AdaptSLAM: Edge-Assisted Adaptive SLAM with Resource Constraints via Uncertainty Minimization

Ying Chen*, Hazer Inaltekin†, Maria Gorlatova*
*Duke University, Durham, NC, †Macquarie University, North Ryde, NSW, Australia
*{ying.chen151, maria.gorlatova}@duke.edu, †hazer.inaltekin@mq.edu.au

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10229009

***Abstract*—Edge computing is increasingly proposed as a solution for reducing resource consumption of mobile devices running simultaneous localization and mapping (SLAM) algorithms, with most edge-assisted SLAM systems assuming the communication resources between the mobile device and the edge server to be unlimited, or relying on heuristics to choose the information to be transmitted to the edge. This paper presents AdaptSLAM, an edge-assisted visual (V) and visual-inertial (VI) SLAM system that adapts to the available communication and computation resources, based on a theoretically grounded method we developed to select the subset of keyframes (the representative frames) for constructing the best local and global maps in the mobile device and the edge server under resource constraints. We implemented AdaptSLAM to work with the state-of-the-art open-source V- and VI-SLAM ORB-SLAM3 framework, and demonstrated that, under constrained network bandwidth, AdaptSLAM reduces the tracking error by 62% compared to the best baseline method.**

***Index Terms*—Simultaneous localization and mapping, edge computing, uncertainty quantification and minimization**

I. Introduction

Simultaneous localization and mapping (SLAM), the process of simultaneously constructing a map of the environment and tracking the mobile device's pose within it, is an essential capability for a wide range of applications, such as autonomous driving and robotic navigation [1]. In particular, visual (V) and visual-inertial (VI) SLAM, which use cameras either alone or in combination with inertial sensors, have demonstrated remarkable progress over the last three decades [2], and have become an indispensable component of emerging mobile applications such as drone-based surveillance [3], [4] and markerless augmented reality [5]–[8].

Due to the high computational demands placed by the V- and VI-SLAM on mobile devices [9]–[11], offloading parts of the workload to edge servers has emerged as a promising solution for lessening the loads on the mobile devices and improving the overall performance [9]–[17]. However, such approach experiences performance degradation under resource limitations and fluctuations. The existing edge-assisted SLAM solutions either assume wireless network resources to be sufficient for unrestricted offloading, or rely on heuristics in making offloading decisions. By contrast, in this paper we develop an edge computing-assisted SLAM framework, which we call AdaptSLAM, that intelligently adapts to both communication and computation resources to maintain high SLAM performance. Similar to prior work [10]–[16], AdaptSLAM runs a real-time tracking module and maintains a local map on the mobile device, while offloading non-time-critical and computationally expensive processes (global map optimization and loop closing) to the edge server. However, unlike prior designs, AdaptSLAM uses a *theoretically grounded method* to build the local and global maps of limited size, and minimize the uncertainty of the maps, laying the foundation for the optimal adaptive offloading of SLAM tasks under the communication and computation constraints.

First, we develop an uncertainty quantification model for the local and global maps in edge-assisted V-SLAM and VI-SLAM. Specifically, since these maps are built from the information contained in the keyframes (i.e., the most representative frames) [18]–[20], the developed model characterizes how the keyframes and the connections between them contribute to the uncertainty. *To the best of our knowledge, this is the first uncertainty quantification model for V-SLAM and VI-SLAM in edge-assisted architectures.*

Next, we apply the developed uncertainty quantification model to efficiently select subsets of keyframes to build local and global maps under the constraints of limited computation and communication resources. The local and global map construction is formulated as NP-hard cardinality-constrained combinatorial optimization problems [21]. We demonstrate that the map construction problems are 'close to' submodular problems under some conditions, *propose a low-complexity greedy-based algorithm to obtain near-optimal solutions*, and present a computation reuse method to speed up map construction. We implement AdaptSLAM in conjunction with the state-of-the-art V- and VI-SLAM ORB-SLAM3 [19] framework, and evaluate the implementation with both simulated and real-world communication and computation conditions. Under constrained bandwidth, AdaptSLAM reduces the tracking error by 62% compared with the best baseline method.

To summarize, the main contributions of this paper are: (i) the first uncertainty quantification model of maps in V- and VI-SLAM under the edge-assisted architecture, (ii) an analytically grounded algorithm for efficiently selecting subsets of keyframes to build local and global maps under computation and communication resource budgets, and (iii) a comprehensive evaluation of AdaptSLAM on two configurations of mobile devices. We open-source AdaptSLAM via GitHub.[1]

The rest of this paper is organized as follows. §II reviews the related work, §III provides the preliminaries, §IV and §V introduce the AdaptSLAM system architecture and model,

[1]https://github.com/i3tyc/AdaptSLAM

§VI presents the problem formulation, and §VII presents the problem solutions. We present the evaluation in §VIII and conclude the paper in §IX.

II. Related Work

V- and VI-SLAM. Due to the affordability of cameras and the richness of information provided by them, V-SLAM has been widely studied in the past three decades [2]. It can be classified into direct approaches (LSD-SLAM [22], DSO [23]), which operate directly on pixel intensity values, and feature-based approaches (PTAM [24], ORB-SLAM2 [18], Pair-Navi [25]), which extract salient regions in each camera frame. We focus on feature-based approaches since direct approaches require high computing power for real-time performance [2]. To provide robustness (to textureless areas, motion blur, illumination changes), there is a growing trend of employing VI-SLAM, that assists the cameras with an inertial measurement unit (IMU) [19], [20], [26]; VI-SLAM has become the de-facto standard SLAM method for modern augmented reality platforms [5], [6]. In VI-SLAM, visual information and IMU data can be loosely [26] or tightly [19], [20] coupled. We implement AdaptSLAM based on ORB-SLAM3 [19], a state-of-the-art open-source V- and VI-SLAM system which tightly integrates visual and IMU information.

Edge-assisted SLAM. Recent studies [4], [9]–[17], [27]–[29] have focused on offloading parts of SLAM workloads from mobile devices to edge (or cloud) servers to reduce mobile device resource consumption. A standard approach is to offload computationally expensive tasks (global map optimization, loop closing), while exploiting onboard computation for running the tasks critical to the mobile device's autonomy (tracking, local map optimization) [10]–[17]. Most edge-assisted SLAM frameworks assume wireless network resources to be sufficient for unconstrained offloading [4], [12]–[15], [28]; some use heuristics to choose the information to be offloaded under communication constraints [9]–[11], [16], [17], [27], [29]. Some frameworks only keep the newest keyframes in the local map to combat the constrained computation resources on mobile devices [13], [15]. Complementing this work, we propose a theoretical framework to characterize how keyframes contribute to the SLAM performance, laying the foundation for the adaptive offloading of SLAM tasks under the communication and computation constraints.

Uncertainty quantification and minimization. Recent work [30]–[32] has focused on quantifying and minimizing the pose estimate uncertainty in V-SLAM. Since the pose estimate accuracy is difficult to obtain due to the lack of ground-truth poses of mobile devices, the uncertainty can guide the decision-making in SLAM systems. In [30], [31], it is used for measurement selection (selecting measurements between keyframes [30] and selecting extracted features of keyframes [31]); in [32], it is used for anchor selection (selecting keyframes to make their poses have 'zero uncertainty'). Complementing this work, we quantify the pose estimate uncertainty of both V- and VI-SLAM under the edge-assisted architecture. After the uncertainty quantification, we study the problem of selecting a subset of keyframes to minimize the uncertainty. This problem is largely overlooked in the literature, but is of great importance for tackling computation and communication constraints in edge-assisted SLAM.

III. Preliminaries

A. Graph Preliminaries

A directed multigraph is defined by the tuple of sets $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the set of nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{C}$ is the set of edge categories. Let $e = ((v_i, v_j), c) \in \mathcal{E}$ denote the edge, where the nodes $v_i, v_j \in \mathcal{V}$ are the head and tail of e, and $c \in \mathcal{C}$ is the category of e. We let w_e be the weight of edge e. We allow multiple edges from v_i to v_j to exist, and denote the set of edges from v_i to v_j by $\mathcal{E}_{i,j}$. Note that the edges in $\mathcal{E}_{i,j}$ are differentiated from each other by their category labels. The total edge weight from nodes v_i to v_j is given by $w_{i,j} = \sum\limits_{e \in \mathcal{E}_{i,j}} w_e$, which is the sum of all edge weights from v_i to v_j.

The weighted Laplacian matrix $\mathbf{L}$ of graph $\mathcal{G}$ is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix where the i,j-th element $\mathbf{L}_{i,j}$ is given by:

$$\mathbf{L}_{i,j} = \begin{cases} -w_{i,j}, & i \neq j \\ \sum\limits_{e \in \mathcal{E}_i} w_e, & i = j \end{cases},$$

where $\mathcal{E}_i \subseteq \mathcal{E}$ is the set of all edges whose head is node v_i. The reduced Laplacian matrix $\tilde{\mathbf{L}}$ is obtained by removing an arbitrary node (i.e., removing the row and column associated to the node) from $\mathbf{L}$.

B. Set Function

We define a set function f for a finite set V as a mapping $f : 2^V \to \mathbb{R}$ that assigns a value $f(S)$ to each subset $S \subseteq V$.

Submodularity. A set function f is submodular if $f(L) + f(S) \geqslant f(L \cup S) + f(L \cap S)$ for all $L, S \subseteq V$.

Submodularity ratio. The submodularity ratio of a set function f with respect to a parameter s is

$$\gamma = \min_{L \subseteq V, S \subseteq V, |S| \leqslant s, x \in V \setminus (S \cup L)} \frac{f(L \cup \{x\}) - f(L)}{f(L \cup S \cup \{x\}) - f(L \cup S)}. \tag{1}$$

where we define $0/0 := 1$.

The cardinality-fixed maximization problem is

$$\max_{S \subseteq V, |S| = s} f(S). \tag{2}$$

The keyframe selection optimization is closely related to the cardinality-fixed maximization problem introduced above, which is an NP-hard problem [33]. However, for submodular set functions, there is an efficient greedy approach that will come close to the optimum value for (2), with a provable optimality gap. This result is formally stated in Theorem 1.

Algorithm 1 Greedy algorithm to solve (2)

```
1: $S^{\#} \leftarrow \emptyset$;
2: while ($|S^{\#}| < s$) do
3:     $x^\star \leftarrow \arg\max_x f(S^{\#} \cup \{x\}) - f(S^{\#})$. $S^{\#} \leftarrow S^{\#} \cup \{x^\star\}$.
```

Theorem 1. *[33], [34] Given a non-negative and monotonically increasing set function f with a submodularity ratio γ, let $S^{\#}$ be the solution produced by the greedy algorithm (Algorithm 1) and $S^{\star}$ be the solution of* (2)*. Then, $f\left(S^{\#}\right) \geqslant (1-\exp(-\gamma))\, f\left(S^{\star}\right)$.*

C. SLAM Preliminaries

The components of SLAM systems include [2], [19], [20]:

Tracking. The tracking module detects 2D *feature points* (e.g., SIFT, SURF, or ORB) in the current frame. Each feature point corresponds to a 3D *map point* (a distinguishable landmark) in the environment. The tracking module uses these feature points to find correspondences with a previous reference frame. It also processes the IMU measurements. Based on the correspondences in feature points and the IMU measurements, it calculates the relative pose change between the selected reference frame and the current frame. The module also determines if this frame should be a keyframe based on criteria such as the similarity to the previous keyframes [19].

Local and global mapping. It finds correspondences (of feature points) between the new keyframe and the other keyframes in the map. It then performs map optimizations, i.e., estimates the keyframe poses given the common feature points between the keyframes and the IMU measurements. *Map optimizations are computationally expensive.* In edge-assisted SLAM, global mapping runs on the server [10]–[16].

Loop closing. By comparing the new keyframe to all previous keyframes, the module checks if the new keyframe is revisiting a place. If so (i.e., if a loop is detected), it establishes connections between the keyframe and all related previous ones, and then performs global map optimizations. Loop closing is computationally expensive and can be offloaded to the edge server in the edge-assisted SLAM [10]–[16].

IV. AdaptSLAM System Architecture

The design of AdaptSLAM is shown in Fig. 1. The mobile device, equipped with a camera and an IMU, can communicate with the edge server bidirectionally. The mobile device and the edge server cooperatively run SLAM algorithms to estimate the mobile device's pose and a map of the environment. AdaptSLAM optimizes the SLAM performance under computation resource limits of the mobile device and communication resource limits between the mobile device and the edge server.

We split the modules between the mobile device and the edge server similar to [10], [12]–[17]. The mobile device offloads loop closing and global map optimization modules to the edge server, while running real-time tracking and local mapping onboard. Unlike existing edge-assisted SLAM systems [10], [12]–[17], *AdaptSLAM aims to optimally construct the local and global maps under the computation and communication resource constraints.* The design of AdaptSLAM is mainly focused on two added modules, local map construction and global map construction highlighted in purple in Fig. 1. In local map construction, due to the computation resource limits, the mobile device selects a subset of keyframes from candidate keyframes to build a local map. In global map

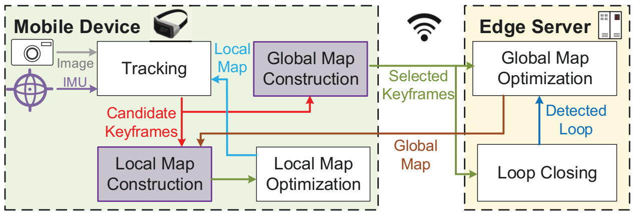


Fig. 1: Overview of the AdaptSLAM system architecture.

construction, to adapt to the constrained wireless connection for uplink transmission, the mobile device also selects a subset of keyframes to be transmitted to the edge server to build a global map. The AdaptSLAM optimally selects the keyframes to build local and global maps, minimizing the pose estimate uncertainty under the resource constraints.

Similar to [10], the selected keyframes are transmitted from the mobile device to the server, and the map after the global map optimization is transmitted from the server to the mobile device. For the uplink transmission, instead of the whole keyframe, the 2D feature points extracted from the keyframes are sent. For the downlink communication, the poses of the keyframes obtained by the global map optimization, and the feature points of the keyframes are transmitted.

V. AdaptSLAM System Model

A. The Pose Graph and the Map

We divide time into slots of equal size of Δt. We introduce the pose graph and the map at time slot t that lasts for Δt seconds. For clarity of notation, we will omit the time index below.

Definition 1 (Pose graph)**.** *For a given index set $\mathcal{K} = \{1,\ldots,|\mathcal{K}|\}$ (indexing camera poses and representing keyframes), the pose graph is defined as the undirected multi-graph $\mathcal{G}=(\mathcal{K},\mathcal{E},\mathcal{C})$, where $\mathcal{K}$ is the node set, $\mathcal{E}$ is the edge set, and $\mathcal{C}=\{\mathsf{IMU},\mathsf{vis}\}$ is the category set. Here,* IMU *stands for the IMU edges, and* vis *stands for the covisibility edges.*

Given a pose graph $\mathcal{G}=(\mathcal{K},\mathcal{E},\mathcal{C})$, there is a camera pose $\mathbf{P}_n = (x,y,z,w_x,w_y,w_z)$ for all $n \in \mathcal{K}$, where the first three entries are the 3-D positions and the last three ones are the Euler angles (yaw, pitch and roll) representing the camera orientation. Edges in $\mathcal{E}$ are represented as $e=((n,m),c)$ for $n,m \in \mathcal{K}$ and $c \in \mathcal{C}$. Two keyframes in $\mathcal{K}$ are connected by a covisibility edge if there are 3D map points observed in both keyframes. Two consecutive keyframes are connected by an IMU edge if there are accelerometer and gyroscope readings from one keyframe to another. There may exist both a covisibility edge and an IMU edge between two keyframes.

For each $e=((n,m),c) \in \mathcal{E}$, we observe relative noisy pose measurements between n and m, which is written as $\Delta_e = \mathbf{P}_m - \mathbf{P}_n + \mathbf{x}_e$, where $\mathbf{x}_e$ is the measurement noise on edge e. The map optimization problem is to find the maximum likelihood estimates $\{\tilde{\mathbf{P}}_n\}_{n\in\mathcal{K}}$ for the actual camera poses $\{\mathbf{P}_n\}_{n\in\mathcal{K}}$. For Gaussian distributed edge noise, the map optimization problem is

$$\min_{\{\tilde{\mathbf{P}}_n\}_{n\in\mathcal{K}}} \sum_{e\in\mathcal{E}} \left(\tilde{\mathbf{x}}_e\right)^{\top} \mathcal{I}_e \tilde{\mathbf{x}}_e, \tag{3}$$

where $\tilde{\mathbf{x}}_e = \Delta_e - \tilde{\mathbf{P}}_m + \tilde{\mathbf{P}}_n$ and $\mathcal{I}_e$ is the information matrix (i.e., inverse covariance matrix) of the measurement error on e [35]. $(\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e$ is the Mahalanobis norm [19], [20] of the *estimated* measurement noise for e with respect to $\mathcal{I}_e$.

Below, we assume that the measurement noise $\mathbf{x}_e$ is Gaussian distributed with isotropic covariance (as in [30], [36], [37]). We assume that the information matrix $\mathcal{I}_e$ can be characterized by a weight assigned to e [36], [38]. Specifically, $\mathcal{I}_e = w_e \mathcal{I}$, where $w_e \geqslant 1$ is the weight for e and $\mathcal{I}$ is the matrix that is constant for all measurements. We note that the relative measurements between keyframes n and m introduce the same information for them. We assume all weights w_e to be independent from each other for edges between different pairs of keyframes as in [19], [20], [38], [39].

The map optimization problem in (3) is solved by standard methods such as Levenberg-Marquardt algorithm implemented in g2o [40] and Ceres solvers [41] as in [19], [20].

Definition 2 (Anchor). *We say that a node is the anchor of the pose graph if the pose of the node is known.*

The map (local or global) consists of the pose graph (in Definition 1) and map points in the environment. In this paper, we will use the terms map and pose graph interchangeably. Without loss of generality, we will also assume that the global (or local) map is anchored on the first node, as in [36], [38]. This assumption is made because SLAM can only estimate the relative pose change based on the covisibility and inertial measurements, while the absolute pose estimate in the global coordinate system cannot be provided.

B. The Local Map

Local map construction. The candidate keyframes are selected from camera frames according to the selection strategy in ORB-SLAM3 [19], and these candidate keyframes form the set $\mathcal{K}$. Due to the constrained computation resources, the mobile device selects a fixed keyframe set $\mathcal{K}_{fixed}$ and a local keyframe set $\mathcal{K}_{loc}$ from the candidate keyframes, where $|\mathcal{K}_{fixed}| \leqslant l_f$ and $|\mathcal{K}_{loc}| \leqslant l_{loc}$. The fixed keyframe set $\mathcal{K}_{fixed} \subseteq \mathcal{K}_{g,user}$ is selected from the global map $\mathcal{K}_{g,user}$ that was last transmitted from the edge server. The poses of keyframes in $\mathcal{K}_{fixed}$ act as fixed priors in the local map optimization. This is because poses of keyframes in $\mathcal{K}_{g,user}$ are already optimized in the global map optimization and hence have low uncertainty. The poses of keyframes in the local keyframe set $\mathcal{K}_{loc} \subseteq \mathcal{K} \setminus \mathcal{K}_{g,user}$ will be optimized according to the map optimization problem introduced above.

The edges between keyframes in $\mathcal{K}_{loc}$ form the set $\mathcal{E}_{loc}$, and the edges whose one node belongs to $\mathcal{K}_{loc}$ and another node belongs to $\mathcal{K}_{fixed}$ form the set $\mathcal{E}_{l,f}$.

Local map optimization. After selecting $\mathcal{K}_{loc}$ in the local map construction, the local map optimization is to optimize the estimated poses $\left\{\tilde{\mathbf{P}}_n\right\}_{n \in \mathcal{K}_{loc}}$ to minimize the sum of Mahalanobis norms $\sum_{e \in \mathcal{E}_{loc} \cup \mathcal{E}_{l,f}} (\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e$. Note that in the local pose graph optimization, the keyframes in $\mathcal{K}_{fixed}$ are included in the optimization with their poses fixed. The local map optimization to solve (3) is

$$\min_{\{\tilde{\mathbf{P}}_n\}_{n \in \mathcal{K}_{loc}}} \sum_{e \in \mathcal{E}_{loc} \cup \mathcal{E}_{l,f}} (\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e. \tag{4}$$

C. The Global Map

Global map construction. Due to the limited bandwidth between the mobile device and the edge server, only a subset of candidate keyframes are offloaded to the edge server to build a global map. The selection of keyframes to be offloaded will be optimized to minimize the pose estimation uncertainty of the global map when considering the underlying wireless network constraints.

The edge server maintains the global map, denoted as $\mathcal{K}_{g,edge}$, holding all keyframes uploaded by the mobile device. The edges between keyframes in the global map $\mathcal{K}_{g,edge}$ constitute the set $\mathcal{E}_{glob}$. Note that $\mathcal{K}_{g,edge}$ may be different from $\mathcal{K}_{g,user}$, because the global map is large and it takes time to transmit the most up-to-date global map from the edge server to the mobile device.

Global map optimization. After selecting $\mathcal{K}_{g,edge}$ in the global map construction, the edge server performs the global map optimization to estimate poses $\tilde{\mathbf{P}}_n$ in $\mathcal{K}_{g,edge}$ and minimize the sum of Mahalanobis norms $\sum_{e \in \mathcal{E}_{glob}} (\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e$. Specifically, the edge solves (3) when $\mathcal{E} = \mathcal{E}_{glob}$ and $\mathcal{K} = \mathcal{K}_{g,edge}$, i.e., the global map optimization is to solve

$$\min_{\{\tilde{\mathbf{P}}_n\}_{n \in \mathcal{K}_{g,edge}}} \sum_{e \in \mathcal{E}_{glob}} (\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e. \tag{5}$$

VI. Problem Formulation

AdaptSLAM aims to efficiently select keyframes to construct optimal local and global maps, i.e., we select keyframes in $\mathcal{K}_{loc}$ and $\mathcal{K}_{fixed}$ for the local map and $\mathcal{K}_{g,edge}$ for the global map. We construct optimal local and global maps by minimizing the uncertainty of the keyframes' estimated poses.

A. Uncertainty Quantification

Let $\mathbf{p}_n = \tilde{\mathbf{P}}_n - \mathbf{P}_n$ denote the pose estimate error of keyframe n. The estimated measurement noise can be rewritten as $\tilde{\mathbf{x}}_e = \mathbf{p}_n - \mathbf{p}_m + \mathbf{x}_e = \mathbf{p}_{n,m} + \mathbf{x}_e$, where $\mathbf{p}_{n,m} = \mathbf{p}_n - \mathbf{p}_m$. We stack all $\mathbf{p}_n, n \in \mathcal{K}$ and get a pose estimate error vector $\mathbf{w} = \left(\mathbf{p}_1^\top, \mathbf{p}_2^\top, \cdots, \mathbf{p}_{|\mathcal{K}|}^\top\right)$. We rewrite the objective function of map optimization in (3) as $\sum_{e \in \mathcal{E}} (\tilde{\mathbf{x}}_e)^\top \mathcal{I}_e \tilde{\mathbf{x}}_e = \sum_{e=((n,m),c) \in \mathcal{E}} \mathbf{p}_{n,m}^\top \mathcal{I}_e \mathbf{p}_{n,m} + 2 \sum_{e=((n,m),c) \in \mathcal{E}} \mathbf{p}_{n,m}^\top \mathcal{I}_e \mathbf{x}_e + \sum_{e \in \mathcal{E}} \mathbf{x}_e^\top \mathcal{I}_e \mathbf{x}_e$. If we can rewrite the quadratic term $\sum_{e=((n,m),c) \in \mathcal{E}} \mathbf{p}_{n,m}^\top \mathcal{I}_e \mathbf{p}_{n,m}$ in the format of $\mathbf{w} \mathcal{I}_w \mathbf{w}^\top$, where $\mathcal{I}_w$ is called the information matrix of the pose graph, the uncertainty of the pose graph is quantified by $-\log\det(\mathcal{I}_w)$ according to the D-optimality [30]–[32].[2]

We denote the pose estimate error vectors for the global and local maps as $\mathbf{w}_g = \left(\mathbf{p}_{u_1}^\top, \cdots, \mathbf{p}_{u_{|\mathcal{K}_{g,edge}|}}^\top\right)$ and $\mathbf{w}_l = \left(\mathbf{p}_{r_1}^\top, \cdots, \mathbf{p}_{r_{|\mathcal{K}_{loc}|}}^\top\right)$, where $u_1, \cdots, u_{|\mathcal{K}_{g,edge}|}$ are the keyframes in $\mathcal{K}_{g,edge}$, and $r_1, \cdots, r_{|\mathcal{K}_{loc}|}$ are the keyframes in $\mathcal{K}_{loc}$. The first pose in the global and local pose graph is known ($\mathbf{p}_{u_1} = 0$, $\mathbf{p}_{r_1} = 0$). We rewrite the quadratic terms of the objective functions of global and local map optimizations in (5) and (4) as $\sum_{e=((n,m),c)\in\mathcal{E}_{glob}} \mathbf{p}_{n,m}^\top \mathcal{I}_e \mathbf{p}_{n,m} = \mathbf{w}_g \mathcal{I}_{glob}(\mathcal{K}_{g,edge}) \mathbf{w}_g^\top$ (or $\sum_{e=((n,m),c)\in\mathcal{E}_{loc}\cup\mathcal{E}_{l,f}} \mathbf{p}_{n,m}^\top \mathcal{I}_e \mathbf{p}_{n,m} = \mathbf{w}_l \mathcal{I}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed}) \mathbf{w}_l^\top$), where $\mathcal{I}_{glob}(\mathcal{K}_{g,edge})$ and $\mathcal{I}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed})$ are called the information matrices of the global and local maps and will be derived later (in Definition 3 and Lemmas 1 and 2).

Definition 3 (Uncertainty)**.** *The uncertainty of the global (or local) pose graph is defined as $-\log\det\left(\tilde{\mathcal{I}}_{glob}(\mathcal{K}_{g,edge})\right)$ (or $-\log\det\left(\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed})\right)$, where $\tilde{\mathcal{I}}_{glob}(\mathcal{K}_{g,edge})$ and $\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed})$ are obtained by removing the first row and first column in the information matrices $\mathcal{I}_{glob}(\mathcal{K}_{g,edge})$ and $\mathcal{I}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed})$.*

From Definition 3, the uncertainty quantification is based on the global and local map optimizations introduced in §V-C and §V-B. After quantifying the uncertainty, we will later (in §VI-B) optimize the local and global map construction which in turn minimizes the uncertainty of poses obtained from local and global map optimizations.

Lemma 1 (Uncertainty of global pose graph)**.** *For the global map optimization, the uncertainty is calculated as $-\log\det\left(\tilde{\mathcal{I}}_{glob}(\mathcal{K}_{g,edge})\right)$, where $\tilde{\mathcal{I}}_{glob}(\mathcal{K}_{g,edge}) = \tilde{\mathbf{L}}_{glob} \otimes \mathcal{I}$ with $\tilde{\mathbf{L}}_{glob}$ being the matrix obtained by deleting the first row and column in the Laplacian matrix $\mathbf{L}_{glob}$, and $\otimes$ being the Kronecker product. The i,j-th element of $\mathbf{L}_{glob}$ is given by*

$$[\mathbf{L}_{glob}]_{i,j} = \begin{cases} -\sum\limits_{e=((u_i,u_j),c)\in\mathcal{E}_{g,edge}} w_e, & i \neq j \\ \sum\limits_{e=((u_i,q),c)\in\mathcal{E}_{g,edge}, u_i\neq q} w_e, & i = j \end{cases}. \tag{6}$$

Proof. See Appendix A in the technical report [44]. □

From Lemma 1, the uncertainty of the global pose graph can be calculated based on the reduced Laplacian matrix ($\tilde{\mathbf{L}}_{glob}$). According to the relationship between the reduced Laplacian matrix and the tree structure [45], the uncertainty is inversely proportional to the logarithm of weighted number of spanning trees in the global pose graph. Similar conclusions are drawn for 2D pose graphs [30] and 3D pose graphs with only covisibility edges [36], [38], where the device can move in 2D plane and 3D space respectively. We extend the results to VI-SLAM where the global pose graph is a multigraph with the possibility of having both a covisibility edge and an IMU edge between two keyframes.

Lemma 2 (Uncertainty of local pose graph)**.** *The uncertainty is $-\log\det\left(\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed})\right)$ for the local map, where $\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc}, \mathcal{K}_{fixed}) = \tilde{\mathbf{L}}_{loc} \otimes \mathcal{I}$ with $\tilde{\mathbf{L}}_{loc}$ being the matrix obtained by deleting the first row and the first column in $\mathbf{L}_{loc}$. The i,j-th element of $\mathbf{L}_{loc}$ (of size $|\mathcal{K}_{loc}| \times |\mathcal{K}_{loc}|$) is given by*

$$[\mathbf{L}_{loc}]_{i,j} = \begin{cases} -\sum\limits_{e=((r_i,r_j),c)\in\mathcal{E}_{loc}} w_e, & i \neq j \\ \sum\limits_{e=((r_i,q),c)\in\mathcal{E}_{l,f}\cup\mathcal{E}_{loc}, q\neq r_i} w_e, & i = j \end{cases}. \tag{7}$$

Proof. See Appendix B in the technical report [44]. □

From Lemma 2, the uncertainty of the local map is proportional to the uncertainty of the pose graph $\mathcal{G}$ anchoring on the first node in $\mathcal{K}_{loc}$ and all nodes in $\mathcal{K}_{fixed}$, where $\mathcal{G}$'s node set is $\mathcal{K}_{fixed} \cup \mathcal{K}_{loc}$ and edge set includes all measurements between any two nodes in $\mathcal{K}_{fixed} \cup \mathcal{K}_{loc}$. Note that keyframe poses in $\mathcal{K}_{fixed}$ are optimized on the edge server and transmitted to the mobile device, and they are considered as constants in the local pose graph optimization. From the uncertainty's perspective, adding fixed keyframes in $\mathcal{K}_{fixed}$ is equivalent to anchoring these keyframe poses (i.e., deleting rows and columns corresponding to the anchored nodes in the Laplacian matrix of graph $\mathcal{G}$). In addition, from Lemma 2, although poses are fixed, the anchored nodes still reduce the uncertainty of the pose graph. Hence, apart from $\mathcal{K}_{loc}$, we will select the anchored keyframe set $\mathcal{K}_{fixed}$ to minimize the uncertainty.

B. Uncertainty Minimization Problems

We now formulate optimization problems whose objectives are to minimize the uncertainty of the local and global maps. For the local map optimization, under the computation resource constraints, we solve Problem 1 for each keyframe k. For the global map optimization, under the communication resource constraints, we solve Problem 2 to adaptively offload keyframes to the edge server.

Problem 1 (Local map construction)**.**

$$\max_{\mathcal{K}_{loc}, \mathcal{K}_{fixed}} \log\det\left(\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc} \cup \{k\}, \mathcal{K}_{fixed})\right) \tag{8}$$

$$\textit{s.t.} \quad |\mathcal{K}_{loc}| \leqslant l_{loc}, \mathcal{K}_{loc} \subseteq \mathcal{K} \setminus \mathcal{K}_{g,user} \tag{9}$$

$$|\mathcal{K}_{fixed}| \leqslant l_f, \mathcal{K}_{fixed} \subseteq \mathcal{K}_{g,user}. \tag{10}$$

The objective of Problem 1 is equivalent to minimizing the uncertainty of the local map. Constraint (9) means that the size of $\mathcal{K}_{loc}$ is constrained to reduce the computational complexity in the local map optimization, and that the keyframes to be optimized in the local map are selected from keyframes that are not in $\mathcal{K}_{g,user}$. Constraint (10) means that the size of $\mathcal{K}_{fixed}$ is constrained, and that the fixed keyframes are selected from $\mathcal{K}_{g,user}$ that were previously optimized on and transmitted from the edge server.

[2]Common approaches to quantifying uncertainty in SLAM are to use real scalar functions of the maximum likelihood estimator covariance matrix [42]. Among them, D-optimality (determinant of the covariance matrix) [36], [38] captures the uncertainty due to all the elements of a covariance matrix and has well-known geometrical and information-theoretic interpretations [43].

Problem 2 (Global map construction).

$$\max_{\mathcal{K}' \subseteq \mathcal{K} \setminus \mathcal{K}_{g,edge}} \log\det\left(\widetilde{\mathcal{I}}_{glob}\left(\mathcal{K}_{g,edge} \cup \mathcal{K}'\right)\right) \tag{11}$$

$$s.t. \quad d\left|\mathcal{K}'\right| \leqslant D. \tag{12}$$

The objective of Problem 2 is equivalent to minimizing the uncertainty of the global map. $\mathcal{K} \setminus \mathcal{K}_{g,edge}$ is set of the keyframes that have not been offloaded to the server, and we select a subset of keyframes, $\mathcal{K}'$, from $\mathcal{K} \setminus \mathcal{K}_{g,edge}$. The constraint (12) guarantees that the keyframes cannot be offloaded from the device to the server at a higher bitrate than the available channel capacity, where D is the channel capacity constraint representing the maximum number of bits that can be transmitted in a given transmission window. We assume that the data size d of each keyframe is the same, which is based on the observation that the data size is relatively consistent across keyframes in popular public SLAM datasets [46], [47].

VII. Local and Global Map Construction

We analyze the properties of approximate submodularity in map construction problems, and propose low-complexity algorithms to efficiently construct local and global maps.

A. Local Map Construction

The keyframes in the local map include those in two disjoint sets $\mathcal{K}_{loc}$ and $\mathcal{K}_{fixed}$. To efficiently solve Problem 1, we decompose it into two problems aiming at minimizing the uncertainty: Problem 3 that selects keyframes in $\mathcal{K}_{loc}$ and Problem 4 that selects keyframes in $\mathcal{K}_{fixed}$. We obtain the optimal local keyframe set $\mathcal{K}_{loc}^{\star}$ in Problem 3. Based on $\mathcal{K}_{loc}^{\star}$, we then obtain the optimal fixed keyframe set $\mathcal{K}_{fixed}^{\star}$ in Problem 4. We will compare the solutions to Problems 3 and 4 with the optimal solution to Problem 1 in §VIII to show that the performance loss induced by the decomposition is small.

Problem 3.

$$\mathcal{K}_{loc}^{\star} = \arg\max_{\mathcal{K}_{loc}} \log\det\left(\widetilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc} \cup \{k\}, \emptyset)\right)$$
$$s.t. \quad (9).$$

Problem 4.

$$\mathcal{K}_{fixed}^{\star} = \arg\max_{\mathcal{K}_{fixed}} \log\det\left(\tilde{\mathcal{I}}_{loc}(\mathcal{K}_{loc}^{\star} \cup \{k\}, \mathcal{K}_{fixed})\right)$$
$$s.t. \quad (10).$$

1) The Selection of Local Keyframe Set $\mathcal{K}_{loc}$: We first solve Problem 3. It is a nonsubmodular optimization problem with constraints, which are NP-hard and generally difficult to be solved with an approximation ratio [21]. Hence, we decompose Problem 3 into subproblems (Problems 5 and 6) that are equivalent to the original Problem 3 and can be approximately solved with a low-complexity algorithm.

In problem 5, assume that we already select a keyframe subset $\mathcal{K}_{base}$ from $\mathcal{K} \setminus \mathcal{K}_{g,user}$ (with the size $l_b \triangleq |\mathcal{K}_{base}| \leqslant l_{loc}$), and we aim to further select a keyframe set $\mathcal{K}_{add}$ to be added to $\mathcal{K}_{base}$ to minimize the local map uncertainty. Rewriting the objective as $\mathsf{Unc}\left(\mathcal{K}_{add} \cup \mathcal{K}_{base} \cup \{k\}\right) \triangleq -\log\det\left(\widetilde{\mathcal{I}}_{loc}\left(\mathcal{K}_{add} \cup \mathcal{K}_{base} \cup \{k\}, \emptyset\right)\right)$, we aim to get the optimal $\mathcal{K}_{add}$ (denoted as $\mathsf{OPT}_{add}(\mathcal{K}_{base})$) given $\mathcal{K}_{base}$:

Problem 5.

$$\mathsf{OPT}_{add}\left(\mathcal{K}_{base}\right) = \arg\max_{\mathcal{K}_{add}} -\mathsf{Unc}\left(\mathcal{K}_{add} \cup \mathcal{K}_{base} \cup \{k\}\right)$$
$$s.t. \quad |\mathcal{K}_{add}| \leqslant l_{loc} - l_b.$$

After getting the solutions (i.e., $\mathsf{OPT}_{add}\left(\mathcal{K}_{base}\right)$) to Problem 5 for all possible $\mathcal{K}_{base}$ of size l_b, we obtain the optimal $\mathcal{K}_{base}$ (denoted as $\mathcal{K}_{base}^{\star}$) in Problem 6.

Problem 6.

$$\mathcal{K}_{base}^{\star} = \arg\max_{\mathcal{K}_{base}} -\mathsf{Unc}\left(\mathsf{OPT}_{add}(\mathcal{K}_{base}) \cup \mathcal{K}_{base} \cup \{k\}\right)$$
$$s.t. \quad |\mathcal{K}_{base}| = l_b.$$

Lemma 3. *Given $l_b \geqslant 0$, let $\mathcal{K}_{base}^{\star}$ be the solution for Problem 6 and $\mathsf{OPT}_{add}\left(\mathcal{K}_{base}^{\star}\right)$ be the solution for Problem 5 for the input set $\mathcal{K}_{base}^{\star}$. Then, $\mathcal{K}_{loc}^{\star}$ is given by $\mathcal{K}_{loc}^{\star} = \mathcal{K}_{base}^{\star} \cup \mathsf{OPT}_{add}\left(\mathcal{K}_{base}^{\star}\right)$.*

Proof. The proof is straightforward and hence omitted. □

We can obtain $\mathcal{K}_{loc}^{\star}$ in Problem 3 by solving Problems 5 and 6. We will show that the objective function of Problem 5 is 'close to' a submodular function when the size of the keyframe set $\mathcal{K}_{base}$ is large. In this case, Problem 5 can be efficiently solved using a greedy algorithm with an approximation ratio. When $|\mathcal{K}_{base}|$ is small, we need to compare the objective function for different combinations of $\mathcal{K}_{base}$ and $\mathcal{K}_{add}$.

Lemma 4. *When $\frac{w_{\max}}{|\mathcal{K}_{base}| w_{\min}} < 1$, the submodularity ratio γ of the objective function in Problem 5 is lower bounded by*

$$\gamma \geqslant 1 + \frac{1}{\vartheta} \log\left(1 - \frac{4|\mathcal{K}_{add}|^2 w_{\max}^2}{|\mathcal{K}_{base}|\, w_{\min} - w_{\max}}\right), \tag{13}$$

where $\vartheta = \min_{m \in \mathcal{K}_{add}} \sum_{n \in \mathcal{K}_{base}} \log w_{n,m}$, $w_{\max} = \max_{n,m \in \mathcal{K}_{base} \cup \mathcal{K}_{add}} w_{n,m}$, and $w_{\min} = \min_{n,m \in \mathcal{K}_{base} \cup \mathcal{K}_{add}} w_{n,m}$. γ is close to 1 when $|\mathcal{K}_{base}|$ is significantly larger than $|\mathcal{K}_{add}|$.

Proof. See Appendix C in the technical report [44]. □

From Lemma 4, the objective function in Problem 5 is 'close to' a submodular function when the size of the existing keyframe set (i.e., $|\mathcal{K}_{base}|$) is much larger than $|\mathcal{K}_{add}|$. Hence, we can use the greedy algorithm to approximately solve Problem 5. According to Theorem 1, the solution obtained by the greedy algorithm for Problem 5, denoted by $\mathsf{OPT}_{add}^{\#}\left(\mathcal{K}_{base}\right)$, has an approximation guarantee that $\mathsf{OPT}_{add}^{\#}\left(\mathcal{K}_{base}\right) \geqslant \left(1 - \exp(-\gamma)\right) \mathsf{OPT}_{add}\left(\mathcal{K}_{base}\right)$.

According to the analysis of the properties of Problems 5 and 6, we now solve Problem 3 to select the local keyframe set $\mathcal{K}_{loc}$ using Algorithm 2 (top-h greedy-based algorithm). Θ is the set of possible keyframe sets that minimize the local map uncertainty, and we only maintain h keyframe sets to save the computation resources. $\Lambda, \Lambda \in \Theta$, denotes the element in Θ and represents one possible keyframe set. When the

Algorithm 2 Selecting local keyframe set $\mathcal{K}_{loc}$ in the local map (top-h greedy-based algorithm)

```
1: $\Theta \leftarrow \emptyset$;
2: while ( $|\Lambda| \leqslant l_{loc}$) do
3:     if $|\Lambda| \leqslant l_{thr}$ then $h \leftarrow H$ else $h \leftarrow 1$;
4:     Select the top-$h$ highest-scoring combinations of $\Lambda, \Lambda \in \Theta$ and $n, n \in \mathcal{K} \setminus \mathcal{K}_{g,user}$ that minimize $\mathsf{Unc}\left(\Lambda \cup \{n, k\}\right)$. $\mathsf{Unc}\left(\Lambda \cup \{n, k\}\right)$ is calculated using the computation reuse algorithm in Algorithm 2;
5:     Update $\Theta$ as the set of $h$ highest-scoring combinations of $\Lambda$ and $n$. Each element of $\Theta$ is a set (i.e., $\Lambda \cup \{n\}$) corresponding to one combination;
6: $\mathcal{K}^{\star}_{loc} \leftarrow \arg\min_{\Lambda \in \Theta} \mathsf{Unc}(\Lambda \cup \{k\})$.
```

Algorithm 3 Computation reuse algorithm

```
1: Input: $\det(\mathbf{A})$, $\mathbf{A}^{-1}$;
2: $\mathbf{B} \leftarrow \mathbf{A}^{-1}$. Calculate $\mathbf{B}_i\mathbf{B}_i^{\top}$, $i = 1, \cdots, |\Lambda|$;
3: Calculate $(\mathbf{A}')^{-1}$ using (15). Calculate $\det(\mathbf{A}')$ using (16). Calculate $\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right))$ using (14).
```

size of Λ is smaller than a threshold l_{thr} ($|\Lambda| \leqslant l_{thr}$), we select the top-$H$ ($H > 1$) highest-scoring combinations of Λ and $n, n \in \mathcal{K} \setminus \mathcal{K}_{g,user}$, that minimize $\mathsf{Unc}\left(\Lambda \cup \{k, n\}\right)$. When $|\Lambda|$ gets larger, we only select the highest-scoring combination. The reasons are as follows. Λ can be seen as the existing keyframe set $\mathcal{K}_{base}$. According to Lemma 4, when the size of the existing keyframe set (which is $|\Lambda|$ here) is small, there is no guarantee that $\mathsf{Unc}\left(\mathcal{K}_{add} \cup \mathcal{K}_{base} \cup \{k\}\right)$ is close to a submodular function (i.e., the submodularity ratio is much smaller than 1). Hence, we need to try different combinations of Λ and n to search for the combination that minimizes the uncertainty after each iteration. As $|\Lambda|$ grows, the submodularity ratio is close to 1, and a greedy algorithm achieves η approximation ($\eta = 1 - \exp(-\gamma)$, $\gamma \to 1$). In this case, we apply the greedy algorithm and only keep the combination that achieves minimal uncertainty at each step.

2) Computation Reuse Algorithm: We use the computation reuse algorithm (Algorithm 3) to speed up Algorithm 2. We observe that for different $n, n \in \mathcal{K} \setminus \mathcal{K}_{g,user}$, only a limited number ($3|\Lambda|+1$) of elements in the matrix $\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right)$ are different. Calculating the log-determinant function of a $(|\Lambda|+1) \times (|\Lambda|+1)$ matrix $\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right)$ has a high computational complexity (of $\mathcal{O}(|\Lambda|+1)^3$) [48]. Hence, instead of computing the objective function for each n from scratch, we reuse parts of computation results for different n.

Letting $\mathbf{A} \triangleq \tilde{\mathcal{I}}\left(\Lambda \cup \{k\}\right)$ denote the information matrix of the local map in the $|\Lambda|$-th iteration (of Algorithm 2), the information matrix in the $(|\Lambda|+1)$-th iteration is $\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right) = \begin{bmatrix} \mathbf{A} + \mathrm{diag}\left(\mathbf{a}\right) & \mathbf{a}^{\top} \\ \mathbf{a} & d \end{bmatrix}$, where $\mathbf{a} = (a_1, a_2, \cdots, a_{|\Lambda|})$ with $a_i = w_{\lambda_i, n}$, λ_i is the i-th element of Λ, and $d = w_{k,n} + \sum_{i=1}^{|\Lambda|} a_i$.

We aim to calculate $\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right))$ using the calculation of $\det(\mathbf{A})$ and $\mathbf{A}^{-1}$ from the previous iteration. Letting $\mathbf{A}' \triangleq \mathbf{A} + \mathrm{diag}(\mathbf{a})$, $\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right))$ is calculated by

$$\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right)) = \left(d - \mathbf{a}(\mathbf{A}')^{-1}\mathbf{a}^{\top}\right)\det(\mathbf{A}'). \tag{14}$$

Next we efficiently calculate $(\mathbf{A}')^{-1}$ and $\det(\mathbf{A}')$ to get $\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right))$. We can rewrite $\mathbf{A}'$ as $\mathbf{A}' = \mathbf{A} + \sum_{i=1}^{|\Lambda|} \beta_i^{\top}\beta_i$ where $\beta_i = \left(0, \cdots, \underbrace{\sqrt{a_i}}_{i-\text{th}}, \cdots, 0\right)$. According to Sherman–Morrison formula [49], $(\mathbf{A}')^{-1}$ is given by

$$(\mathbf{A}')^{-1} \approx \underbrace{\mathbf{B}}_{\text{Reuse}} - \sum_{i=1}^{|\Lambda|} \frac{a_i}{1 + a_i \mathbf{B}_{i,i}} \underbrace{\mathbf{B}_i\mathbf{B}_i^{\top}}_{\text{Reuse}}, \tag{15}$$

where $\mathbf{B} = \mathbf{A}^{-1}$, $\mathbf{B}_{i,i}$ is the i,i-th element of $\mathbf{B}$, and $\mathbf{B}_i$ is the i-th column vector of $\mathbf{B}$. Using (15), $\mathbf{B}$ and $\mathbf{B}_i\mathbf{B}_i^{\top}$ can be computed only once to be used for different $n, n \in \mathcal{K} \setminus \mathcal{K}_{g,user}$, which greatly reduces the computational cost. According to the rank-1 update of determinant [49], $\det(\mathbf{A}')$ can be written as

$$\det\left(\mathbf{A}'\right) = \det\left(\mathbf{A}\right)\left(1 + a_1\mathbf{B}_{\mathbf{1},\mathbf{1}}\right)\Bigg\{\mathbb{1}(|\Lambda| = 1) + \mathbb{1}(|\Lambda| > 1) \times \prod_{i=2}^{|\Lambda|}\left(1 + a_i\left[\mathbf{B} - \sum_{j=1}^{i-1}\frac{a_j\mathbf{B}_j\mathbf{B}_j^T}{1 + a_j\mathbf{B}_{j,j}}\right]_{i,i}\right)\Bigg\}. \tag{16}$$

$\left(\mathbf{B} - \sum_{j=1}^{i-1}\frac{a_j\mathbf{B}_j\mathbf{B}_j^T}{1 + a_j\mathbf{B}_{j,j}}\right)$ is already calculated in (15), which reduces the computational complexity. Substituting (15) and (16) into (14), we get the final results of $\det(\tilde{\mathcal{I}}\left(\Lambda \cup \{n, k\}\right))$.

The computation complexity of different algorithms. If we select keyframes in $\mathcal{K}_{loc}$ using a brute-force algorithm based on exhaustive enumeration of combinations of keyframes in $\mathcal{K}_{loc}$, the complexity is $\mathcal{O}\left(\binom{\rho}{l_{loc}} l_{loc}^3\right)$, where $\rho = |\mathcal{K} \setminus \mathcal{K}_{g,user}|$ is the number of keyframes that have not been offloaded to the edge server. Without computation reuse, the computation complexity of the proposed top-h greedy-based algorithm is $\mathcal{O}(H\rho l_{loc}^4)$. With computation reuse, it is reduced to $\mathcal{O}(Hl_{loc}^4) + \mathcal{O}(H\rho l_{loc}^3)$. Since we only keep l_{loc} keyframes in $\mathcal{K}_{loc}$ of the local map and a small H in Algorithm 2 to save computation resources, i.e., $\rho \gg l_{loc} > H$, the proposed greedy-based algorithm with computation reuse significantly reduces the computational complexity.

3) The Selection of Fixed Keyframe Set $\mathcal{K}_{fixed}$: After selecting the local keyframe set $\mathcal{K}_{loc}$ by solving Problem 3, we solve Problem 4 to select the fixed keyframe set.

Lemma 5. *Problem 4 is non-negative, monotone and submodular with a cardinality-fixed constraint.*

Proof. The proof is straightforward and hence omitted. □

Lemma 5 indicates that the problem can be approximately solved with greedy methods in Algorithm 1 [33]. For each iteration, the algorithm selects one keyframe from $\mathcal{K}_{g,user}$ to be added to the fixed keyframe set $\mathcal{K}_{fixed}$. The approximation ratio $\eta = 1 - \exp(-1)$ guarantees that worst-case performance of a greedy algorithm cannot be far from optimal.

B. Global Map Construction

We use a low-complexity algorithm to solve Problem 2 to construct the global map. The objective function of Problem 2 can be rewritten as $-\mathsf{Unc}\left(\mathcal{K}_{g,edge} \cup \mathcal{K}'\right)$, which has the same structure as that of Problem 3. Problems 2 and 3 both add keyframes to the existing keyframe sets to construct a pose graph and optimize the keyframe poses in the pose graph. Hence, Algorithms 2 and 3 can be used to solve Problem 2. In Algorithm 2, l_{loc} is replaced by $\frac{D}{d}$, and $\mathcal{K} \setminus \mathcal{K}_{g,user}$ is replaced by $\mathcal{K} \setminus \mathcal{K}_{g,edge}$. Calculating the uncertainty of a large global map is computationally intensive, and hence the proposed low-complexity algorithm is essential to reducing the computational load on the mobile device.

VIII. Evaluation

We implement AdaptSLAM on the open-source ORB-SLAM3 [19] framework which typically outperforms older SLAM methods [24], [25], with both V- and VI- configurations. The edge server modules are run on a Dell XPS 8930 desktop with Intel (R) Core (TM) i7-9700K CPU@3.6GHz and NVIDIA GTX 1080 GPU under Ubuntu 18.04LTS. In §VIII-A, the mobile device modules are run on the same desktop under simulated computation and network constraints. In §VIII-B, the mobile device modules are implemented on a laptop (with an AMD Ryzen 7 4800H CPU and an NVIDIA GTX 1660 Ti GPU), using a virtual machine with 4-core CPUs and 8GB of RAM. The weight w_e, $e = ((n,m),c)$ is set as the number of common map features visible in keyframes n and m for covisibility edges, similar to [19], [50], and the IMU edge weight is set as a large value (i.e., 500) as the existence of IMU measurements greatly reduces the tracking error. We empirically set $H = 5$ and $l_{thr} = 30$ in Algorithm 2 to ensure low complexity and good performance at the same time.

Metric. We use root mean square (RMS) absolute trajectory error (ATE) as the SLAM performance metric which is commonly used in the literature [19], [51]. ATE is the absolute distance between the estimated and ground truth trajectories.

Baseline methods. We compare AdaptSLAM with 5 baselines. **Random** selects the keyframe randomly. **DropOldest** drops the oldest keyframes when the number of keyframes is constrained. **ORBBuf**, proposed in [27], chooses the keyframes that maximize the minimal edge weight between the adjacent selected keyframes. **BruteForce** examines all the combinations of keyframes to search for the optimal one that minimizes the uncertainty (in Problems 1 and 2). BruteForce can achieve better SLAM performance than AdaptSLAM but is shown to have exponential computation complexity in §VII-A. In the **original ORB-SLAM3**, the local map includes all covisibility keyframes, and the global map includes all keyframes. The original ORB-SLAM3 also achieves better SLAM performance and consumes more computation resources than AdaptSLAM as the numbers of keyframes in both local and global maps are large.

Datasets. We evaluate AdaptSLAM on public SLAM datasets containing V and VI sequences, including TUM [47] and EuRoC [46]. The difficulty of a SLAM sequence depends on the extent of device mobility and scene illumination. We use EuRoC sequences V101 (easy), V102 (medium), and V103 (difficult), and difficult TUM VI room1 and room6 sequences. We report the results over 10 trials for each sequence.

A. Simulated Computation and Network Constraints

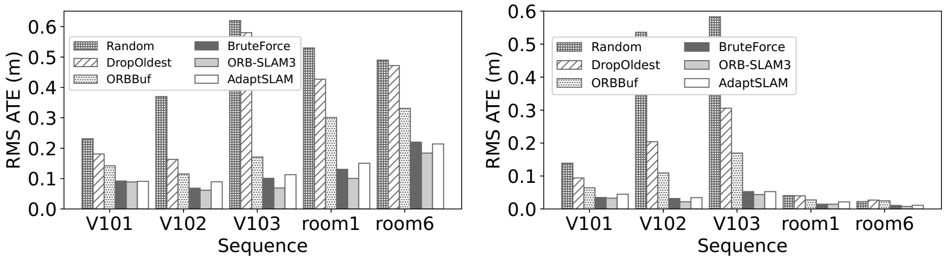
First, we limit the number of keyframes in the local map under computation constraints, and all keyframes are used to build the global map without communication constraints. Second, we maintain local maps as in the default settings of ORB-SLAM3, and limit the number of keyframes in the global map under constrained communications, where D in Problem 2 is set according to the available bandwidth.

Local map construction. We demonstrate the RMS ATE of different keyframe selection methods, for different V-SLAM (Fig. 2a) and VI-SLAM (Fig. 2b) sequences. The size of the local map is limited to 10 keyframes and 9 anchors in V-SLAM sequences, and 25 keyframes and 10 anchors in VI-SLAM sequences (to ensure successful tracking while keeping a small local map). AdaptSLAM reduces the RMS ATE compared with Random, DropOldest, and ORBBuf *by more than 70%, 62%, and 42%*, averaged over all sequences. The performance of AdaptSLAM is close to BruteForce, which demonstrates that our greedy-based algorithms yield near-optimal solutions, with substantially reduced computational complexity. Moreover, *the performance of AdaptSLAM is close to the original ORB-SLAM3* (less than 0.05 m RMS ATE difference for all sequences) *even though the size of the local map is reduced by more than 75%.*

The influence of the number l_{loc} of keyframes in the local map on the RMS ATE for different methods is shown in Fig. 3. We present the results for EuRoC V102 (of medium difficulty), which are representative. When l_{loc} is reduced from 30 to 10, AdaptSLAM increases the RMS ATE by only 6.7%, to 0.09 m, as compared to 0.37, 0.16, and 0.12 m for, correspondingly, Random, DropOldest, and ORBBuf. This indicates that AdaptSLAM achieves low tracking error under stringent computation resource constraints.

Global map construction. First, we examine the case where only half of all keyframes are offloaded to build a global map, for V-SLAM (Fig. 4a) and VI-SLAM (Fig. 4b) sequences. AdaptSLAM reduces the RMS ATE compared with the closest baseline ORBBuf *by 27% and 46% on average* for V- and VI-SLAM, and has small performance loss compared with the original ORB-SLAM3, despite reducing the number of keyframes by half.

Next, in Fig. 5, we examine four methods whose performance is impacted by the available bandwidth, under different levels of communication constraints. Without bandwidth limitations, all methods have the same performance as the global map holds all keyframes. When the bandwidth is limited, Random and DropOldest have the worst performance as they ignore the relations of keyframes in the pose graph. The ORBBuf performs better, but the tracking error is increased by 4.0× and 9.8× when the bandwidth is limited to 80 and 40 Mbps. AdaptSLAM achieves the best performance,



(a) V-SLAM (b) VI-SLAM

Fig. 2: RMS ATE for 6 keyframe selection methods in the local map construction for 5 sequences in EuRoC and TUM.

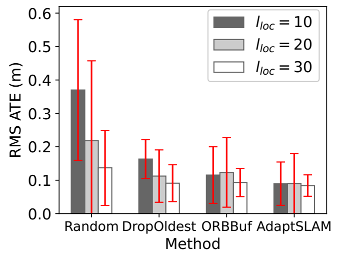
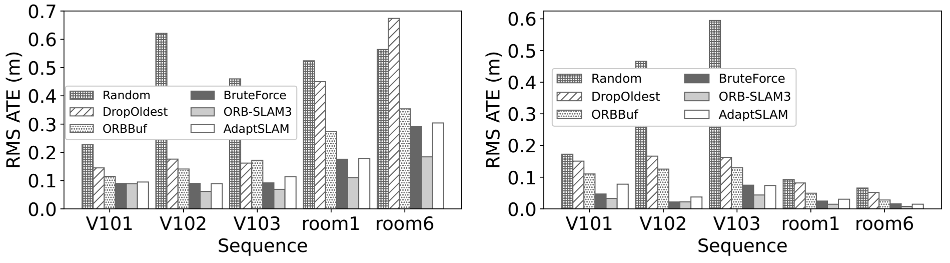


Fig. 3: RMS ATE for different sizes of local keyframe set (for EuRoC V102).



(a) V-SLAM (b) VI-SLAM

Fig. 4: RMS ATE for 6 keyframe selection methods in the global map construction for 5 sequences in EuRoC and TUM.

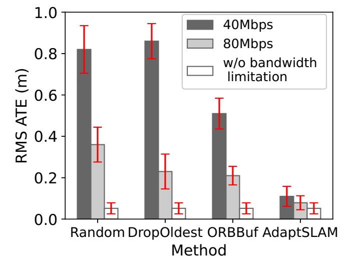


Fig. 5: RMS ATE for different available bandwidth for offloading keyframes (for EuRoC V102).

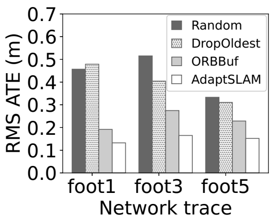


Fig. 6: RMS ATE for difference network traces.

Method	Latency (ms)
Random	133.0±86.3
DropOldest	139.9±53.7
ORBBuf	149.3±75.6
BruteForce	863.4±123.5
ORB-SLAM3	556.4±113.7
AdaptSLAM	162.8±68.9

TABLE I: The latency for local map construction and optimization.

reducing the RMS ATE compared to ORBBuf by 62% and 78% when network bandwidth is 80 and 40 Mbps, correspondingly. This highlights the superiority of AdaptSLAM in achieving high tracking accuracy under communication constraints.

B. Real-World Computation and Network Constraints

Following the approach of splitting modules between the edge server and the mobile device [10], we split the modules as shown in Fig. 1. The server and the device are connected via a network cable to minimize other factors. To ensure reproducibility, we replay the network traces collected from a 4G network [52]. Focusing on mobile devices carried by users, we choose network traces (foot1, foot3, and foot5) collected by pedestrians. We set D in Problem 2 according to the traces.

We examine the RMS ATE under the network traces in Fig. 6 for the EuRoC V102 sequence. The results for only four methods are presented because the overall time taken for running the SLAM modules onboard is high for BruteForce and the original SLAM. *AdaptSLAM reduces the RMS ATE by 65%, 61%, and 35%* (averaged over all traces) compared with Random, DropOldest, and ORBBuf. AdaptSLAM achieves high tracking accuracy under real-world network traces.

Table I shows the computation latency of mobile devices for all six methods. We compare the latency for running local map construction and optimization, which is the main source of latency for modules running onboard [10]. Compared with AdaptSLAM, the original ORB-SLAM3 takes 3.7× as much time for optimizing the local map as all covisibility keyframes are included in the local map without keyframe selection. Without the edge-assisted architecture, the original ORB-SLAM3 also runs global mapping and loop closing onboard which have even higher latency [10]. BruteForce takes 5.3× as much time for examining all the combinations of keyframes to minimize the local map uncertainty. The latency for constructing and optimizing local maps using AdaptSLAM is close to that using Random and DropOldest (<12.3% difference). Low latency for local mapping shows that edge-assisted SLAM is appealing, as local mapping is the biggest source of delay for modules executing onboard after offloading the intensive tasks (loop closing and global mapping).

IX. Conclusion

We present AdaptSLAM, an edge-assisted SLAM that efficiently select subsets of keyframes to build local and global maps, under constrained communication and computation resources. AdaptSLAM quantifies the pose estimate uncertainty of V- and VI-SLAM under the edge-assisted architecture, and minimizes the uncertainty by low-complexity algorithms. AdaptSLAM reduces the size of the local keyframe set by 75% compared with the original ORB-SLAM3 with a small performance loss. The authors have provided public access to their code at https://github.com/i3tyc/AdaptSLAM.

Acknowledgments

This work was supported in part by NSF grants CSR-1903136, CNS-1908051, and CNS-2112562, NSF CAREER Award IIS-2046072, by an IBM Faculty Award, and by the Australian Research Council under Grant DP200101627.

References

[1] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 215–242, 2021.

[2] 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 Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple micro aerial vehicles," in *Proc. IEEE/RSJ IROS*, 2013.

[4] R. Williams, B. Konev, and F. Coenen, "Scalable distributed collaborative tracking and mapping with micro aerial vehicles," in *Proc. IEEE/RSJ IROS*, 2015.

[5] Google. (2022) ARCore. https://developers.google.com/ar.

[6] Apple. (2022) ARKit. https://developer.apple.com/augmented-reality/arkit/.

[7] T. Scargill, G. Premsankar, J. Chen, and M. Gorlatova, "Here to stay: A quantitative comparison of virtual object stability in markerless mobile AR," in *Proc. IEEE/ACM Workshop on Cyber-Physical-Human System Design and Implementation*, 2022.

[8] Y.-J. Yeh and H.-Y. Lin, "3D reconstruction and visual SLAM of indoor scenes for augmented reality application," in *Proc. IEEE ICCA*, 2018.

[9] 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*, 2020.

[10] A. J. Ben Ali, Z. S. Hashemifar, and K. Dantu, "Edge-SLAM: Edge-assisted visual simultaneous localization and mapping," in *Proc. ACM MobiSys*, 2020.

[11] 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. Embed. Comput. Syst.*, vol. 22, no. 1, pp. 1–31, 2022.

[12] I. Deutsch, M. Liu, and R. Siegwart, "A framework for multi-robot pose graph SLAM," in *Proc. IEEE RCAR*, 2016.

[13] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM—collaborative visual-inertial SLAM," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2762–2769, 2018.

[14] F. Li, S. Yang, X. Yi, and X. Yang, "CORB-SLAM: a collaborative visual SLAM system for multiple robots," in *CollaborateCom*. Springer, 2017.

[15] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *J. Field Robot.*, vol. 36, no. 4, pp. 763–781, 2019.

[16] K.-L. Wright, A. Sivakumar, P. Steenkiste, B. Yu, and F. Bai, "Cloud-SLAM: Edge offloading of stateful vehicular applications," in *Proc. IEEE/ACM SEC*, 2020.

[17] 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*, 2022.

[18] 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, 2017.

[19] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, 2021.

[20] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, 2018.

[21] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for greedy maximization of non-submodular functions with applications," in *Proc. PMLR ICML*, 2017.

[22] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Springer ECCV*, 2014.

[23] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, 2017.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE ISMAR*, 2007.

[25] E. Dong, J. Xu, C. Wu, Y. Liu, and Z. Yang, "Pair-Navi: Peer-to-peer indoor navigation with mobile visual SLAM," in *Proc. IEEE INFOCOM*, 2019.

[26] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *Proc. IEEE ICRA*, 2012.

[27] Y.-P. Wang, Z.-X. Zou, C. Wang, Y.-J. Dong, L. Qiao, and D. Manocha, "ORBBuf: A robust buffering method for remote visual SLAM," in *Proc. IEEE/RSJ IROS*, 2021.

[28] L. Riazuelo, J. Civera, and J. M. Montiel, "C^2TAM: A cloud framework for cooperative tracking and mapping," *Robot. Auton. Syst.*, vol. 62, no. 4, pp. 401–413, 2014.

[29] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multi-robot simultaneous localization and mapping," *IEEE Internet Things J.*, 2022.

[30] K. Khosoussi, M. Giamou, G. S. Sukhatme, S. Huang, G. Dissanayake, and J. P. How, "Reliable graphs for SLAM," *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 260–298, 2019.

[31] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 1–20, 2018.

[32] Y. Chen, L. Zhao, Y. Zhang, S. Huang, and G. Dissanayake, "Anchor selection for SLAM based on graph topology and submodular optimization," *IEEE Trans. Robot.*, 2021.

[33] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[34] A. Das and D. Kempe, "Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 74–107, 2018.

[35] L. Carlone, G. C. Calafiore, C. Tommolillo, and F. Dellaert, "Planar pose graph optimization: Duality, optimal solutions, and verification," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 545–565, 2016.

[36] J. A. Placed and J. A. Castellanos, "Fast autonomous robotic exploration using the underlying graph structure," in *Proc. IEEE/RSJ IROS*, 2021.

[37] K. Khosoussi, S. Huang, and G. Dissanayake, "Tree-connectivity: Evaluating the graphical structure of SLAM," in *Proc. IEEE ICRA*, 2016.

[38] Y. Chen, S. Huang, L. Zhao, and G. Dissanayake, "Cramér–Rao bounds and optimal design metrics for pose-graph SLAM," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 627–641, 2021.

[39] N. Boumal, A. Singer, P.-A. Absil, and V. D. Blondel, "Cramér–Rao bounds for synchronization of rotations," *Information and Inference: A Journal of the IMA*, vol. 3, no. 1, pp. 1–39, 2014.

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

[41] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[42] M. L. Rodríguez-Arévalo, J. Neira, and J. A. Castellanos, "On the importance of uncertainty representation in active SLAM," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 829–834, 2018.

[43] F. Pukelsheim, *Optimal design of experiments*. SIAM, 2006.

[44] Y. Chen, H. Inaltekin, and M. Gorlatova, "AdaptSLAM: Edge-assisted adaptive SLAM with resource constraints via uncertainty minimization," Tech. Rep., 2023. [Online]. Available: https://arxiv.org/abs/2301.04620

[45] K. Khosoussi, S. Huang, and G. Dissanayake, "Novel insights into the impact of graph structure on SLAM," in *IEEE/RSJ IROS*, 2014.

[46] 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. Rob. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[47] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The TUM VI benchmark for evaluating visual-inertial odometry," in *Proc. IEEE/RSJ IROS*, 2018.

[48] G. Strang, *Linear algebra and its applications*. Thomson, Brooks/Cole, 2006.

[49] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[50] Y. Chen, L. Zhao, K. M. B. Lee, C. Yoo, S. Huang, and R. Fitch, "Broadcast your weaknesses: Cooperative active pose-graph SLAM for multiple robots," *IEEE Robot. Autom. Lett*, vol. 5, no. 2, pp. 2200–2207, 2020.

[51] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *Proc. IEEE/RSJ IROS*, 2018.

[52] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, 2016.