods, most recent studies have employed Analyze-Then-Compress (ATC) methods, which are the focus of this study. Our examined ATC methods accommodate both uncompressed features and compressed features.

	CTA		ATC	
SLAM System	Uncompr. Image	Compr. Image	Uncompr. Feature	Compr. Feature
C^{2}TAM [43]	✓			
Cloud-Based Cooperative 3D Mapping [44]		✓		
Edge-SLAM [45], [46]			✓	
AdaptSLAM [47]			✓	
edgeSLAM [48], [49]			✓	
SwarmMap [50]			✓	
Collab. V-SLAM using Compressed Feature Exchange [42]				✓
Edge Cloud-based Collab. Visual SLAM [51]		✓	✓	✓

is to use artificial markers, i.e., unique identifiers that have to be placed in the environment, as done in the sSLAM approach [52].

B. EDGE/CLOUD-ASSISTED VISUAL SLAM

1) COMPRESS-THEN-ANALYZE (CTA)

An alternative approach to reduce the computing power that is required on the mobile device are edge- or cloud-assisted visual SLAM systems (see Section II-A), which offload computationally intensive tasks to the edge or cloud. Early versions of edge-assisted visual SLAM systems used CTA methods to transfer data from the mobile device to the edge cloud. A cloud-based collaborative 3D mapping architecture, where mobile devices perform a dense visual odometry algorithm for pose calculation, has been implemented in [44]. Key frames are sent as Portable Network Graphic (PNG) compressed red, green, blue, and depth (RGB-D) pictures to the cloud, which performs a key frame pose optimization and merging process. The optimized key frame poses are then sent back to the mobile device.

C^{2}TAM [43] implemented a mapping in the cloud service architecture where tracking still takes place at the mobile device itself. Only individual key frames are sent from the mobile device to the cloud. Those key frames are sent as uncompressed 640×480 pixel red, green, and blue (RGB) images, resulting in an average required throughput of 1 MBytes/s, whereby the rate of sent key frames was not explicitly examined. Sending uncompressed image data consumes (requires) very high throughputs. In [53] it was shown that lossy image compression at low bitrates has strong negative effects on subsequent feature detectors and their feature quality. In addition, feature-preserving image compression methods are presented in [53].

The study [51] investigated the influence of the type of data transmission and the place of execution of the tracking module in edge-assisted SLAM systems. H.264 video compression was used as CTA method. The experimental investigations showed that the video compression method is advantageous for mobile devices with low computational power, since the computationally intensive extraction of features in ATC methods leads to large processing latencies. However, by using lossy video compression, the highest accuracy cannot be achieved.

The CTA method has also the disadvantage that the mobile device has no analyzed input camera data as a result of the transmission of the input data in order to run its own tracking module. In other words, pure CTA transmission of input data to the edge does not produce natively analyzed input data as a "byproduct" which could be directly used on the mobile device in a function-split edge-assisted visual SLAM system. In most state-of-the-art edge-assisted visual SLAM systems, the tracking module is executed on the mobile device so that realtime localization results can be obtained for short time periods independent of the existing infrastructure. Following this function-split approach, which executes the tracking module on the mobile device, in combination with CTA transmission methods, the mobile device has to perform the feature extraction in addition to the image compression in order to be able to perform its own tracking. For this reason, most modern systems use ATC transmission methods. Since the mobile device has to extract features itself anyway, one can use this analyzed data directly for forwarding to the edge cloud.

2) ANALYZE-THEN-COMPRESS (ATC)

In Edge-SLAM [45], [46], the three main modules of ORB SLAM2 (tracking, local mapping, and loop closure) are split between the mobile device and the edge cloud. ORB SLAM2 is a widely used basis for visual SLAM systems and offers very accurate and reliable trajectory estimations [54]. In Edge-SLAM [45], [46], the mobile device runs the tracking module and sends the analyzed feature representations and general tracking results of important frames, i.e., so-called key frames, to the edge cloud and is thus a representative approach of the ATC method. The edge cloud performs the local mapping and loop closure modules. The edge cloud sends the local map updates to the mobile device. Since the mobile device can drive autonomously for short periods of time without any problems, the local map updates do not have to be performed at a high frequency. The local map updates received on the mobile device replace the previous local map. Due to its simplicity, this method is very robust against various errors, such as losing or incorrectly mapping individual map points. However, experiments have demonstrated that due to increased latencies and processing latencies, the system can have low reliability, especially at fast movement speeds and bandwidth constraints. As a result, the realtime execution of test sequences from the KITTI [55] (vehicle on road) and EuRoC MAV [56] (drone in machine hall) benchmarking datasets is not successful on mobile devices with low computational power. By inserting a circular frame buffer on the mobile device, the reliability for increased movement speeds and network latencies was increased in [57].

In AdaptSLAM [47], which is based on ORB SLAM3 [58] and Edge-SLAM [45], [46], the selection of key frames is adapted to the limited data processing and communication resources. More specifically, to reduce the computational load on the mobile device, only a subset of key frames is selected from all possible key frames to create the local map. To reduce the throughput required for sending key frames from the mobile device to the edge cloud, a subset of key frames is also selected. The subset selection is based on local and global map optimization procedures that strive to minimize the map uncertainty while using as few key frames as possible.

Similarly to Edge-SLAM [45], [46], in edgeSLAM [48], [49] and SwarmMap [50], the main modules are split between mobile device and edge, and the analyzed input data is sent from the mobile device to the edge cloud. Additional functionalities, such as semantic segmentation algorithm or collaborative support are added. However, there is a significant difference in the method of information exchange from the edge cloud to the mobile device. Only targeted changes of individual map points are transferred to the mobile device instead of exchanging the local map completely. This has the advantage that the map update process is less prone to failure. On the other hand, the size of the map on the mobile device increases steadily, which makes tracking more and more complex. Furthermore, the structure of the local and global map must be closely synchronized. This results in a strong dependency on infrastructure, as there must be no loss of information. Furthermore, as will be explained in detail in Section IV, it is essential that the mobile device runs the tracking module first so that the global map on the edge cloud can mirror the local map. This leaves less room for flexibility in the amount and type of data to be transferred, as well as the reordering of tasks.

In [41], an edge-assisted visual SLAM system is presented, in which only the analysis of input data is performed on the mobile device. The remaining SLAM tasks are performed on the edge cloud. As a result, localization results or resulting instructions must be sent back to the mobile device. Therefore, the mobile device cannot continue to work autonomously (not even for a short period of time) in the event of a communication interruption. However, the core of the work in [41] deals with the efficient compression of binary feature vectors to minimize the required transmission throughput. In [42], the concept of feature compression is improved by adding support for additional stereo and depth information coding modes. Additionally, a collaborative edge-assisted visual SLAM system is presented, which can better allocate the existing bandwidth among multiple mobile devices due to the reduced throughputs.

IV. ANALYZE-THEN-COMPRESS (ATC) METHODS

A. MOTIVATION

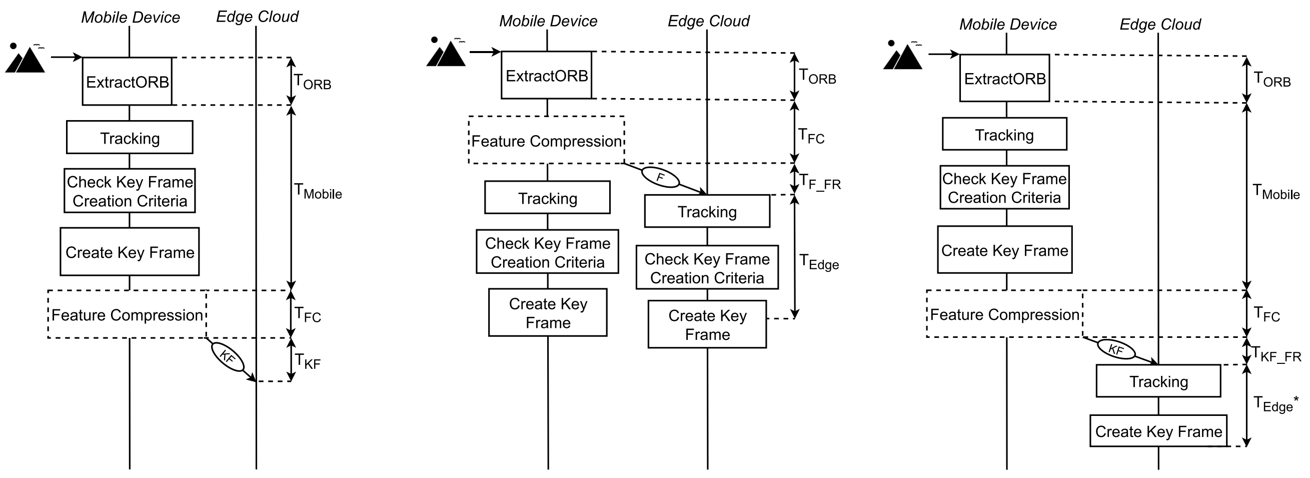
In most state-of-the-art edge-assisted visual SLAM systems, the mobile device itself runs the tracking module to conduct localization in realtime. Therefore, the incoming camera data has to be converted into feature representations for further processing. In terms of transferring the input data from the mobile device to the edge cloud, the feature representation has the advantage of being only a fraction of the size of the original images.

To further reduce throughput, often only key frames are transmitted. Key frames are used in SLAM systems to insert only images with significant levels of new information into the map. This has the advantage of keeping the graph-based map small, which means that the map is kept free of unnecessary information, resulting in reduced computational complexity for all SLAM modules. In order to decide whether an incoming image should be considered a key frame, it must pass through the tracking module, where its features are compared with the map points of the local map and the features of the previous key frame. Subsequently, the key frame creation criteria determine whether the image is accepted as a key frame.

In Edge-SLAM [45], [46] as well as SwarmMap [50], after the decision and the creation of a key frame, the feature representation is sent together with the tracking results to the edge cloud, which subsequently generates a very accurate global map. The tracking results are very large compared to the pure feature representation, which increases the throughput required when sending to the edge cloud. The largest part of the tracking results per key frame consists of the generated map points and their connectivity graph information. At the expense of increased throughput, including tracking results can offer great benefits in some cases. In SwarmMap [50], sending the tracking results has the advantage that the map created on the edge cloud, mirrors the layout of the mobile device map. This allows optimizations of the map on the edge cloud side to be sent as targeted changes to individual map points within the mobile device map. In Edge-SLAM [45], [46], this mirroring of the local map is not necessary, as updates are sent as whole sections that replace the existing map on the mobile device as a whole.

B. SPECIFICATION OF PROPOSED ATC METHODS

Since sending the tracking results as part of the transmitted key frames from the mobile device to the edge cloud is



(a) Conventional (b) FREF (c) FROKF

FIGURE 1. Flowchart comparison of the performed tasks between the instant of capturing an image on the mobile device and presence of the corresponding key frame (and tracking results) on the edge cloud for the conventional ATC workflow and the proposed Feature-Representation-Every-Frame(FREF) and Feature-Representation-Only-Key Frame (FROKF) ATC workflows.

not strictly necessary in most edge-assisted SLAM systems, we propose two alternative approaches to transfer input information from the mobile device to the edge cloud, with the goal of minimizing the required throughput as well as end-to-end latency. The basic idea is to send the pure feature representation of a frame, which in turn requires functionality changes of the edge cloud system. It should be noted that these approaches are not applicable to all edge-assisted visual SLAM systems, as concepts based on mirroring the map structure between mobile device and edge cloud, as in SwarmMap, depend on the transmission of tracking results.

As the basis of our edge-assisted visual SLAM system transmission methods investigation, we use Edge-SLAM [45], [46]. As described in Section IV-A, in the map synchronization process between the edge cloud and the mobile device, entire independent map sections are transferred, which means that there are no dependencies between the global map and the local map. This allows the edge cloud to use its own tracking results to create the global map, rather than relying on the results of the mobile device. This flexibility allows us to use Edge-SLAM [45], [46] to investigate different transmission methods.

1) CONVENTIONAL

Fig. 1(a) shows the conventional workflow from the time an image is captured by the mobile device until the key frame is available on the edge cloud for further processing. The conventional procedure extracts ORB features after capturing the input image. The features are then used to determine the pose of the mobile device within the tracking module. Subsequently, the key frame creation criteria are used to check whether the frame has enough new information to be considered as a key frame. If this is the case, the key frame is generated and sent to the edge. The optional feature compression, explained in Section V-E, should be performed only on key frames in order to reduce unnecessary computation. The key frame contains the compressed or uncompressed feature representation as well as the tracking results. The workflow of the conventional method is used in [45], [46], [47], and [50]; however, the implementation or functionality of individual modules, such as the tracking module, may vary from system to system.

Denoting T_{ORB} for the computation time for extracting the ORB features from the input image, T_{Mobile} for the computation time for the remaining tasks of the tracking module, T_{FC} for the computation time for the optionally applicable feature compression, and T_{KF} for the transmission latency of the key frame (plus tracking results), the end-to-end latency T_{E2E} between the capture of an image on the mobile device and the presence of the corresponding key frame on the edge cloud can be evaluated as:

$$T_{E2E} = T_{ORB} + T_{Mobile} \, (+T_{FC}) + T_{KF}. \tag{1}$$

2) FEATURE-REPRESENTATION-EVERY-FRAME (FREF)

We propose the Feature-Representation-Every-Frame (FREF) method, which sends the feature representation to the edge cloud immediately after the ORB extractor function has finished, or, respectively, after the feature compression has finished. Fig. 1(b) shows the workflow of the proposed FREF transmission method. Since the mobile device does not yet have any information about whether the frame is to be evaluated as a key frame, the feature representation must be sent for each frame. This has the disadvantage of increased throughput.

However, the loss of information between the mobile device and the edge cloud is reduced. Since the edge cloud has increased computing resources at its disposal, the quality of the map can be improved by gaining additional input data. To optimize the quality of the map generated on the edge cloud as well as the accuracy, the key frame creation criteria could be modified, so that more key frames are generated on the edge cloud than on the mobile device. The edge cloud has to execute the tracking module on its own after receiving the feature representation.

The basic concept of the FREF method is the only possible input data transmission method in "full-offload" ATC edge-assisted SLAM systems, in which all SLAM modules are offloaded to the edge cloud, as the mobile device cannot make key frame decisions in these systems. FREF is used, for example, in [42] and [51]. To the best of our knowledge, we are the first to examine FREF for "functional-split" edge-assisted SLAM systems, where a functional split occurs between the mobile device and the edge cloud.

Denoting T_{F_FR} for the transmission latency of sending the feature representation of a frame and T_{Edge} for the computation time for the remaining tasks of the tracking module on the edge cloud, T_{E2E} can be evaluated as:

$$T_{E2E} = T_{ORB} \; (+T_{FC}) + T_{F\_FR} + T_{Edge}. \qquad (2)$$

It can be assumed that due to the increased processing power of the edge cloud, the processing latency of the remaining tracking tasks T_{Edge} is lower than T_{Mobile} [59]. In addition, T_{F_FR} is smaller than T_{KF}, as the tracking results are not transmitted to the edge cloud in FREF. It is important to note that the mobile device still needs to run the tracking module itself to ensure realtime localization.

3) FEATURE-REPRESENTATION-ONLY-KEY FRAME (FROKF)

The second proposed transmission method, shown in Fig. 1(c), is to send the feature representation of a key frame after creating a key frame on the mobile device. This reduces the throughput compared to the conventional method, since the tracking results belonging to the key frame are *not* sent. In addition, the throughput is reduced compared to the FREF method, since only key frames are sent to the edge cloud instead of all frames. Denoting T_{KF_FR} for the transmission latency of sending the feature representation of a key frame (without tracking results) and T_{Edge*} for the computation time for the remaining tasks of the tracking module (without key frame creation criteria check), T_{E2E} can be evaluated as:

$$T_{E2E} = T_{ORB} + T_{Mobile} \; (+T_{FC}) + T_{KF\_FR} + T_{Edge*}. \qquad (3)$$

By not sending the tracking results of the key frames generated on the mobile device, the edge cloud must run the tracking module itself, whereby the edge cloud is already aware that the received feature representation belongs to a key frame.

Since FROKF does not send any tracking results of the key frame that is created on the mobile device to the edge cloud (i.e., the "KF" received by the edge cloud contains only the feature representation), the edge cloud has to perform its own tracking and in order to build the global map the edge cloud has to create its own key frame. This results in a higher end-to-end latency than in FREF. The optional feature compression should be performed only on key frames in the

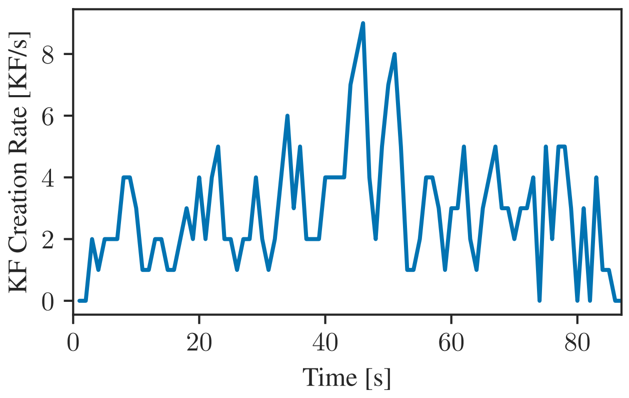


FIGURE 2. Key frame creation rate for *fr3/long_office_household (fr3)* test sequence (`ORBextr.nFeat` = 1000).

Feature-Representation-Only-Key Frame (FROKF) method in order to prevent unnecessary computation. To the best of our knowledge, FROKF is entirely novel and has not been examined for any existing edge-assisted SLAM system.

V. EVALUATION

A. TESTBED

To evaluate the required throughput and end-to-end latency of the proposed FREF and FROKF transmission methods, we employ the following testbed. As edge-assisted visual SLAM system, the Edge-SLAM [45], [46] implementation is used. The conventional method of transmitting the feature representation plus tracking results is implemented as the default benchmark. The study of the feature compression method is based on the implementation of [41].

Since processing latencies depend on the used platform, several platform combinations are tested. We considered the following platforms: a Raspberry Pi 4, a Fitlet2-CE3950, a Fujitsu Esprimo K102-A100 with an Intel Core i7-6700T CPU (2.8 GHz, 8 cores) and 16 GB RAM, and a powerful workstation with an Intel Xeon W-2155 CPU (3.3 GHz, 20 cores) and 128 GB RAM; whereby any platform is considered as the mobile device, while the Fujitsu Esprimo and powerful workstation are considered as the edge cloud. The mobile device and the edge device are connected via Ethernet.

We consider three test sequences: The *fr2/desk (fr2)* [98 seconds] and *fr3* [87 seconds] sequences from the TUM RGB-D benchmarking dataset [60] are handheld indoor recordings with varying translational and rotational speeds. The original sequences contain 30 FPS. For our tests, the sequences were modified to 10 FPS by only playing every third frame, which means that the recordings are still played at unchanged speed. The reason for this is that devices with low processing power were not able to process 30 FPS input data in the visual SLAM system in realtime in our tests. As third sequence, the *kitti_data_odometry_gray_sequence_04 (KI04)* [28 seconds] sequence from the KITTI odometry benchmarking dataset [55] is used. This KITTI sequence is recorded on a standard station wagon driving on a straight public road with high translational and very low rotational speed. The sequence contains 10 FPS.

B. SERIALIZATION INFLUENCE

Serialization is the process of converting a non-serial data structure to a format that can be stored or transmitted, while deserialization is the reverse process of recreating the object from the stored or transmitted serialized format. Serialization is necessary when transmitting data over a network. Serialization incurs a data overhead, which can vary greatly depending on the used serialization method and type. In the open-source implementation of Edge-SLAM [45], [46], the text serialization method from the Boost C++ library is used. The text serialization overhead can be very large, depending on the data structure. A binary serialization method is much more efficient, as we have evaluated as follows.

We have measured the sizes of the serialized feature representation frame objects with both serialization methods for the *fr3* sequence, with 1000 features per frame. Subsequently, we divided the frame size by the number of contained features to obtain the average data size of a feature in a serialized frame. Text serialization gave an average data size of 1610.9 bit; whereas, binary serialization gave 480.7 bit, which is close to the theoretical 480 bit noted in Section II-B. We use binary serialization for all subsequent evaluations in this study.

C. REQUIRED THROUGHPUT

1) PRELIMINARIES

Since the throughput of the analyzed input data transmission in edge-assisted visual SLAM systems depends on more than just the transmission method, we first examine the influence of auxiliary factors. In general, the throughput is jointly governed by the number of features per frame, the data size per feature, and the transmitted frame rate, or, respectively, key frame rate. In the FREF method, the rate of transmitted frames can be freely chosen and in our case is equal to the FPS value of the incoming camera data rate of 10 FPS. By using a lower rate of transmitted frames, there is a possibility that information from possibly important key frames will not be available on the edge cloud, thus decreasing reliability. For the conventional and FROKF methods, the number of transmitted frames depends on the rate at which key frames are generated.

In turn, the key frame creation rate depends on the respective application and can differ greatly depending on the speed and type of movement. The key frame creation criteria can also be customized according to the appropriate environment, hardware, and channel conditions. Fig. 2 shows the key frame (KF) creation rate (KF/s) for the conventional Edge-SLAM system on the Fujitsu mobile device for the *fr3* test sequence. The periodic drops of the KF creating rate are due to the local map update mechanism. Every five seconds, the mobile device receives a map update, which blocks the tracking module. The incoming frames are lost during this

TABLE 2. Average measured throughputs and key frame (KF) creation rates for *fr3* sequence as a function of the `ORBextr.nFeat` parameter (which approx. equals the actual number of features per frame) for the conventional method.

ORBExtractor Features	500	1000	1500	2000
Avg. Throughput [Mbps]	**1.90**	**4.34**	**6.22**	**8.81**
Total # of KF Sent	99	110	117	134
Avg. KF Creation Rate [KF/s]	1.14	1.26	1.35	1.54
Avg. Throughput per KF [MBits/KF]	1.67	3.43	4.63	5.86
Avg. Throughp. per Feat. [Bits/Feat.]	3346	3430	3085	2928

TABLE 3. Required average throughput per frame and KF for the *fr3* sequence: calculated from Eqn. (5) for FREF and FROKF (and validated with testbed measurements which very closely align with the calculated values).

ORBExtractor Features	500	1000	1500	2000
Throughput per (Key)Frame [MBits/KF]	0.26	0.51	0.77	1.03
Avg. Throughput FREF [Mbps]	**2.57**	**5.13**	**7.69**	**10.25**
Avg. Throughput FROKF [Mbps]	**0.29**	**0.65**	**1.03**	**1.58**

time, which means that fewer key frames can be generated. Also, we observe from Fig. 2 that the key frame creation rate is higher around the middle of the test sequence than the beginning or the ending, which is caused by the increased movement speed in the middle of the *fr3* test sequence.

The maximum number of features accepted per frame can be set in Edge-SLAM using the `ORBextractor.nFeatures` parameter, which we abbreviate to `ORBextr.nFeat`. The more features are used per frame, the more complex the calculations become, but the higher the accuracy can be. Due to different environments and hardware, it is not possible to give a universal ideal value for `ORBextr.nFeat`, and we therefore conduct evaluations for a range of `ORBextr.nFeat` values.

2) EVALUATION SETUP

Table 2 shows the measured average throughput between the mobile device and the edge cloud while running the *fr3* test sequence using the conventional Edge-SLAM architecture, where the feature representation plus tracking results are sent from the mobile device to the edge for each key frame. The throughput from the edge cloud to the mobile device (local map update mechanism) is not included, since this study focuses on the transmission of the input data from the mobile device to the edge cloud. The measurements were performed for `ORBextr.nFeat` values from 500–2000. In addition, we evaluate the number of total transmitted key frames and the resulting average key frame creation rates.

In order to provide generally applicable results that are not unique to specific test sequences, we evaluate the throughput to the next smaller entities; specifically, to the throughput per key frame and the throughput per feature. In order to determine the total required throughput for other test sequences, only the respective key frame creation rate has to be measured and extrapolated with the help of the required throughput per key frame given in Table 2. For this reason, we only use the *fr3* test sequence for the throughput evaluations in this study.

3) RESULTS

We observe from Table 2 that the throughput increases consistently with the increase in `ORBextr.nFeat` value and thus the number of features per frame. We note that `ORBextr.nFeat` is the maximum number of transmitted (accepted) features per frame. However, in most images, much more than 2000 features are found (e.g., in fr2, about 6000 features per frame). Thus, for our "low" `ORBextr.nFeat` values of 500–2000, the actual number N of features per frame approximately equals the `ORBextr.nFeat` value. Accordingly, we can conclude from the roughly linear relationship between the `ORBextr.nFeat` value and the throughput that sending twice the number of features roughly doubles the throughput. However, the `ORBextr.nFeat` value also affects the functionality of the SLAM system, which further increases the throughput. The key frame creation rate increases because the key frame creation criteria are unchanged, and thus are more likely to be met if more features are present, which, in turn, increases the throughput. The individual average feature size decreases with increasing features per frame, since the tracking results are distributed over more features; however, this throughput reducing effect of the reduced average feature size is negligible compared to the increased number of features per frame.

Next, we examine the throughputs of the FREF and FROKF methods. Let S denote the serialization overhead, N the number of features per frame or KF, F the feature descriptor size, K the key point size, and D the depth value size. Then, the size M of the data per frame or key frame is clearly determined and can be formulated as:

$$M = S + N \cdot (F + K + D). \qquad (4)$$

The corresponding values for Edge-SLAM [45], [46] are:

$$M_{Edge}\text{SLAM} = 99 + N \cdot (32 + 28 + 4) \text{ in Bytes}. \qquad (5)$$

It should be noted that the serialization overhead S is independent of the number of features N.

The throughput per frame and key frame values are shown in Table 3. The total throughput of the FREF method is calculated by multiplying the FPS (in our case, 10 FPS) by the frame data size. For FROKF, we use the key frame creation rates from Table 2 to calculate the total average throughput to allow a fair comparison with the conventional method. As expected, the throughput required for FROKF is far below the throughput required for the conventional method in Table 2. By omitting the tracking results, the FROKF throughput can be reduced to less than one-fifth of the throughput of the conventional method.

Since every frame is transmitted in the FREF method, the average throughput required for FREF exceeds the throughput requirement of the conventional method (for the key frame creation rates in Table 2). However, with slightly increased key frame creation rates, the FREF method would achieve lower required throughput than the conventional method, since sending additional key frames increases the average throughput. Depending on the `ORBextr.nFeat` value, the average key frame creation rate, from which the average throughput of FREF would be below the conventional method, is between 1.5–1.75 KF/s. Since the key frame creation rate can vary greatly depending on the type of movement of the mobile device and the environment (see Fig. 2), the throughput in the conventional and FROKF methods can be highly dynamic. In contrast, the FREF method has a constant throughput by sending every frame.

D. LATENCY

The end-to-end latency between the mobile device and the edge cloud should be as low as possible. In [31] it was demonstrated that the latency between the mobile device and the edge cloud has a major impact on the reliability of the system. This is mainly due to the local map update mechanism used in Edge-SLAM, where the mobile device deletes its own map when receiving an update. Transmission latencies and processing latencies create a time window that is missing as part of the new local map, but may be highly relevant for the success of further tracking. In [57] it was then demonstrated that by using a key frame buffer on the mobile device, the missing time window in the local map can be shortened so as to improve reliability. Nevertheless, long latencies generally reduce reliability and accuracy. In order to examine the end-to-end latencies [see Eqns. (1)–(3)] of the investigated transmission methods, we proceed to examine the individual latency components in detail.

1) TRANSMISSION LATENCY

The number M of bits to be transmitted per frame or key frame and the bandwidth (bitrate) R [in bits/s] govern the transmission latency $L_T = M/R$. Thus, the average frame and key frame sizes in Tables 2 and 3 and the respective available bandwidths give the transmission latencies in Table 4. In general, the maximum acceptable transmission latency depends on the circumstances. For instance, relatively long latencies can be tolerated if the mobile device moves slowly. On the other hand, fast rotations require short latencies.

We observe from Table 4 that the reduced data size of the frame and key frames in FREF and FROKF substantially reduces the transmission latencies, and this latency reduction is especially relevant at low bandwidths. More specifically, we observe from Table 4 that the FREF and FROKF transmission latencies are consistently roughly one-sixth of the corresponding conventional transmission latency, i.e., the proportional transmission latency reduction is constant. However, from a practical perspective, the absolute transmission latency reductions are particularly pronounced for low bandwidths. For instance, for `ORBextr.nFeat` = 1000, the transmission latency is reduced by over 500 ms for a bandwidth of 5 Mbps, compared to a reduction by less than 30 ms

TABLE 4. Transmission latency [ms] for transmitting the feature representation or the feature representation with tracking results of one frame or key frame as a function of the `ORBextr.nFeat` value and the available bandwidth.

ORB / BW [Mbps]	Feature Representation (FREF/FROKF)				Feature Representation + Tracking Results (Conv.)			
	500	1000	1500	2000	500	1000	1500	2000
5	51.4	102.6	153.8	205.0	334.6	686.0	925.6	1171.2
10	25.7	51.3	76.9	102.5	167.3	343.0	462.8	585.6
20	12.9	25.5	38.5	51.3	83.7	171.5	231.3	292.8
50	5.1	10.3	15.4	20.5	33.5	68.6	92.6	117.1
100	2.6	5.1	7.7	10.3	16.7	34.3	46.8	58.6

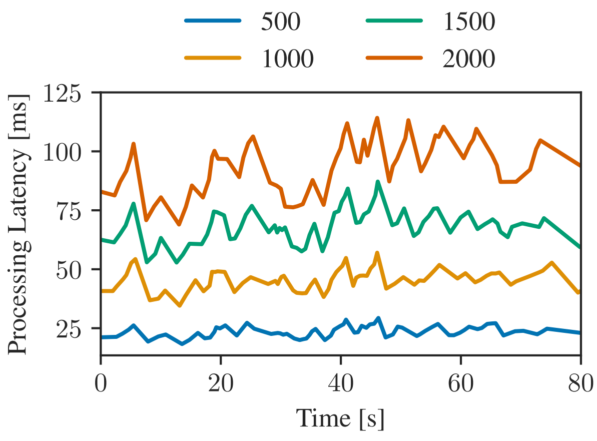


FIGURE 3. Processing latency of the tracking module on the Fitlet mobile device for the test sequence *fr3* for different `ORBextr.nFeat` values.

for a bandwidth of 100 Mbps. A 500 ms latency reduction will typically have a much more profound impact on the reliability than a 30 ms latency reduction. Therefore, the FREF and FROKF methods are advantageous over the conventional method, especially at low bandwidths.

2) PROCESSING LATENCY

Next, we measure the processing latencies for the different transmission methods. Recall that Fig. 1 conceptually illustrates the respective processing latencies of the different transmission methods. The measurements are performed on multiple platforms, since the mobile device and edge cloud can have more or less available computation resources. The Raspberry Pi 4, Fitlet, Fujitsu, and the workstation are considered as the mobile device. In addition, the effect of the `ORBextr.nFeat` value must be taken into account, since it strongly influences the computational complexity. We do not include the processing latency of the *ExtractORB* module in the measurement, since it is the same for all transmission methods. The processing latencies T_{Mobile}, T_{Edge}, and T_{Edge*} in Fig. 1 are all incurred for the same execution of the tracking module as well as the key frame criteria check (requires less than 0.01 ms and is thus negligible) and key frame creation after feature extraction. Therefore, we only measure T_{Mobile} for the conventional method since

TABLE 5. Mean processing latency [ms] of the tracking module [which approx. equals T_{Mobile} in Fig. 1(a), T_{Edge} in Fig. 1(b), and T_{Edge*} in Fig. 1(c)] for *fr2*, *fr3*, and *KI04* test sequences; each evaluated on various mobile device hardware, for different `ORBextr.nFeat` values. The symbol "/" indicates that the run was unsuccessful.

	ORB / Hardware	500	1000	1500	2000
fr2	Workstation	/	23.4	34.6	48.4
	Fujitsu	/	23.2	37.0	63.9
	Fitlet	/	46.0	71.4	99.2
	Raspberry Pi 4	/	155.5	232.4	/
fr3	Workstation	13.4	23.4	36.6	44.6
	Fujitsu	13.4	24.1	38.7	55.9
	Fitlet	23.6	45.6	68.4	93.6
	Raspberry Pi 4	84.0	166.3	227.4	/
KI04	Workstation	/	16.1	21.5	28.2
	Fujitsu	/	16.6	23.9	31.9
	Fitlet	/	31.9	45.6	59.1
	Raspberry Pi 4	/	/	/	/

the differences between T_{Mobile}, T_{Edge}, and T_{Edge*} are negligible. That is, the transmission method does not influence the actual processing latency of the tracking module; rather, the transmission method determines where the tracking module is executed [and thus indirectly influences the processing latency (which is hardware dependent and included in the end-to-end latency T_{E2E})].

Fig. 3 shows the processing latency T_{Mobile} on the Fitlet over the course of the *fr3* test sequence for `ORBextr.nFeat` values between 500–2000. The increasing computational complexity due to more features to be processed can be clearly observed.

Table 5 presents the average measured processing latencies T_{Mobile} for the different combinations of test sequence, platform, and `ORBextr.nFeat` value. We observe from Table 5 that the hardware strongly influences the processing latency. For example, the measured processing latencies on the Raspberry Pi 4 are very long (over three time longer than for the Fitlet). Due to these long Raspberry Pi 4 processing latencies, the test sequences cannot be run successfully for high `ORBextr.nFeat` values. Also, for

TABLE 6. **End-to-end latency for Fitlet mobile device and Fujitsu edge cloud for conventional, FREF, and FROKF transmission as a function of bandwidth and `ORBextr.nFeat` value, for *fr3* sequence.**

	Conventional			FREF			FROKF		
ORB \ BW [Mbps]	10	50	100	10	50	100	10	50	100
500	190.9	57.1	40.3	39.1	18.5	16.0	62.7	42.1	39.6
1000	388.6	114.2	79.9	75.4	34.4	29.2	121.0	78.0	74.8
1500	531.2	161.0	115.2	115.6	54.1	46.4	184.0	122.5	114.8
2000	679.2	210.7	152.2	158.4	76.5	66.2	252.0	170.1	159.8

the demanding *KI04* test sequence, a successful run was not possible with the Raspberry Pi 4 mobile device even with lower `ORBextr.nFeat` values.

Also, for the *fr2* and *KI04* test sequences, no successful test runs could be achieved with an `ORBextr.nFeat` value of 500, regardless of the mobile device platform. The reason for this is probably the nature of the observed environments, which are very feature-rich. By limiting the maximum number of features per frame, important features may be rejected. If previously rejected features are accepted in subsequent frames, then no feature matches can be found, resulting in a lost track.

Overall, we conclude that the Raspberry Pi 4 can only be used for applications with very slow movement speeds and static environments. The Fitlet is a good alternative, since its form factor corresponds to that of a Raspberry Pi 4, but still achieves shorter processing latencies. Nevertheless, the Fitlet processing latencies are in a critical range for high `ORBextr.nFeat` values. For applications with high movement speeds and dynamic environments, hardware similar to the Fujitsu should probably be used as a mobile device to ensure sufficient reliability.

In addition, we observe from Table 5 that the processing latencies of the tracking module are generally substantially shorter for the *KI04* sequence than for the *fr2* and *fr3* sequences. This is due to the fast movement speed in the *KI04* sequence and thus the low number of existing map points in the current environment. This means that fewer features need to be compared with each other, which takes up less computation time.

3) END-TO-END LATENCY

Based on the measured transmission latencies (Table 4) and processing latencies (Table 5), Eqns. (1), (2), and (3) can be used to calculate the end-to-end latencies for the respective transmission methods. Table 6 gives the end-to-end latencies for the Fitlet mobile device and Fujitsu edge cloud. We observe from Table 6 that FREF achieves the lowest end-to-end latencies. The end-to-end latency reductions with FREF become more pronounced for lower bandwidth; for the 10 Mbps bandwidth, the FREF end-to-end latency is less than one-quarter of the conventional end-to-end latency. On the other hand, the higher the available bandwidth, the lower the transmission latencies and thus the smaller the end-to-end

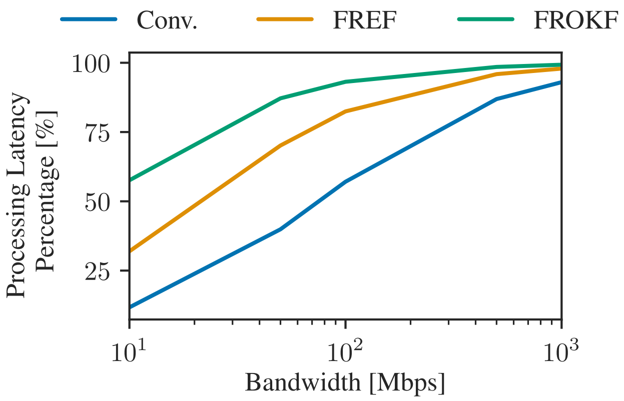


FIGURE 4. **Percentage of processing latency included in end-to-end latency for `ORBextr.nFeat` value of 1000.**

latency reduction due to reduced frame data sizes. For the 100 Mbps bandwidth, the FREF end-to-end latency is a little less than half the conventional end-to-end latency.

Fig. 4 shows the percentage of the processing latency in the total end-to-end latency as a function of the available bandwidth. The higher the percentage, the lower the advantage of reduced transmission latency due to reduced frame data sizes for FREF and FROKF.

FROKF also achieves lower end-to-end latencies than the conventional method at low bandwidths. However, there is a tradeoff, as illustrated in Fig. 5: Since the FROKF processing latencies are higher than the processing latencies of the conventional method, there exists a threshold bandwidth value. Above that threshold bandwidth value, the FROKF end-to-end latency is longer than the end-to-end latency of the conventional method. We observe in Table 6 that for an `ORBextr.nFeat` value of 2000, the FROKF end-to-end delay exceeds the conventional end-to-end delay for a bandwidth of 100 Mbps, i.e., 100 Mbps exceeds the threshold bandwidth value. More specifically, from Fig. 5, or by equating Eqns. (1) and (2), i.e., by equating $T_{KF} = T_{KF\_FR} + T_{Edge*}$ and solving for the transmission bitrate, we find that this threshold bandwidth is 121 Mbps for an `ORBextr.nFeat` value of 1000.

E. FEATURE COMPRESSION

Since the feature representation is transmitted in all examined transmission methods, feature compression has in absolute

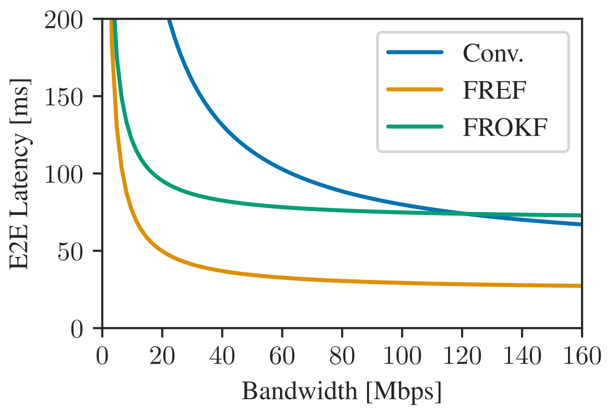


FIGURE 5. End-to-End latency for Conventional, FREF, and FROKF for Fitlet mobile device and Fujitsu edge for `ORBextr.nFeat` value of 1000 as a function of bandwidth.

TABLE 7. Percentages of performed coding (feature compression) modes depending on `ORBextr.nFeat` value measured on Fujitsu.

	Mode \ ORB	500	1000	1500	2000
fr2	Intra	/	29.1	27.8	26.0
	Inter	/	70.6	71.8	73.6
	Inter-Skip	/	0.3	0.3	0.3
fr3	Intra	25.4	23.8	22.0	20.1
	Inter	74.3	75.8	77.7	79.5
	Inter-Skip	0.4	0.4	0.4	0.4
KI04	Intra	/	56.4	55.0	53.6
	Inter	/	43.5	44.8	46.2
	Inter-Skip	/	0.2	0.2	0.2

terms the same impact in all transmission methods. Only the percentages of the resulting throughput reduction and processing latency increase are different. We examine the intra, inter, and inter-skipping coding schemes [41] as possible feature compression methods. First, we evaluate the probabilities for these compression methods by processing the test sequences on the Fujitsu and saving the used coding mode for each feature. Table 7 gives the percentages for every coding mode for different `ORBextr.nFeat` values. In *fr2* and *fr3*, mainly inter-frame coding is applied, since the mobile device moves slowly, and thus many features remain similar in successive frames. In *KI04*, intra-descriptor coding is applied more often due to the fast movement speed. For all test sequences, increasing the `ORBextr.nFeat` value, increases the proportion of inter-frame coding. Inter-skipping coding is almost never applied, because inter-skipping requires that the feature descriptors of two frames are almost identical (which should presumably occur more often during standstill).

1) THROUGHPUT

We evaluate the influence of feature compression on the required throughputs of the transmission methods. To do this, we measure the data sizes of the compressed feature representation of all frames of the *fr3* test sequence. Table 8 gives the average frame sizes and the average feature sizes as a function of the `ORBextr.nFeat`. The higher the `ORBextr.nFeat` value, the smaller the average feature size is likely to be, since more features are compressed by inter-frame coding (see Table 7), which achieves higher compression. As described in Section II-B, an uncompressed feature has the size of 480 bits. Compression reduces the average size of a feature down to less than 150 bits, i.e., down to less than one third of the uncompressed feature size. This significant size reduction is achieved at the cost of quality loss and additional processing latency, which will be considered in Section V-E2.

TABLE 8. Average measured frame and feature sizes depending on `ORBextr.nFeat` parameter for test sequence *fr3*.

ORBExtractor Features	500	1000	1500	2000
Avg. Frame Size [KBits]	74.0	149.7	222.7	285.2
Avg. Feature Size [Bit]	148.0	149.7	148.5	142.6

TABLE 9. Calculated average throughput [Mbps] with feature compression as a function of the `ORBextr.nFeat` value for *fr3* test sequence.

Transm. Method \ ORB	500	1000	1500	2000
Conv	1.74	4.01	5.73	8.14
FREF	0.72	1.52	2.15	2.88
FROKF	0.09	0.19	0.30	0.44

Table 9 lists the average required throughputs for the studied transmission methods when using the feature compression. For the conventional method, the savings per feature were subtracted from the throughput measurements in Table 2. For FREF and FROKF, the average frame sizes were calculated as in Table 3. In absolute terms, all transmission methods reduce the required throughput by the same amount due to the feature compression. Relatively speaking, the throughput of the conventional method is reduced by only 8% on average (compared to Table 2); whereas, in FREF and FROKF, the throughput is reduced by approximately 70% on average (compared to Table 3). Overall, the required throughput of FROKF with feature compression (see Table 9) is only about one-twentieth compared to the throughput of the conventional method without feature compression (see Table 2).

2) PROCESSING LATENCY

The reduced required throughput comes at the cost of increased computation on the mobile device and the edge cloud. For each feature, it must be decided which coding scheme should be used. The respective coding scheme probabilities have already been reported in Table 7. The coding schemes have different complexities, with correspondingly varying processing latencies. Also, the used platform strongly influences the processing latencies.

TABLE 10. Average encoding delay [μs] per feature on various mobile device hardware. The symbol "/" indicates that the run was unsuccessful.

	HW	ORB / Mode	500	1000	1500	2000
Fitlet	fr2	Intra	/	35.2	36.0	35.1
		Inter	/	14.0	14.0	14.1
	fr3	Intra	34.4	34.5	34.6	34.3
		Inter	13.8	14.0	13.9	14.1
	KI04	Intra	/	35.9	33.9	34.3
		Inter	/	14.7	14.3	14.8
Fujitsu	fr2	Intra	/	23.6	25.5	25.8
		Inter	/	11.5	14.7	14.6
	fr3	Intra	24.8	25.6	25.9	26.2
		Inter	11.5	14.3	15.2	14.9
	KI04	Intra	/	27.1	23.1	22.9
		Inter	/	13.2	11.2	11.9

TABLE 11. Average decoding delay [μs] per feature on various edge computing hardware.

	HW	ORB / Mode	500	1000	1500	2000
Fujitsu	fr2	Intra	/	23.1	23.1	21.6
		Inter	/	21.3	24.4	24.4
	fr3	Intra	20.3	23.1	22.8	21.3
		Inter	19.1	22.2	24.5	24.3
	KI04	Intra	/	18.1	15.8	19.0
		Inter	/	19.2	16.8	19.1
Worksta.	fr2	Intra	/	16.4	16.6	16.7
		Inter	/	17.3	17.4	17.0
	fr3	Intra	16.3	16.5	15.7	15.5
		Inter	15.8	17.0	18.1	18.0
	KI04	Intra	/	13.4	14.1	16.7
		Inter	/	15.1	16.1	18.4

To investigate this, we measure the processing latencies of each feature encoding and decoding occurrence over the complete course of the respective test sequences. The workstation and the Fujitsu are used as the edge cloud and thus decoder. The Fitlet and the Fujitsu are used as the mobile device and thus the encoder. The Raspberry Pi 4 could not be used with the existing feature compression C++ implementation due to errors caused by its processor architecture. Since the measured processing latencies and end-to-end latencies in Section V-D are already quite high with the Raspberry Pi 4, the additional coding processing latencies would make the behavior even worse.

Tables 10 and 11 give the measured mean encoding and decoding times, respectively. We report these coding latencies separately for intra coding and for inter coding for a range of `ORBextr.nFeat` values. Inter-skipping is not considered, since the processing latency for decoding and encoding of one feature is less than one microsecond and in combination with the very low probabilities, the impact of inter-skipping is negligible.

The following conclusions can be drawn from Tables 10 and 11: The used platform has a strong impact on the processing latency of the intra-frame encoding, as well as on the intra- and inter-frame decoding. The test sequence has only a minor influence. The `ORBextr.nFeat` value also has no clear influence on the coding processing latencies of the individual features.

The mean encoding and decoding processing latencies per feature, the used ORBextracor.nFeatures value, and the coding scheme probabilities enable the calculation of the average overall (end-to-end) encoding and decoding latencies given in Table 12. In relation to the end-to-end latencies without feature compression given for the Fitlet-Fujitsu combination in Table 6, the overall encoding and decoding processing latencies in Table 12 are relatively large. Specifically, for `ORBextr.nFeat` values 1000–2000, the average feature

TABLE 12. Average aggregate (sum) of the encoding latency and the decoding latency [ms] per frame for different mobile device-edge computing combinations, averaged over all test sequences.

ORB / HW	500	1000	1500	2000
Fitlet-Fujitsu	19.2	41.1	63.6	83.3
Fitlet-Workstation	17.4	35.6	53.9	71.1
Fujitsu-Fujitsu	17.2	39.2	62.3	81.4
Fujitsu-Workstation	15.3	33.8	52.5	69.2

encoding and decoding latencies in Table 12 exceed the FREF end-to-end latencies (without feature compression) for bandwidths of 50 and 100 Mbps in Table 6.

3) END-TO-END LATENCY

The measurements in Section V-E2 indicate that feature compression adds relatively long coding processing latencies to the end-to-end latency. On the other hand, the reduction of the frame or key frame data sizes reduces the transmission latency. Fig. 6 shows the processing, transmission, and feature compression processing latency components for the investigated transmission methods. For comparison, the latency components are also shown without feature compression.

Fig. 6 indicates that the feature compression (coding) processing latencies are substantially longer than the reductions of the transmission latency, i.e., that the feature compression does not pay off for the examined scenario. There is a threshold bandwidth below which the reduction in transmission latency due to the reduction in frame data sizes outweighs the added processing latencies. With an `ORBextr.nFeat` value of 1000 and the average data sizes measured for the *fr3* test sequence, this bandwidth is 6.29 megabits per second (Mbps) for the conventional method and 8.84 Mbps for FREF

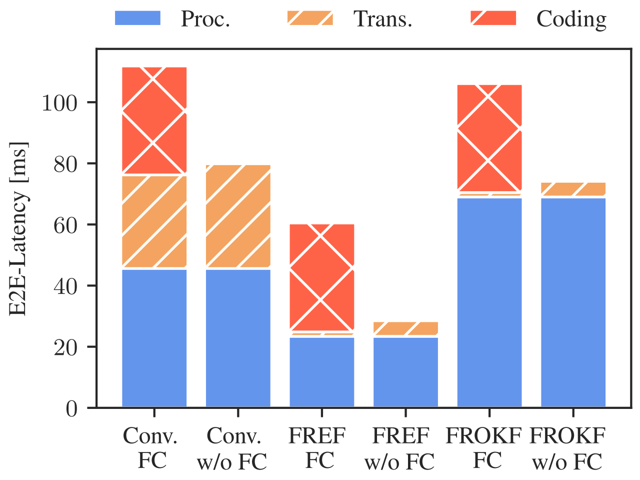


FIGURE 6. End-to-end latencies for conventional, FREF, and FROKF transmission methods with and without feature compression (FC) which incurs the coding latency; Fitlet mobile device, Fujitsu edge cloud, 100 Mbps bandwidth, and 1000 `ORBextr.nFeat` for *fr3* test sequence.

and FROKF. Below these bandwidths, feature compression reduces the end-to-end latency.

VI. CONCLUSION

In edge-assisted visual SLAM systems, there are several options for the data transfer from the mobile device to the edge cloud. These options differ in the amount of data to be transmitted and the resulting end-to-end latency from image capture until the corresponding data is available for further processing on the edge cloud. This study investigated the Analyze-Then-Compress (ATC) transmission methods. Two novel methods, namely FREF and FROKF, were introduced with the goal of reducing end-to-end latency and required throughput. For repeatable and meaningful results, the investigated transmission methods were tested on four different platforms and with three benchmarking test sequences. In addition, consideration was given to various parameters, such as key frame creation rate and `ORBextr.nFeat`, in order to obtain results that are as broadly applicable as possible.

Compared to the conventional transmission method of transmitting the feature representation plus tracking results of every key frame, the FREF method (which transmits the feature representation without tracking results for every frame) reduced the end-to-end latency down to less than one quarter at a bandwidth of 10 Mbps. However, with the FREF method, the mobile device is forced to transmit each input frame to the edge cloud, which increases the required throughput above that of the conventional method. The increased throughput comes with the advantage that the edge cloud receives more information, which can be used to increase accuracy. For use cases with sufficient available bandwidth, the FREF method should provide the most reliable results. The FROKF method (which transmits the feature representation without tracking results only for key frames) reduces the required throughput down to less than one-fifth of the throughput required by the conventional method. Whether the resulting FROKF end-to-end latency is higher or lower than that of the conventional method depends on the available bandwidth.

To further reduce the required throughput in ATC methods, feature compression can be applied. This was also tested experimentally in our testbed to investigate the impact on end-to-end latency and throughput. The measurements indicate that feature compression reduces the required throughput for the FREF and FROKF methods by approximately 70%. The measurements also indicate that the additional encoding and decoding tasks for feature compression incur significant processing latencies that exceed the reduction in transmission latency at moderate to high bandwidths.

Overall, our findings indicate that the choice of an appropriate transmission method for a specific application depends on several factors, including the available bandwidth and the available computation capacity on the mobile device. The choice has an impact on the resulting end-to-end latency as well as the required throughput. By reducing the end-to-end latency between the mobile device and the edge cloud, the reliability of edge-assisted visual SLAM system can generally be substantially improved. This can offer great potential for realtime sensitive application areas, such as autonomous driving, by enabling higher mobile device speeds. Reduced throughput is beneficial in application areas with severely limited bandwidths, allowing, for example, more mobile devices to operate simultaneously in collaborative SLAM systems.

When considering our specific measured values, it is important to understand that they do not universally represent all visual SLAM systems. Using other test sequences or modifying the SLAM system may lead to different measurement values. Therefore, our measured values should be interpreted as a reference to better understand the influence of the different transmission methods and to demonstrate their respective advantages and disadvantages.

In future work, the proposed new transmission methods, which have been examined in this study for a single mobile device, could be investigated in the context of swarm scenarios where multiple mobile devices, e.g., a swarm of mobile robots, collaborate to map and to localize themselves in an environment [61], [62]. It would be particularly interesting to explore heterogeneous scenarios, where the mobile devices have different computing capabilities and different available bandwidth, e.g., due to varied wireless signal propagation conditions. Another direction for future research could be to revisit the CTA methods, e.g., by exploring adaptations of the image compression to the requirements of edge-assisted visual SLAM systems.

REFERENCES

[1] G. Lu, H. Yang, J. Li, Z. Kuang, and R. Yang, "A lightweight real-time 3D LiDAR SLAM for autonomous vehicles in large-scale urban environment," *IEEE Access*, vol. 11, pp. 12594–12606, 2023.

[2] B. Amjad, Q. Z. Ahmed, P. I. Lazaridis, M. Hafeez, F. A. Khan, and Z. D. Zaharis, ''Radio SLAM: A review on radio-based simultaneous localization and mapping,'' *IEEE Access*, vol. 11, pp. 9260–9278, 2023.

[3] M. R. Gkeka, A. Patras, N. Tavoularis, S. Piperakis, E. Hourdakis, P. Trahanias, C. D. Antonopoulos, S. Lalis, and N. Bellas, ''FPGA accelerators for robust visual SLAM on humanoid robots,'' in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2022, p. 51.

[4] A. M. Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, ''A comprehensive survey of visual SLAM algorithms,'' *Robotics*, vol. 11, no. 1, pp. 24:1–24:27, Feb. 2022.

[5] T. Taketomi, H. Uchiyama, and S. Ikeda, ''Visual SLAM algorithms: A survey from 2010 to 2016,'' *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, pp. 1–11, Dec. 2017.

[6] M. Wasala, H. Szolc, and T. Kryjak, ''An efficient real-time FPGA-based ORB feature extraction for an UHD video stream for embedded visual SLAM,'' *Electronics*, vol. 11, no. 14, Jul. 2022, Art. no. 2259.

[7] M. Aizat, A. Azmin, and W. Rahiman, ''A survey on navigation approaches for automated guided vehicle robots in dynamic surrounding,'' *IEEE Access*, vol. 11, pp. 33934–33955, 2023.

[8] X. Karamanos, Y. Karamitsos, D. Bechtsis, and D. Vlachos, ''Mobile industrial robotic vehicles: Navigation with visual SLAM methodologies,'' in *Autonomous Vehicles Perspectives and Applications*. London, U.K.: IntechOpen, 2023.

[9] D. Esparza and G. Flores, ''The STDyn-SLAM: A stereo vision and semantic segmentation approach for VSLAM in dynamic outdoor environments,'' *IEEE Access*, vol. 10, pp. 18201–18209, 2022.

[10] K. Lv, Y. Zhang, Y. Yu, Z. Wang, and J. Min, ''SIIS-SLAM: A vision SLAM based on sequential image instance segmentation,'' *IEEE Access*, vol. 11, pp. 17430–17440, 2023.

[11] S. Mokssit, D. B. Licea, B. Guermah, and M. Ghogho, ''Deep learning techniques for visual SLAM: A survey,'' *IEEE Access*, vol. 11, pp. 20026–20050, 2023.

[12] S. Song, H. Lim, S. Jung, and H. Myung, ''G2P-SLAM: Generalized RGB-D SLAM framework for mobile robots in low-dynamic environments,'' *IEEE Access*, vol. 10, pp. 21370–21383, 2022.

[13] Y. Park and S. Bae, ''Keeping less is more: Point sparsification for visual SLAM,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 7936–7943.

[14] J. Peng, Y. Hou, H. Xu, and T. Li, ''Dynamic visual SLAM and MEC technologies for B5G: A comprehensive review,'' *EURASIP J. Wireless Commun. Netw.*, vol. 2022, no. 1, pp. 1–23, Oct. 2022.

[15] S. Yang, C. Zhao, Z. Wu, Y. Wang, G. Wang, and D. Li, ''Visual SLAM based on semantic segmentation and geometric constraints for dynamic indoor environments,'' *IEEE Access*, vol. 10, pp. 69636–69649, 2022.

[16] S. Zhang, L. Zheng, and W. Tao, ''Survey and evaluation of RGB-D SLAM,'' *IEEE Access*, vol. 9, pp. 21367–21387, 2021.

[17] X. Cui, C. Lu, and J. Wang, ''3D semantic map construction using improved ORB-SLAM2 for mobile robot in edge computing environment,'' *IEEE Access*, vol. 8, pp. 67179–67191, 2020.

[18] T. Chase, A. J. Ben Ali, S. Y. Ko, and K. Dantu, ''PRE-SLAM: Persistence reasoning in edge-assisted visual SLAM,'' in *Proc. IEEE 19th Int. Conf. Mobile Ad Hoc Smart Syst. (MASS)*, Oct. 2022, pp. 458–466.

[19] D. Dechouniotis, D. Spatharakis, and S. Papavassiliou, ''Edge robotics experimentation over next generation IIoT testbeds,'' in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2022, pp. 1–3.

[20] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, ''Edge robotics: Edge-computing-accelerated multi-robot simultaneous localization and mapping,'' 2021, *arXiv:2112.13222*.

[21] D. Van Opdenbosch, ''Data compression for collaborative visual SLAM,'' Ph.D. dissertation, TUM School Comput., Inf. Technol., Tech. Univ. Munich, Germany, 2019.

[22] A. Redondi, L. Baroffio, M. Cesana, and M. Tagliasacchi, ''Compress-then-analyze vs. analyze-then-compress: Two paradigms for image analysis in visual sensor networks,'' in *Proc. IEEE 15th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2013, pp. 278–282.

[23] M. Fukui, Y. Ishiwata, T. Ohkawa, and M. Sugaya, ''IoT edge server ROS node allocation method for multi-SLAM on many-core,'' in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Other Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 421–426.

[24] P. Huang, L. Zeng, K. Luo, J. Guo, Z. Zhou, and X. Chen, ''ColaSLAM: Real-time multi-robot collaborative laser SLAM via edge computing,'' in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2021, pp. 242–247.

[25] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, ''Offloading SLAM for indoor mobile robots with edge-fog-cloud computing,'' in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, May 2019, pp. 1–6.

[26] T. V. Doan, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, ''FAST: Flexible and low-latency state transfer in mobile edge computing,'' *IEEE Access*, vol. 9, pp. 115315–115334, 2021.

[27] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, ''Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping,'' *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[28] H. Jin, M. A. Gregory, and S. Li, ''A review of intelligent computation offloading in multiaccess edge computing,'' *IEEE Access*, vol. 10, pp. 71481–71495, 2022.

[29] D. Lan, A. Taherkordi, F. Eliassen, L. Liu, S. Delbruel, S. Dustdar, and Y. Yang, ''Task partitioning and orchestration on heterogeneous edge platforms: The case of vision applications,'' *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7418–7432, May 2022.

[30] L. Militano, A. Arteaga, G. Toffetti, and N. Mitton, ''The cloud-to-edge-to-IoT continuum as an enabler for search and rescue operations,'' *Future Internet*, vol. 15, no. 2, Jan. 2023, Art. no. 55.

[31] P. Sossalla, J. Hofer, J. Rischke, C. Vielhaus, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, ''DynNetSLAM: Dynamic visual SLAM network offloading,'' *IEEE Access*, vol. 10, pp. 116014–116030, 2022.

[32] G. Younes, D. C. Asmar, and E. A. Shammas, ''A survey on non-filter-based monocular visual slam systems,'' 2016, *arXiv:1607.00470*.

[33] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, ''ORB: An efficient alternative to SIFT or SURF,'' in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[34] E. Rosten and T. Drummond, ''Machine learning for high-speed corner detection,'' in *Proc. Europ. Conf. Comput. Vis.*, 2006, pp. 430–443.

[35] D. G. Lowe, ''Distinctive image features from scale-invariant keypoints,'' *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[36] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, ''Speeded-up robust features (SURF),'' *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[37] M. Calonder, V. Lepetit, C. Strecha, and P. V. Fua, ''BRIEF: Binary robust independent elementary features,'' in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2010, pp. 778–792.

[38] R. Mur-Artal and J. D. Tardós, ''ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,'' *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[39] L. Baroffio, A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, ''Coding local and global binary visual features extracted from video sequences,'' *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3546–3560, Nov. 2015.

[40] D. Van Opdenbosch, M. Oelsch, A. Garcea, and E. Steinbach, ''A joint compression scheme for local binary feature descriptors and their corresponding bag-of-words representation,'' in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2017, pp. 1–4.

[41] D. Van Opdenbosch, M. Oelsch, A. Garcea, T. Aykut, and E. Steinbach, ''Selection and compression of local binary features for remote visual SLAM,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7270–7277.

[42] D. Van Opdenbosch and E. Steinbach, ''Collaborative visual SLAM using compressed feature exchange,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 57–64, Jan. 2019.

[43] L. Riazuelo, J. Civera, and J. M. M. Montiel, ''C2TAM: A cloud framework for cooperative tracking and mapping,'' *Robot. Auto. Syst.*, vol. 62, no. 4, pp. 401–413, Apr. 2014.

[44] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, ''Cloud-based collaborative 3D mapping in real-time with low-cost robots,'' *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 423–431, Apr. 2015.

[45] A. J. B. Ali, M. Kouroshli, S. Semenova, Z. S. Hashemifar, S. Y. Ko, and K. Dantu, ''Edge-SLAM: Edge-assisted visual simultaneous localization and mapping,'' *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 1, pp. 1–31, Jan. 2023.

[46] A. J. B. Ali, Z. S. Hashemifar, and K. Dantu, ''Edge-SLAM: edge-assisted visual simultaneous localization and mapping,'' in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 325–337.

[47] Y. Chen, H. Inaltekin, and M. Gorlatova, "AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization," in *Proc. IEEE INFOCOM*, Sep. 2023, pp. 1–12.

[48] H. Cao, J. Xu, D. Li, L. Shangguan, Y. Liu, and Z. Yang, "Edge assisted mobile semantic visual SLAM," *IEEE Trans. Mobile Comput.*, early access, Aug. 24, 2022, doi: 10.1109/TMC.2022.3201000.

[49] J. Xu, H. Cao, D. Li, K. Huang, C. Qian, L. Shangguan, and Z. Yang, "Edge assisted mobile semantic visual SLAM," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 1828–1837.

[50] J. Xu, H. Cao, Z. Yang, L. Shangguan, J. Zhang, X. He, and Y. Liu, "SwarmMap: Scaling up real-time collaborative visual SLAM at the edge," in *Proc. USENIX NSDI*, Apr. 2022, pp. 977–993.

[51] S. Eger, R. Pries, and E. Steinbach, "Evaluation of different task distributions for edge cloud-based collaborative visual SLAM," in *Proc. IEEE 22nd Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2020, pp. 1–6.

[52] F. J. Romero-Ramirez, R. Muñoz-Salinas, M. J. Marín-Jiménez, M. Cazorla, and R. Medina-Carnicer, "SSLAM: speeded-up visual SLAM mixing artificial markers and temporary keypoints," *Sensors*, vol. 23, no. 4, Feb. 2023, Art. no. 2210.

[53] J. Chao, "Feature-preserving image and video compression," Ph.D. dissertation, TUM School Comput., Inf. Technol., Tech. Univ. Munich, Munich, Germany, 2016. [Online]. Available: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20160301-1277867-1-5

[54] D. Sharafutdinov, M. Griguletskii, P. Kopanev, M. Kurenkov, G. Ferrer, A. Burkov, A. Gonnochenko, and D. Tsetserukou, "Comparison of modern open-source visual SLAM approaches," *J. Intell. Robotic Syst.*, vol. 107, no. 3, Mar. 2023, Art. no. 43.

[55] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[56] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, Sep. 2016.

[57] J. Hofer, P. Sossalla, J. Rischke, C. L. Vielhaus, M. Reisslein, and H. P. and F. Fitzek, "Circular frame buffer to enhance map synchronization in edge assisted SLAM," in *Proc. IEEE ICC*, Rome, Italy, Jun. 2023, pp. 1–6.

[58] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," 2020, *arXiv:2007.11898*.

[59] P. Sossalla, J. Rischke, J. Hofer, and F. H. P. Fitzek, "Evaluating the advantages of remote SLAM on an edge cloud," in *Proc. 26th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2021, pp. 1–4.

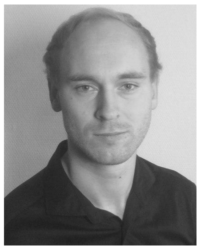
[60] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (IROS)*, Vilamoura, Portugal, Oct. 2012, pp. 573–580.

[61] S. Lee, "An efficient coverage area re-assignment strategy for multi-robot long-term surveillance," *IEEE Access*, vol. 11, pp. 33757–33767, 2023.

[62] E. A. A. Memon, S. R. U. N. Jafri, and S. M. U. Ali, "A rover team based 3D map building using low cost 2D laser scanners," *IEEE Access*, vol. 10, pp. 1790–1801, 2022.

JOHANNES HOFER (Graduate Student Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from Technische Universität Dresden, Germany, in 2022. His current research interests include cloud robotics and in particular the offloading of computation in SLAM systems.

PETER SOSSALLA (Graduate Student Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from Technische Universität Dresden (TU Dresden), in 2019, where he is currently pursuing the Ph.D. degree. He is a Development Engineer with Audi. His current research interests include robotics, V2X communication, and edge computing.



CHRISTIAN L. VIELHAUS received the Dipl.-Ing. degree in electrical engineering from Technische Universität Dresden (TU Dresden), Germany, in 2019, where he is currently pursuing the Ph.D. degree in electrical engineering with the Deutsche Telecom Chair of Communication Networks. His current research interests include machine learning, network simulation, and congestion control.

JUSTUS RISCHKE received the Dipl.Ing. degree in electrical engineering from Technische Universität Dresden (TU Dresden), Dresden, Germany, in 2017. He is currently pursuing the Ph.D. degree with the Deutsche Telekom Chair of Communication Networks. His current research interests include network coding and reinforcement learning in software-defined networks (SDN) for low-latency communication.

MARTIN REISSLEIN (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998. He is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA. He is an Associate Editor of IEEE Access and IEEE Transactions on Network and Service Management. He currently serves as the Area Editor for Optical Communications of IEEE Communications Surveys and Tutorials and a Co-Editor-in-Chief for *Optical Switching and Networking*.

FRANK H. P. FITZEK (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen University, Germany, in 1997, and the Ph.D. (Dr.-Ing.) degree in electrical engineering from Technische Universität Dresden, Germany, in 2002. He is currently a Professor and the Head of the Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, where he is coordinating the 5G Laboratory Germany. He is the Spokesman of the DFG Cluster of Excellence CeTI.