Dense Decentralized Multi-robot SLAM based on locally consistent TSDF submaps

Rodolphe Dubois, Alexandre Eudes, Julien Moras, Vincent Frémont

Dense Decentralized Multi-robot SLAM based on locally consistent *TSDF* submaps

Rodolphe Dubois[1,2], Alexandre Eudes[1], Julien Moras[1], Vincent Frémont[2]
[1] DTIS, ONERA, Université Paris Saclay, F-91123 Palaiseau, France
[2] Centrale Nantes, LS2N, UMR 6004, Nantes, France
E-mail: [1] firstname.lastname@onera.fr, [2] firstname.lastname@ls2n.fr

***Abstract*— This article introduces a decentralized multi-robot algorithm for Simultaneous Localization And Mapping (SLAM) inspired from previous work on collaborative mapping [1]. This method makes robots jointly build and exchange i) a collection of 3D dense locally consistent submaps, based on a Truncated Signed Distance Field (TSDF) representation of the environment, and ii) a pose-graph representation which encodes the relative pose constraints between the TSDF submaps and the trajectory keyframes, derived from the odometry, inter-robot observations and loop closures. Such loop closures are spotted by aligning and fusing the TSDF submaps. The performances of this method have been evaluated on multi-robot scenarios built from the EuRoC dataset [2].**

I. Introduction

In mobile robotics, many tasks such as exploration, inspection or navigation in cluttered areas rely on a real-time 3D mapping of the environment. Such 3D models can be built incrementally using Simultaneous Localization And Mapping (SLAM) techniques which jointly estimate the map and the trajectory of the robot, and refine them by spotting loop closures within the trajectory based on the reconstructed map. A wide variety of visual and LiDAR-based SLAM algorithms have been introduced, with different output representations for the mapping such as point clouds, occupancy grids and meshes (see [3]). In this work, we focus on stereo-inertial SLAM to build a TSDF-based (Truncated Signed Distance Field [4]) dense mapping of the environment, which is suitable for navigation and collision avoidance.

While several single-robot SLAM algorithms of that kind already exist (see section III), one current challenge is to extend such algorithms into a multi-robot framework. It combines three key ingredients: i) a task and data allocation scheme, ii) an adapted communication policy and iii) a matching & merging strategy. The allocation scheme governs how each task and data is either centralized, decentralized or distributed. The communication policy supervises the topology, planning and content of inter-robot data exchanges. Finally, the matching & merging strategy handles the fusing of the received data and its propagation to refine the map and trajectory estimates.

In a decentralized multi-robot framework, several robots exchange information in a peer-to-peer fashion to mutually refine their knowledge of the trajectories and the environment. Such architectures bring three main advantages: i) it enhances the efficiency of the fleet by allowing a faster coverage of the explored area; ii) it makes the fleet more flexible by granting the autonomy of the agents compared to centralized architectures, and iii) it allows each agent to improve its estimation accuracy by fusing the shared information. However, decentralized SLAM specifically introduces three kinds of challenges. The first challenges stem from communication issues; they involve handling contact losses and recoveries between the robots due to limited communication range, and scale the quantity of exchanged data to the available bandwidth. Besides, the clocks of the robots should be synchronized precisely enough to properly process inter-robot observations. The second kind of challenges regard the processing of real-time multi-robot interactions. One must ensure that sending information to the other robots and integrating packets received from them do not excessively monopolize the limited computational resources of the agents. Finally, as a third challenge, robots should properly integrate the data received from other robots, which assumes that they are able to estimate the odometry reference frames of the other agents, and that they avoid double-counting issues to ensure consistent re-estimations.

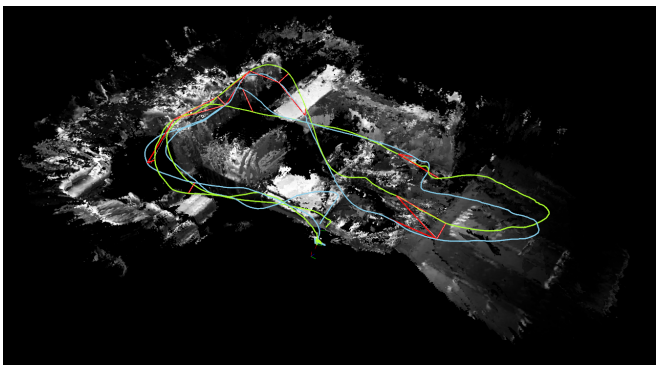

Fig. 1: Screenshot of reconstructed map and mesh for robot 4 in scenario 3 (see section X). Submap matches are shown in red.

II. Contributions and Paper Organization

To the best of our knowledge, Duhautbout et al. [1] introduced the first decentralized collaborative mapping framework relying on the exchange of TSDF submaps. As a main contribution, we extend their work into a fully decentralized multi-robot SLAM pipeline through punctual improvements to the Back-End modules. More specifically, we make the Back-End inference more flexible by adopting a factor-graph formulation as a unified framework to integrate odom-

etry, direct multi-robot observation and submap matches contraints instead of merely propagating cumulated match corrections to subsequent submaps as done in [1]. We also robustify the TSDF submap matching module to avoid or mitigate spurious matches. We then propose an adjusted communication policy to handle decentralization constraints. Finally, the proposed method does not assume common initial odometry reference frames. The article is organized as follows. Section III recaps some previous works on dense collaborative mapping and decentralized multi-robot SLAM. Section IV details some notations used throughout the paper. Section V gives an overview of the proposed method. Its algorithmic blocks such as its Front-End, its global inference, submap matching and submap exchange modules are respectively detailed in sections VI, VII, VIII and IX. Experimental results are discussed in section X.

III. Related works

A. Dense mapping

Dense mapping methods provide high resolution 3D models of the observed obstacle surfaces. They contrast to sparse mapping methods, commonly used in visual SLAM, and which use sparse point clouds as a support to formulate geometric errors. Sketching only the raw geometry of the environment in an unstructured way, they are neither suitable for visual rendering nor for path planning.

A first family of methods use dense unstructured clouds of geometric primitives such as points or surfels [5], eventually augmented with photometric information. Dense point clouds may result from LiDAR scans [6], the triangulation from stereo image pairs [7] or the projection of localized depth maps [8]. However, such representations remain unsuitable for collision-avoidance tasks due to their unstructured nature.

A second family of methods use occupancy grids to explicitly represent the surfaces. For instance, *Octomap* [9] defines a 3D occupancy grid of voxels with a given resolution and internally structured by an Octree model. In this representation, each voxel is either occupied, free or unknown, and the obstacle surface is explicitly defined as the collection of all occupied voxels. However, space discretization intrinsically limits the quality of the visual rendering, and the grid needs to be augmented with a distance map [10] to be used for path-planning.

As a third family, 3D polygonal meshes provide an alternative explicit representation of surfaces. They can be triangulated from sparse point clouds [11], but such process makes them hard to update. One alternative solution is to generate meshes from implicit representations like Signed Distance Fields (SDF) [4] which model surfaces as their zero-level sets. Driven by the increasing embarked computational capabilities and the development of new RGB-D sensors (e.g. Intel Realsense, Kinect), implicit surface representations eventually outperformed explicit methods in terms of mapping accuracy and computational efficiency, and turned out to be more adequate for path-planning issues.

B. Dense single-robot SLAM

To be used into a SLAM framework, a dense map representation should ideally meet three requirements: i) allow incremental building ; ii) provide an underlying model for pose estimation for odometry and loop closure and iii) be malleable enough to maintain local and global consistency.

Dense SLAM has first relied on dense point clouds for LiDAR [6], RGB-D [8] and stereo [7] SLAM. Brand et al. [7] proposed a submap-based approach which allows to maintain global consistency by aligning the locally consistent submaps via ICP point-cloud registration. Working with dense surfel clouds, *ElasticFusion* [5] introduces a model-to-model surface loop closure to ensure local consistency through non-rigid surface deformation, while it maintains global consistency using global loop closures.

SDF-based mapping was popularized by the *Kinect Fusion* algorithm [12]. It builds on RGB-D information to compute a Truncated SDF (TSDF) in a spatial grid. Further developments have been made on this idea, like the *Voxblox* [13], dedicated to robotic navigation. Finally, *C-Blox* [14] replaced the monolithic map by a set of locally consistent TSDF submaps, whose localization was refined using a co-constructed feature-based map, and allowed the fusion of redundant overlapping submaps.

C. Decentralized Dense multi-robot SLAM

Dub et al. [15] proposed *SegMap* which decomposes dense LiDAR point clouds into 3d segments identified and matched using learnt descriptors. Such descriptors are exchanged between the robots and used for dense reconstruction. Schuster et al. [16] proposed a stereo-visual decentralized SLAM framework. It makes each robot share some local dense point-cloud submaps with its neighbours, which are then aligned via ICP point-cloud registration. In this approach, the odometry reference frames of the robots are first aligned based on inter-robot observations. The framework proposed in this paper shows many similarities with [16] as it combines dense submapping, submap matching and pose-graph optimization.

A decentralized mapping framework based on the TSDF submaps was first introduced by the work of Duhautbout et al. [1]. The authors proposed to correct the mapping of each robot by integrating the submaps broadcast by the other robots and matched using ICP algorithms over point clouds extracted from the reconstructed surface polygonal meshes. However, only the mapping part is refined by those a posteriori corrections, while the trajectory and previous submaps are not impacted by the spotted matches.

IV. Notations

In the next sections, $\mathbb{SE}_3$ and $\mathbb{SO}_3$ respectively denote the *Special Euclidean Group* and the *Special Orthogonal Group* of dimension 3. We parameterize $\mathbb{SE}_3$ using the composite Lie group $\mathbb{SO}_3 \times \mathbb{R}^3$. The operator $\oplus$ denotes the $\mathbb{SE}_3$ pose

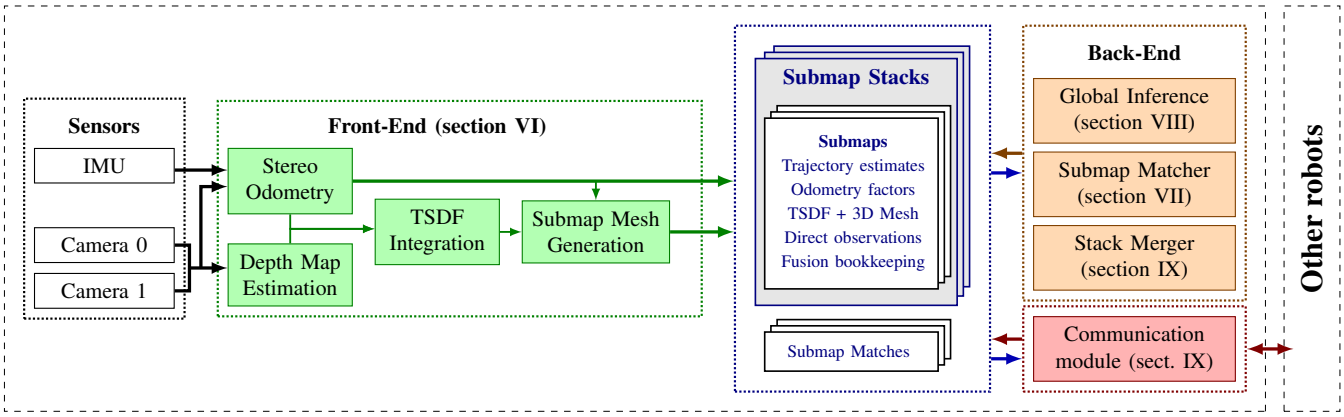

Fig. 2: System overview: The map is initialized by the Front-End module which estimates the trajectory, builds the TSDF submaps and computes the associated meshes. The Back-End modules then refine the map by spotting submap matches and re-estimating the trajectory. Finally, the communication module sends the completed submaps, the spotted matches and the fusion bookkeeping to the other robots, and integrates received submaps.

product. We define the $\boxminus$ operator as follows:

$$(\forall\ T_1, T_2 \in \mathbb{SE}_3) \quad T_2 \boxminus T_1 = \begin{bmatrix} \log_{\mathbb{SO}_3}(\boldsymbol{R}_2 \cdot \boldsymbol{R}_1^\top)^\vee \\ \boldsymbol{t}_2 - \boldsymbol{t}_1 \end{bmatrix} \quad (1)$$

where $\boldsymbol{R}_{(\cdot)} \in \mathbb{SO}_3$ and $\boldsymbol{t}_{(\cdot)} \in \mathbb{R}^3$ parameterize $T_{(\cdot)} \in \mathbb{SE}_3$, $\log_{\mathbb{SO}_3}$ maps from $\mathbb{SO}_3$ to its Lie algebra $\mathfrak{so}_3$ and the *vee* operator maps from $\mathfrak{so}_3$ to $\mathbb{R}^3$. Interested readers may refer to [17] for more details on operations on Lie groups. Finally, given $n \in \mathbb{N}$, let $\boldsymbol{A} \in \mathcal{M}_n(\mathbb{R})$ be a positive definite symmetric matrix and $\boldsymbol{x} \in \mathbb{R}^n$, the $\boldsymbol{A}$-weighted Mahalanobis norm of $\boldsymbol{x}$ is defined as follows: $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^\top \cdot \boldsymbol{A}^{-1} \cdot \boldsymbol{x}}$.

V. System overview

The Figure 2 gives an overview of the proposed method. The Front-End module is very similar to one used in [1]. We initialize the trajectory estimates using the eVO algorithm [18], but any other stereo odometry algorithm may be used instead. In parallel, the trajectory is splitted on-the-fly into a succession of submaps based on the estimated travelled distance and change in orientation. While the current submap is still active, the TSDF representation of the observed environment is updated using the dense depth maps computed by the *Efficient LArge Scale stereo-matching* (ELAS) algorithm [19]. Once this submap is closed, we use the *Marching Cube* algorithm [20] to convert its TSDF into a polygonal mesh which represents the surface of the observed obstacles. Those processes allow to fill in the map which consists of the trajectory estimates and the submaps, each submap including a TSDF and the associated polygonal mesh.

The Back-End modules aim at ensuring the global consistency of the map by spotting loop closures between the submaps and re-estimating the trajectory. As detailed in section VII and similarly to [1], each newly computed submap is matched against its most overlapping neighbour (excluding consecutive submaps) using the *Iterative Closest Point* registration. While in [1], submap matchings are only used to correct the pose of the current and posterior submaps, we include it as an additional trajectory constraint in a factor graph that is maintained in parallel. This factor graph may also include factors derived from inter-robot observations such as direct observations between the robots (see [21]) or the observation of common landmarks. We can then refine the whole trajectory by performing a 6-DoF optimization over the factor graph.

Finally, the communication module handles the multi-robot interactions. As in [1], submaps are the elementary exchange units between the robots and each newly completed submap is immediately broadcast to its neighbours. In the proposed method, we exchange additional information regarding inter-robot observations, submap matches and patch fusions, and we propose to manage inter-robot interactions (including contact losses and recoveries) based on a communication history (see section IX).

VI. Front-End and patch generation

A. Submap creation

The submap creation procedure is mostly similar to [1]. The trajectory is splitted online into a succession of submaps, relying on eVO pose estimates[1]. It defines the underlying estimation model by encoding the relative pose stochastic constraints derived from the eVO outputs. Each time a new keyframe pose estimate is received from eVO, it is either added to the current submap or a new submap is created if the accumulated distance or the accumulated changes in orientation in that submap exceeds some user-predefined thresholds, $l_{\max}$ and $\theta_{\max}$ respectively. If a new submap has to be created, its TSDF is queued to be converted into a polygonal mesh representing the surface of the observed obstacles. Note that each submap defines its own frame of reference with coincides with the pose of its first keyframe, whose covariance matrix is initialized by propagating the uncertainties along the odometry factors.

B. Submap surface computation

In parallel to the odometry, ELAS [19] estimates depth maps from the stereo image pairs. We associated each newly estimated depth map to its corresponding eVO pose estimate. Both are integrated to build a SDF representation attached

[1]We use a slightly modified version of eVO [18] which additionally returns the covariance matrix of the relative pose from the previous keyframe to the current one $\boldsymbol{\Sigma}_{\mathrm{KF_{k-1}},\mathrm{KF_k}}$.

to the current submap. A SDF Φ implicitely defines the surface $\mathcal{S}$ as its zero-level set i.e. $\mathcal{S} = \Phi^{-1}(\{0\})$. The TSDF builds an incremental approximation of the SDF from range measurements. It first relies on spatial voxellization. We use the open-source *OpenChisel* library [22] which organizes the voxels into dynamically allocated chunks and allows constant-time access using a spatially-hashed index. Secondly, it requires distance estimation. Given a range measurement, we measure for each observed voxel $\boldsymbol{v}$ the projective distance $d_m(\boldsymbol{v})$ to the measured impact point along the observation ray. By convention, the distance is signed positively when the voxel lies between the sensor and the surface, and negatively when it is behind the surface. It is truncated at a given threshold $d_{\max}$ as it approximates the Euclidean distance only within the surface's neighborhood. *OpenChisel* scales this threshold dynamically w.r.t. the depth uncertainty. To account for measurement uncertainty, each voxel is assigned a weight $\hat{w}(\boldsymbol{v})$ which reflects the estimate's reliability. Each new range measurement $d_m(\boldsymbol{v})$ of weight $w_m(\boldsymbol{v})$ is fused with the current distance estimate $\hat{d}(\boldsymbol{v})$ of weight $\hat{w}(\boldsymbol{v})$ using a running weighted average:

$$\begin{cases} \hat{d}(\boldsymbol{v}) & \leftarrow \dfrac{\hat{w}(\boldsymbol{v}) \cdot \hat{d}(\boldsymbol{v}) + w_m(\boldsymbol{v}) \cdot d_m(\boldsymbol{v})}{\hat{w}(\boldsymbol{v}) + w_m(\boldsymbol{v})} \\ \hat{w}(\boldsymbol{v}) & \leftarrow \hat{w}(\boldsymbol{v}) + w_m(\boldsymbol{v}) \end{cases} \tag{2}$$

Such incremental refinement ensures the local consistency of the TSDF. Furthermore, observed voxels located close to the range sensor and whose distance sign is inconsistent with the current measurement are reset using *space carving* [22]. Finally, we use the *Marching Cube* algorithm [20] to generate explicit representations of the surface as the polygonal mesh which best fits the TSDF zero-level set. Each TSDF is built in the reference frame of its associated submap, and its polygonal mesh is computed as soon as the submap is closed. Once computed and as in [16], it is queued for submap matching and stored in a dedicated stack, where it is identified by two IDs: a robot ID – of the robot which computed it – and a patch ID.

VII. ICP-BASED SUBMAP MATCHING

As soon as the mesh of a submap $\mathcal{S}_i$ is computed, we try to match it with previously computed submap meshes according to the following procedure inspired from [1]. First, we compute the Axis-Aligned Bounding Box (AABB) AABB_i that encompasses its mesh; this rough volume estimation allows fast pairwise comparisons for overlapping. We then select the submap $\mathcal{S}_j$ whose AABB_j shows the highest overlap, skipping consecutive or already-matched submaps. We proceed if $v_{ij} = \text{volume}(\text{AABB}_i \cap \text{AABB}_j) \geq v_{\min}$ where $v_{\min}$ is a user-defined threshold. We restrict the matching procedure to most overlapping submap for real-time requirements. We then perform a model-to-model alignment [23] based on the *Iterative Closest Point* (ICP) algorithm [24] to estimate the relative pose $T_{\mathcal{S}_i\mathcal{S}_j} \in \mathbb{SE}_3$ between both meshes and thus between their associated submaps. For that purpose, we first extract the nodes of the submap meshes within the identified common chunks as point clouds $\mathcal{P}_i$ and $\mathcal{P}_j$, along with their normals. The ICP is then run using a point-to-plane criterion (which widens the convergence bassin) within a RANSAC scheme, implemented in the *Point Cloud Library* (PCL) [25].

Algorithm 1– Submap alignment procedure (inspired from [1])

Candidate submap to align: $\mathcal{S}_i$;
Find the most overlapping submap $\mathcal{S}_j$ such that $v_{ij} = \text{volume}(\text{AABB}_i \cap \text{AABB}_j) \geq v_{\min}$;
Extract the points clouds $\mathcal{P}_i$ and $\mathcal{P}_j$ within the identified overlapping areas;
Compute the relative pose transformation $\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j}$ using ICP;
if *ICP converges* **then**
  Check the numbers of inlier correspondences exceeds $\min_{\text{inliers}}$;
  Check the ICP RMSE is below $\max_{\text{RMSE}}$;
  Check the mean angle between corresponding normals is below $\max_{\angle}$;
  Check that $\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j}$ is consistent with the estimated uncertainties on $T_{\text{W}\mathcal{S}_i}$ and $T_{\text{W}\mathcal{S}_i}$;
  Estimate the ICP covariance matrix $\boldsymbol{\Sigma}_{\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j}}$;
  Add the submap match factor between $\mathcal{S}_i$ and $\mathcal{S}_j$;
  if $v_{ij} \geq v^{\textit{fusion}}_{\min}$ **then**
    Fuse TSDF_i into TSDF_j;

Once the ICP has run, we consider the match as successful it passes the four following tests. First, the estimated pose $\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j} \in \mathbb{SE}_3$ should be supported by at least $\min_{\text{inliers}}$ inlier correspondances. Secondly, the Root Mean Squared Error (RMSE) RMSE_{ICP} computed over the inlier correspondences must not exceed $\max_{\text{RMSE}}$, nor should the mean angle between corresponding normals exceed $\max_{\angle}$. Thresholds $\min_{\text{inliers}}$, $\max_{\text{RMSE}}$ and $\max_{\angle}$ are user-defined. Finally and overall, $\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j}$ should be consistent with the estimated covariance matrix $\hat{\boldsymbol{\Sigma}}_{\text{T}_{\mathcal{S}_i\mathcal{S}_j}}$:

$$\left\| \left(T^{-1}_{\text{WS}_i} \oplus T_{\text{WS}_j}\right) \boxminus \hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j} \right\|^2_{\hat{\boldsymbol{\Sigma}}_{\text{T}_{\mathcal{S}_i\mathcal{S}_j}}} \leq \chi^2_{6;95\%} \tag{3}$$

where W denotes the world frame, T_{WS_i} and T_{WS_j} are the current submap poses extracted from the map, and $\chi^2_{6;95\%}$ is the 95% fractile of a χ^2 distribution with 6 degrees of freedom. $\hat{\boldsymbol{\Sigma}}_{\text{T}_{\mathcal{S}_i\mathcal{S}_j}}$ is computed from $\hat{\boldsymbol{\Sigma}}_{\text{T}_{\text{W}\mathcal{S}_i}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{T}_{\text{W}\mathcal{S}_j}}$ using the classical error propagation law. If the alignment succeeds, the covariance of the ICP-estimated relative pose has to be computed in order to include the submap match as a new factor in the underlying pose graph. Inspired by [7], we approximate it within its actual convergence bassin as a spherical matrix whose standard deviations depend on RMSE_{ICP}

$$\sigma_{\{\text{x,y,z}\}} = \max\left(\text{RMSE}_{\text{ICP}}, \sigma^{\min}_t\right) \tag{4a}$$

$$\sigma_{\{\text{roll,pitch,yaw}\}} = \max\left(\arctan\left(\frac{2 \cdot \text{RMSE}_{\text{ICP}}}{d_{\{\text{yz,xz,xy}\}}}\right), \sigma^{\min}_\theta\right) \tag{4b}$$

where $d_{\{\text{yz,xz,xy}\}}$ denotes the diameters of the AABB overlap between the matched submaps in the respective planes and $\sigma^{\min}_t$ and $\sigma^{\min}_\theta$ are user-defined threshold introduced to avoid overconfident estimation. Finally, if the overlap v_{ij} between both submaps exceeds a predefined threshold $v^{\text{fusion}}_{\min} > v_{\min}$, then the query submap $\mathcal{S}_i$ is fused into the matched submap

$\mathcal{S}_j$. This requires to project the voxels of TSDF$_i$ into TSDF$_j$ using $\hat{T}^{\text{ICP}}_{\mathcal{S}_i\mathcal{S}_j}$ to spot the impacted voxels of TSDF$_j$. Then, the values of TSDF$_i$ at the center of impacted voxels of TSDF$_j$ are computed via trilinear interpolation, and integrated using equation (2). The mesh of the fused submap is then deleted from memory and replaced by a reference to the fused one for further matches. We require each submap to keep tracks of such fusions as the set of the submap IDs whose mesh has been fused into it. The whole process is summarized in algorithm 1.

VIII. Factor Graph and Back-End inference

Contrary to the work of [1] which only used matches to correct the drift of posterior submaps, we use an underlying factor graph to refine the whole trajectory and the mapping. The structure of the maintained factor graph, similar to [16] for the sequential case, is represented in the Figure 3.

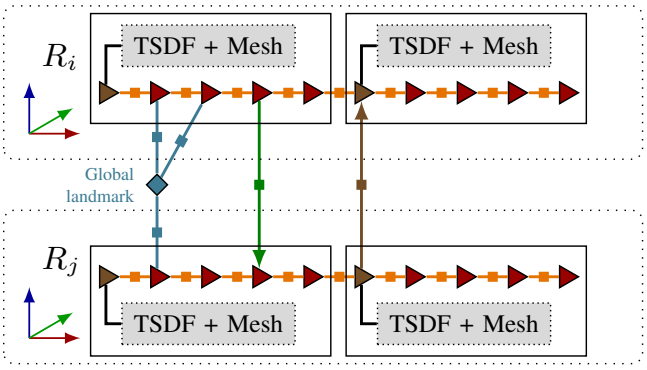

Fig. 3: Structure of the pose-graph with relative pose odometry factors (orange), submap matching factors (brown), inter-robot observation factors (green) and global landmark observation factor (blue).

In this graph, each factor between two keyframe pose variables $T_\text{i}, T_\text{j} \in \mathbb{SE}_3$ encodes a stochastic constraint:

$$\boldsymbol{\xi}_{\text{ij}} \triangleq \left(T_\text{i}^{-1} \oplus T_\text{j}\right) \boxminus \hat{T}_{\text{ij}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\text{ij}}) \tag{5}$$

where $\mathcal{N}$ denotes the Gaussian distribution and $\boldsymbol{\Sigma}_{\text{ij}}$ is the covariance matrix of the residual expressed in $\mathfrak{m}$. Thus, the likelihood of $\hat{T}_{\text{ij}}$ w.r.t. T_i and T_j is:

$$p(\hat{T}_{\text{ij}}|T_\text{i}, T_\text{j}) \propto \exp\left(-\frac{1}{2}\left\|\boldsymbol{\xi}_{\text{ij}}\right\|^2_{\boldsymbol{\Sigma}_{\text{ij}}}\right) \tag{6}$$

Batch inference on the full pose graph is performed by minimizing the negative log-likelihood of the relative pose estimates w.r.t. to the pose estimates:

$$\Theta^* = \arg\min_{\Theta} \{-\log p(\mathcal{Z}|\Theta)\} \tag{7}$$

where Θ is the set of all optimized pose variables, and $\mathcal{Z}$ the set of all relative pose *measurements* (may they be direct inter-robot observations or spotted submap matches). $p(\mathcal{Z}|\Theta)$ is proportional to the product of the factors defined in equation (6). The problem (7) yields a nonlinear least-square optimization problem, solved using the Levenberg-Marquardt algorithm. Similarly to [16], we also use a Cauchy loss function with parameter $\chi^2_{6;95\%}$ on the loop closure factors to mitigate eventual spurious matches.

In the proposed framework, the pose graph is built incrementally using the statistically independant pose and covariance estimates provided by eVO. Batch optimization is performed in parallel for each new submap match, using the *Ceres* solver [26] which then allows to extract the submap pose covariance matrices.

IX. Multi-Robot Framework

A. Task and data allocation scheme

The proposed multi-robot framework is fully decentralized. Each robot independently estimates its own instance of the map, as it holds valuable information for higher-level path-planning tasks. Besides, each robot separately runs its own matching and inference modules. However, as discussed below, the task of spotting of inter-robot matches is splitted among the robots as each robot only looks for inter-robot matches between received submaps and its own, and found matches are then shared between the robots.

B. Communication policies

In a decentralized framework with limited communication range, robots may occasionally loose contact between each other when covering large areas, and thus miss some valuable information broadcast in the meanwhile. Therefore, the data exchange policy must cover i) regular communications for robots within communication range, and ii) a regularization policy to handle contact recovery.

1) Regular communication policy: In the work of Duhautbout et al. [1], the submaps, which includes the TSDF and its reference pose, were the elementary exchange unit between the robots. Similarly, we make the robots broadcast each submap they complete to all their reachable neighbours. Each submap message includes the submap's ID, its associated TSDF, its fusion bookkeeping, its associated subgraph of poses as well as the relative pose factors derived from direct inter-robot observations. The polygonal mesh is not communicated and must be re-computed by the receiver. We also make robots communicate the submap matches they found, and wheter such match triggered a submap fusion or not. Received matches and fusions are queued and integrated to the map as soon as possible.

2) Regularization policy: When covering large areas, robots may occasionally loose contact between each other. When they meet again and recover communication, they should regularize their knowledge. For that purpose, we make each robot hold a reception bookkeeping. Each robot i stores the following information for each other robot $j \neq i$: i) the timestamp of the most recently received submap of robot j, ii) the timestamp of the most recently received submap match found by robot j. In the decentralized framework, some submaps and matches from robot j may also have been relayed to robot i by third robot. When two robots meet again after a loss of contact, they first exchange their current reception bookkeeping. They can then regularize their knowledge by sending to the other robot all the submaps and all the matches which are posterior to most recently received ones. The reception of submaps and matches must

be acknowledged by the receiver robot to pursue. For each robot, information must be sent in chronological order not to break the consistency of the reception history if the communication was to be lost again during the process.

C. Matching and Merging strategy

1) Stack Merging: When a robot receives a submap from another robot, it integrates it into its associated submap stack. In [1], received patches were integrated into a public patch manager and eventually used to correct the robot's own patches stored into a distinct private patch manager. In the proposed method, if the relative pose transformations between its odometry reference frame and the one of the host robot is unknown, we initialize one submap stack per robot. Thus, each submap stack first stands as a separate map with its own reference frame. During the process, those submap stacks, which evolve independently from each other, are to be fused as the transformations between their odometry reference frames get estimated. Received inter-robot correspondences are either added or buffered, while we check if previously buffered correspondences can be added. Each time a new submap is received, the Stack Merger Module (see Figure 2) checks whether there are enough inter-robot correspondences to estimate the relative transformation between the reference frames of the two involved mapping stacks. If it does, we select the most consensual relative pose transformation hypothesis induced by the inter-robot correspondences and fuse both stacks. All submaps of the aligned stack are from now on considered as *anchored*.

2) Inter-robot submap matching: For each new submap $\mathcal{S}_i$ created by the host robot, the process described in algorithm 1 is performed for each *anchored* trajectory. For each new submap, the robot tries to match it against its own most overlapping submap and against the received anchored most overlapping submap. Each robot only looks for submap matches involving its own submaps, and may benefit from the matches received from other robot between the submaps of other robots. If the robot receives a fusion information, it fuses the two involved submaps and updates their fusion bookkeeping.

3) Multi-robot global inference: Figure 3 shows the structure of the underlying pose graph to handle multi-robot global inference. A bach pose graph optimization is required each time a new submap match is found or received, and as soon as a mapping stack gets anchored. However, the optimization is performed only over the anchored trajectories.

X. Performance evaluation

A. Considered test scenarios

The proposed algorithm has been evaluated on multiple multi-robot scenarios. We built those scenarios using sequences taken from the EuRoC dataset [2] (see Figure 4). We built multi-robot scenarios by synchronizing the individual sequences. Table I recaps the main characteristics for each scenario, as well as the performance metrics of eVO [18] on each sequence (see §X-B).

We simulated noisy and limited range relative pose inter-robot based on the detection of markers carried by the robots (see [21]). Given a camera C and a marker T, we sampled a relative pose measurement only if i) the T fully projected onto the image plane of C, and ii) the relative distance between C and T and the tilt observation angle where respectively below $d_{\text{CT}}^{\max} = 5\text{m}$ and $\theta_{CT}^{\max} = 60°$. Relative measurement were sampled using Gaussian noises with standard deviations $\sigma_t = 0.1\text{m}$ and $\sigma_\theta = 5°$ for translation and orientation. However, Gaussian noise is a simplifying simulation assumption, while real marker measurements exhibit biases and ambiguity effects which can cause large outliers at certain view angles and larger distances, as analyzed by [27].

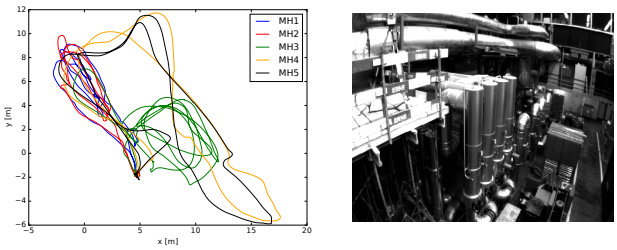

Fig. 4: EuRoC Groundtruth trajectories (left) and screenshot from sequence MH1 (right)

B. Performance metrics

In each scenario and for each robot, we first evaluated the accuracy of its trajectory estimates. As advocated in [28], trajectories are first aligned with the groundtruth on the 3D similarity group $\mathbb{S}\text{im}_3$, to compute the Absolute Translation Errors (ATE) and the Absolute Rotation Errors (ARE):

$$\begin{bmatrix}\boldsymbol{\delta\theta}_n & \boldsymbol{\delta p}_n\end{bmatrix}^\top = \hat{T}_{\text{WI}_n} \boxminus T_{\text{WI}_n} \in \mathbb{R}^6 \tag{8a}$$

$$\text{ATE} = \sqrt{\sum_{n=1}^{N} \frac{\|\boldsymbol{\delta p}_n\|^2}{N}} \quad \text{ARE} = \sqrt{\sum_{n=1}^{N} \frac{\|\boldsymbol{\delta\theta}_n\|^2}{N}} \tag{8b}$$

where $\hat{(\cdot)}$ denotes the estimate. For each robot in each scenario, we jointly aligned all the trajectories to evaluate those metrics. The joint accuracy thus also accounts for the error on the alignment of the odometry reference frames. We also evaluated the communication load induced by the exchanged submaps and matches between the robots, as well as the mean duration to compute the TSDF meshes, integrate the received submaps and compute the matches.

C. Experimental results

1) Implementation details: Simulations were carried out using the ROS middleware [29] on an Inter® Xeon(R) W-123 CPU 3.60 GHz × 8 processor. We used the Maplab framework [30] for the pose graph structure and the *OpenChisel* library [22] to handle the TSDF. We set the switch threshold for run distance and accumulated changes in orientation respectively to $l_{\max} = 3m$ and $\theta_{\max} = 90°$, and we used a voxel resolution of 8cm for the TSDF. For submap

matching, we set $\min_{\text{inliers}} = 1000$, $\max_{\text{RMSE}} = 5$cm and $\max_{\angle} = 30°$. Figure 1 displays the map obtained for robot 4 in the multi-robot scenario 3.

Seq.	Length [m]	Duration [s]	ATE [m]	ARE [deg]	Scale [%]	Scenarios 1	2	3	4
EuRoC dataset									
MH1	80.6	182	0.620	14.04	0.490	✗	✗		✗
MH2	73.5	150	0.515	10.66	1.211	✗			✗
MH3	130.9	132	1.018	14.46	1.943		✗		✗
MH4	91.7	99	1.069	10.15	3.245			✗	
MH5	97.6	111	1.049	10.28	3.351			✗	

TABLE I: Properties of the EuRoC scenarios and estimation accuracy of the eVO algorithm on each sequence

2) Processing times: Table II recaps the processing time statistics for the main tasks. Contrary to optimization, submap integration and mesh computation which are performed quickly, the submap matching process appears to be the bottleneck of the method, with a median processing time close to 5 seconds, most of this time being dedicated the ICP alignment.

Task	Duration statistics min [s]	q1 [s]	median [s]	q3 [s]	max [s]	Period [s]
Integration	7.1e-4	1.09e-3	1.52e-3	0.018	0.129	4.5
Matching	0.173	2.910	4.757	6.876	16.77	6.1
Optimization	0.022	0.127	0.199	0.263	0.582	19.0
Mesh computation	0.022	0.055	0.073	0.095	0.264	13.9

TABLE II: Processing times statistics computed from all the multi-robot scenarios. $q1$ and $q3$ respectively denote the first and third quartiles of the associated distribution.

3) Communication load: Statistics about the mean memory consumption and exchange periods are reported in Table III. On average, TSDF submaps weight around 4 MBytes (most of this weight being explained by the TSDF representation) and are exchanged every 7 seconds, which is affordable in a multi-robot framework.

Item	Weight	Sending period [s]	Reception period 2 robots [s]	3 robots [s]
Submaps	3,9 MB ($\pm$ 1.2 MB)	7.4	5.7	5.6
Matches	288 B	10.3	9.4	5.3

TABLE III: Mean weight of exchanged items and exchange periods averaged over all the scenarios

4) Spotted matches: The numbers of spotted matches for each robot are respectively reported in Tables VI and V for the single and multi robot cases, while errors statistics on the derived relative poses are reported in Table IV. On average, the median error on matches is about 10cm. Low precision may result from the sparsity of the registered point clouds, extracted from the submap meshes.

5) Estimation accuracy: Joint trajectory accuracies and are respectively reported in Tables VI and VII for single and multi-robot cases. While spotted matches combined to pose graph optimization help to slightly improve the accuracy on the single-robot case, they do not bring significant gains in accuracy in the multi-robot case as accuracies on the host robot trajectories are comparable to the single-robot case. Notheless, joint accuracy metrics show that each robot manages to correctly align the received submaps into its own map.

Sc.	Total matches	Submap match ATE distributions [m] min	q1	median	q3	max
1	37	0.040	0.073	0.126	0.162	0.299
2	18	0.026	0.088	0.108	0.154	0.232
3	20	0.029	0.079	0.111	0.173	0.239
4	47	0.025	0.079	0.132	0.195	0.317

TABLE IV: Submap match accuracy statistics. $q1$ and $q3$ respectively denote the first and third quartiles of the associated distribution.

Sc.	Robot	Submap Matches Inter-Robot	Spotted	Attempted
1	**Robot 1**	11	19	30
	Robot 2	10	18	31
2	**Robot 1**	2	9	41
	Robot 3	2	9	44
3	**Robot 4**	9	13	37
	Robot 5	4	7	40
4	**Robot 1**	10	18	37
	Robot 2	9	15	30
	Robot 3	5	14	34

TABLE V: Submap matching statistics

Robot	Joint trajectories ATE [m]	ARE [deg]	Scale [%]	Number of spotted/attempted submap matches
Robot 1	0.320	6.484	0.81	5 / 21
Robot 2	0.310	8.396	0.34	5 / 17
Robot 3	0.843	10.970	0.10	11 / 28
Robot 4	1.003	8.095	3.71	1 / 14
Robot 5	0.851	7.116	3.26	2 / 19

TABLE VI: Trajectory accuracies for the single-robot case

Sc.	Robot	Joint trajectories ATE [m]	ARE [deg]	Scale [%]	Host robot trajectory ATE [m]	ARE [deg]	Scale [%]
1	**Robot 1**	0.316	7.447	0.13	0.318	6.454	0.58
	Robot 2	0.316	7.484	0.21	0.308	8.326	0.01
2	**Robot 1**	0.630	9.083	0.62	0.347	6.479	1.74
	Robot 3	0.631	9.049	0.78	0.799	10.94	0.65
3	**Robot 4**	0.884	7.618	3.43	0.931	8.074	3.80
	Robot 5	0.904	7.608	3.38	0.809	7.100	2.90
4	**Robot 1**	0.539	8.784	0.53	0.326	6.455	0.91
	Robot 2	0.541	8.785	0.54	0.311	8.336	0.04
	Robot 3	0.540	8.801	0.59	0.797	10.99	0.29

TABLE VII: Joint trajectory RMSE (computed over all the trajectories for each robot in each scenario) vs Host robot trajectory accuracy

XI. Conclusion and perspectives

In this article, we proposed a decentralized multi-robot method for simultaneous localization and dense 3D mapping based on the construction and the exchange of locally consistent TSDF submap. We evaluated it on some multi-robot scenarios built from the EuRoC sequences to demonstrate its suitability to real-time requirements, and its ability to build a locally consistent global map integrating the data received from other robots.

The proposed method shows some limitations which should be addressed in future works so as to ensure scalability with the number of robots and improve the matching

performance. The first limitation stems from the submap matching process. First, the search for candidate submaps for matching could benefit from place-recognition approaches based for instance on sparse point cloud descriptors [31], which could also help to pre-align submaps when dealing with significant drift. Furthermore, ICP is known to require a good initial alignment not to converge to local minima. Lacking a pre-alignment step since the extracted clouds are too sparse to use classical 3D keypoints and descriptors for matching, the current matching process hence restricts to low drift hypothesis. Besides, increasing the resolution of the TSDF to get denser point clouds vainly affects the system performances. A more extensive evaluation of the method on dataset with longer trajectories is necessary to assess its performances in face of significant drifts. Furthermore, a perspective would be to adapt the ICP-based matching to the Euclidean SDF representation, as proposed in *Voxgraph* [32]. The submap registration appears to be the bottleneck of the proposed framework and should be made faster. Potential matches could also be organized into a priority queue according to their heuristically estimated information gain as done by [16]. Finally, even though the full back optimization is performed quickly, it may benefit from an incremental formulation.

ACKNOWLEDGEMENTS

This work was supported by the *Direction Générale de l'Armement* (DGA). We also thank Thibault Duhautbout for providing the source code of [1].

REFERENCES

[1] T. Duhautbout, J. Moras, and J. Marzat, "Distributed 3d tsdf manifold mapping for multi-robot systems," in *2019 European Conference on Mobile Robots (ECMR)*, Sep. 2019, pp. 1–8.

[2] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

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

[4] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016.

[5] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph." Robotics: Science and Systems, 2015.

[6] J.-E. Deschaud, "Imls-slam: scan-to-model matching based on 3d data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2480–2485.

[7] C. Brand, M. J. Schuster, H. Hirschmüller, and M. Suppa, "Submap matching for stereo-vision based indoor/outdoor slam," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5670–5677.

[8] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[10] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1116–1130, 2013.

[11] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8648–8654.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking." in *ISMAR*, vol. 11, no. 2011, 2011, pp. 127–136.

[13] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 Ieee/rsj International Conference on Intelligent Robots and Systems (iros)*. IEEE, 2017, pp. 1366–1373.

[14] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena, "C-blox: A scalable and consistent tsdf-based dense mapping approach," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 995–1002.

[15] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3d segment mapping using data-driven descriptors," in *Robotics: Science and Systems (RSS)*, 2018.

[16] M. J. Schuster, K. Schmid, C. Brand, and M. Beetz, "Distributed stereo vision-based 6d localization and mapping for multi-robot teams," *Journal of Field Robotics*, vol. 36, no. 2, pp. 305–332, 2019.

[17] J. Solà Ortega, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018.

[18] M. Sanfourche, V. Vittori, and G. Le Besnerais, "evo: A realtime embedded stereo odometry for mav applications," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2107–2114.

[19] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian conference on computer vision*. Springer, 2010, pp. 25–38.

[20] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

[21] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3400–3407.

[22] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields." in *Robotics: science and systems*, vol. 4, 2015, p. 1.

[23] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4475–4483.

[24] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[25] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[26] S. Agarwal and K. Mierle, "Ceres solver: Tutorial & reference," *Google Inc*, vol. 2, p. 72, 2012.

[27] G. Schweighofer and A. Pinz, "Robust pose estimation from a planar target," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2024–2030, 2006.

[28] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[30] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[31] T. Cieslewski, E. Stumm, A. Gawel, M. Bosse, S. Lynen, and R. Siegwart, "Point cloud descriptors for place recognition using sparse visual information," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4830–4836.

[32] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, 2019.