RO-LOAM: 3D Reference Object-based Trajectory and Map Optimization in LiDAR Odometry and Mapping

Martin Oelsch, *Graduate Student Member, IEEE*, Mojtaba Karimi, and Eckehard Steinbach, *Fellow, IEEE*

***Abstract*—We propose an extension to the LiDAR Odometry and Mapping framework (LOAM) that enables reference object-based trajectory and map optimization. Our approach assumes that the location and geometry of a large reference object are known, e.g., as a CAD model from Building Information Modeling (BIM) or a previously captured dense point cloud model. We do not expect the reference object to be present in every LiDAR scan. Our approach uses the poses of the LOAM algorithm as an initial guess to refine them with scan-to-model alignment. To evaluate if the alignment was accurate, an EKF-based motion prior filtering step is employed. Subsequently, the past trajectory is optimized by adding the model-aligned pose as a pose graph constraint and the map of the LOAM algorithm is corrected to improve future localization and mapping. We evaluate our approach with data captured in a visual airplane inspection scenario inside an aircraft hangar. A 3D LiDAR sensor is mounted via a gimbal on an Unmanned Aerial Vehicle (UAV) and is continuously actuated. We compare the localization accuracy of the LOAM and R-LOAM algorithms when enabling or disabling our proposed reference object-based trajectory and map optimization extension. For three recorded datasets, enabling the proposed extension yields a reduction in Absolute Pose Error compared to conventional LOAM and R-LOAM, while being able to run online. This reduces drift and improves map quality.**

***Index Terms*—SLAM, localization, mapping, range sensing.**

I. Introduction

UAVS are increasingly used for visual inspection tasks. Especially for infrastructure with limited access, such as bridges or skyscrapers, UAVs can significantly speed up and simplify visual inspection for defects. Often, architectural CAD models of the objects exist, or dense point clouds can be created with highly accurate devices from BIM. These models can be leveraged as prior knowledge to improve 3D LiDAR-SLAM accuracy. Visual inspections are a repetitive process. Hence, 3D models can be reused many times as prior knowledge. To take inspection images, UAVs need to get as close as possible to the object. This demands for high-accuracy SLAM and navigation.

We propose an approach for reference object-based trajectory and map optimization (TMO). It serves as an extension to existing LiDAR-SLAM algorithms. We term the approach Reference Object-LOAM (RO-LOAM). The approach assumes that the position of a static, large reference object is either known in a global coordinate frame or determinable and that the geometry, e.g., a 3D model in the form of a dense point cloud, is available. Essentially, our approach refines the poses of the SLAM algorithm and only triggers a TMO when a highly accurate scan-to-model alignment relative to the reference object could be verified. For this, LiDAR scans are registered to the model using Iterative Closest Point (ICP) to refine the SLAM pose. However, not all scans converge well and the Mean Squared Errors (MSEs) from the alignments are unsuitable to ensure a low pose error due to alignment ambiguities. Hence, we propose to evaluate a sequence of model-aligned poses with a motion prior filtering step. It is based on an Extended Kalman Filter (EKF) and used to determine the accuracy of model-aligned poses. If successful, the last refined pose of the sequence is added as a high-confidence constraint to a pose graph. A Pose Graph Optimization (PGO) adjusts the trajectory between two TMOs. The map of the LiDAR-SLAM algorithm can then be corrected with the PG-optimized poses, improving subsequent localization accuracy and map quality. The LOAM algorithm [1] itself remains unmodified and only if there is an information gain with an accurate model-aligned pose, the map is corrected.

The proposed approach may be applied to indoor or outdoor scenarios when going around or above a reference object, i.e., during manipulation or inspection tasks without the requirement of permanent visibility. The reference object does not need to be sufficient for a localization-only approach. In this letter, we use a B737 airplane as a reference object in a visual damage inspection scenario inside a hangar. We use a UAV equipped with a 3D LiDAR sensor mounted on a gimbal for continuous actuation. Fig. 1 shows a recorded flight trajectory of our UAV with the airplane as a reference object. Green circles represent successful

Manuscript received February 22, 2022; accepted May 16, 2022. Date of publication May 25, 2022; date of current version June 3, 2022. This letter was recommended for publication by Associate Editor J. Zhang and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported by the Space Agency of the German Aerospace Center with funds from the Federal Ministry for Economic Affairs and Climate Action on the basis of a resolution of the German Bundestag under Grant 20X1707C. *(Corresponding author: Martin Oelsch.)*

The authors are with the Chair of Media Technology of the Department of Electrical and Computer Engineering, and the Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, 80333 Munich, Germany (e-mail: martin.oelsch@tum.de; mojtaba.karimi@tum.de; Eckehard.Steinbach@tum.de).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3177846, provided by the authors. It contains a video giving more details on the proposed approach. This material is 47.2 MB in size.

Digital Object Identifier 10.1109/LRA.2022.3177846

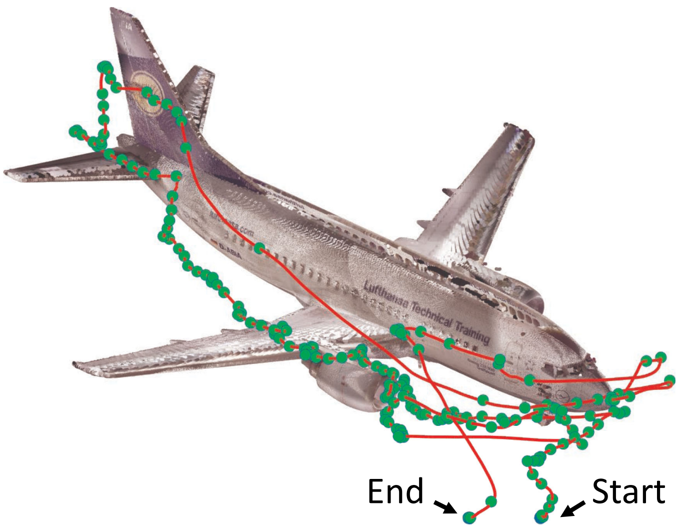


Fig. 1. Exemplary trajectory (red) followed by a UAV along one side of a real B737 airplane for a visual damage inspection. The point cloud model of the airplane shown is used for the proposed reference object-based TMO for 3D LiDAR-SLAM. Green circles represent poses of successful TMOs.

TMOs. We evaluate the proposed approach by attaching our TMO extension to the LOAM [1] and R-LOAM [2] algorithms.

The contributions of this letter can be summarized as follows:

1) A novel extension to 3D LiDAR-SLAM algorithms to enable reference object-based trajectory and map optimization is proposed. The methodology is explained by combining it with the LOAM framework [1].
2) Capability to run online in a fully parallelized manner on an edge cloud in contrast to R-LOAM [2].
3) Results show superior performance when enabling the extension for LOAM [1] and R-LOAM [2] on three datasets in a visual airplane inspection scenario using a UAV.

The rest of this letter is organized as follows: We first discuss related work in Section II and then explain our proposed TMO extension in Section III in detail. In Section IV we elaborate on the acquisition of the datasets and give details on the implementation. The experimental results are presented in Section V and limitations are discussed in Section VI. We conclude the letter in Section VII.

II. Related Work

Several algorithms for LiDAR-SLAM have been proposed in recent years, starting with the use of 2D laser scanners. In the following, we give a brief overview of existing SLAM algorithms. Schadler *et al.* [3] proposed an algorithm based on Monte Carlo Tracking for localization and mapping. They use multi-resolution surfels for point cloud registration from a vertically-mounted and continuously rotating 2D LiDAR sensor. Hess *et al.* [4] then proposed Cartographer as a 2D LiDAR-SLAM algorithm with support for real-time loop closure. Opromolla *et al.* [5] suggested a LiDAR-IMU algorithm suitable for Micro Aerial Vehicles (MAVs). Line features extracted from 2D LiDAR scans are used for mapping and the pose is estimated with a modified version of ICP. Pathak *et al.* [6] proposed a mapping and pose estimation algorithm by extracting and registering planar surface patches from 3D range scans. Kumar *et al.* [7] developed an experimental setup for UAVs with two LiDAR sensors, a primary for horizontal and a secondary for vertical position estimation. The two pose estimates are fused in a Kalman filter. Dröschel *et al.* [8]–[10] proposed several methods for autonomous navigation using multi-resolution maps for 3D LiDAR. Loam_livox [11] is a software package by Lin *et al.*, which was specifically developed for LiDAR sensors with small Field-of-View (FoV), such as solid-state LiDAR sensors, e.g., MEMS. SLOAM [12] by Chen *et al.* uses a Fully Convolutional Neural Network (FCN) for semantic point cloud segmentation to detect trees for accurate localization and mapping in forestry environments on a UAV. Zhang *et al.* [1] proposed a method for LiDAR and LiDAR-IMU localization and mapping with corner and surface features extracted from 2D or 3D LiDAR sensors. The method is generally known as LOAM (LiDAR Odometry and Mapping). Several variations of the LOAM algorithm have been proposed. LeGO-LOAM [13] leverages the knowledge of an existing ground plane in the lower horizontal scan lines of a 3D LiDAR. It also features loop closure detection using a pose graph. Zhu *et al.* proposed G-LOAM [14], a pose graph-based SLAM algorithm for 3D LiDAR using GPS and loop closure constraints for drift reduction developed for long-term localization.

Also, several works have been proposed, leveraging prior knowledge for LiDAR-SLAM. Sandy *et al.* [15] built an In Situ Fabricator. It is equipped with a 2D LiDAR, taking scans while stationary. A construction site and the work piece are modeled as CAD with geometric primitives. The relative pose to the work piece is refined by aligning scans of a Laser Range Finder (LRF) to the CAD model of the work piece. However, no TMO is performed to improve map quality. Mielle *et al.* [16] leverage a 2D emergency map to support SLAM performance. Later, they improved their work [17] by making thresholds less user dependent. Mielle *et al.* also proposed URSIM [18], a method to match 2D sketches to metric maps. Similarly, Boniardi *et al.* [19] improve 2D LiDAR-SLAM performance by leveraging a 2D architectural CAD floor plan. They improved their work later on, by being able to handle not only static, but also changing environments [20]. Gawel *et al.* [21] proposed a system for high-accuracy robotic building construction. A 3D CAD model is point sampled and converted to an initial map for localization. For high-accuracy localization of the end-effector, ray-tracing into the CAD model with an LRF is performed. This is done by taking several orthogonal range measurements while the robot is static. Héry *et al.* [22], [23] proposed works to estimate the relative pose between two consecutively driving vehicles. They use a 2D LiDAR and a 2D polygonal shape of cars.

Most of the discussed approaches leverage prior knowledge in the form of a previously built map or floor plans to improve LiDAR-SLAM performance. To the best of our knowledge, we are the first to leverage a known 3D reference object for trajectory and map optimization with the capability of online edge cloud processing. In this letter, we do not make any assumptions about the environment itself, but only use the known reference object as prior knowledge. Also, the reference object does not need to be

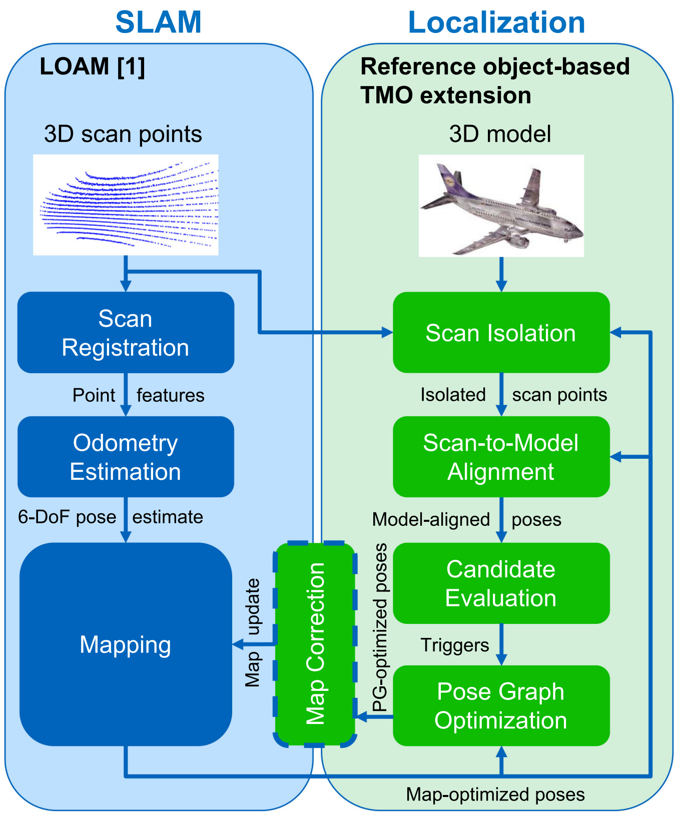


Fig. 2. Overview of the proposed reference object-based TMO pipeline. The left side (blue) shows the conventional LOAM [1] modules and the right side (green) shows our extension. The map correction module represents the tightly-coupled interface between our extension and the LOAM algorithm.

sufficient for a localization-only approach. In our previous work, we proposed R-LOAM [2]. It also leverages prior knowledge about a 3D reference object, but tightly integrates the model into the map optimization process of LOAM [1]. This limits the number of map optimization iterations when running in online mode. In this letter, the proposed extension is independent of the used LiDAR-SLAM algorithm and only the map correction is tightly coupled. In contrast to R-LOAM, the proposed method may be offloaded to an edge cloud, limiting the additional computational load on the robot to a minimum.

III. Methodology

In this section, we explain the proposed reference object-based trajectory and map optimization for the LOAM algorithm [1]. It still performs best on the well-known KITTI odometry benchmark [24] among LiDAR-only algorithms as of today.[1]

Fig. 2 shows an overview of the proposed pipeline. The left side of the figure shows the components of the conventional LOAM algorithm [1]. Our proposed TMO extension is shown on the right side of the figure. The core idea of our extension is to refine a map-optimized pose by global cues to retrieve a highly accurate absolute pose of the robot suitable for trajectory and map optimization. The global cues are the known position and geometry of the reference object in the global frame.

[1]May 2022

A point cloud, e.g., a LiDAR scan $\mathcal{P} = \{p_0, p_1, \ldots, p_l\}$ consists of l individual scan points $p = [x, y, z]$. Point clouds are originally in the local LiDAR sensor frame $\{L\}$ after acquisition at a time t denoted as $\mathcal{P}^{L_t}$. We use the terminology of *world* or $\{W\}$ for the global coordinate frame and $\{D\}$ for the robot frame. A motion is described as a transformation $\boldsymbol{T}$ denoted as a homogeneous 4×4 matrix:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}, \tag{1}$$

with $\boldsymbol{R}$ representing the rotational 3×3 component and $\boldsymbol{t}$ being the 3×1 translational vector. For ease of computation, $\boldsymbol{R}$ may be converted to a quaternion $\boldsymbol{q}$. The forward kinematics of the LiDAR sensor $\{L\}$ in the global frame $\{W\}$ can then be described as

$$^{W}\boldsymbol{T}_L = {}^{W}\boldsymbol{T}_D {}^{D}\boldsymbol{T}_L. \tag{2}$$

$^{D}\boldsymbol{T}_L$ is the transformation of the LiDAR sensor relative to the robot frame. Since the sensor is continuously actuated, this can be computed from the actuator readings. $^{W}\boldsymbol{T}_D$ is the current pose of the robot $\{D\}$ in $\{W\}$, which is estimated by the SLAM algorithm.

A. Loam

The LOAM algorithm [1] consists of three modules: Scan registration, odometry estimation and mapping. In the scan registration module, corner and surface features $\mathcal{P}_{\mathcal{E}}^{L_t}$ and $\mathcal{P}_{\mathcal{H}}^{L_t}$ are extracted from a LiDAR scan $\mathcal{P}^{L_t}$. At this point, we transform the feature points to the robot frame using the actuator transform $^{D_t}\boldsymbol{T}_{L_t}$ to remove the actuation distortion. The feature points are then processed in the odometry estimation module, where the relative motion $^{D_{t-1}}\boldsymbol{T}_{D_t}$ between the last two consecutive LiDAR scans is estimated by a correspondence estimation and optimization step. The mapping module integrates $^{D_{t-1}}\boldsymbol{T}_{D_t}$ and performs two correspondence estimation and map optimization iterations by default. LiDAR scan points are inserted with the map-optimized transform $^{W}\boldsymbol{T}_{D_t}$. The map points are then voxelized and stored in a cube structure to keep computational complexity tractable for future map optimizations. The ultimate output is the map-optimized pose $^{W}\boldsymbol{T}_{D_t}$, which serves as input to our extension.

B. Scan Isolation

For our extension, raw LiDAR scans are used instead of the extracted features. These are first transformed to $\{W\}$ with the actuator readings and the corresponding map-optimized pose $^{W}\boldsymbol{T}_D$ according to Eq. 2. The purpose of scan isolation is to only keep points in the scan, which belong to the reference object. This can be achieved with a bounding box cropping method.

C. Scan-to-Model Alignment

The general purpose of the scan-to-model alignment module is to refine a map-optimized pose $^{W}\boldsymbol{T}_D$ to retrieve a highly accurate model-aligned pose $^{W}\tilde{\boldsymbol{T}}_D$ suitable for trajectory and map optimization. This is achieved by aligning isolated scans to the reference model. However, scan-to-model alignment may even increase the pose error due to a wrong convergence and ambiguities. Hence, we align a short sequence of consecutive isolated scans with their respective map-optimized poses as an initial guess to the model. The converged model-aligned poses are then used as input to an EKF to evaluate the accuracy of the last model-aligned pose in the sequence by comparing it with the motion prior.

After receiving L isolated LiDAR scans, the most recent $M + 1$ scans are aligned to the model, if each of the $M + 1$ scans has more than 50 points. Further scans and corresponding poses are buffered, until this condition is fulfilled. $M + 1$ isolated scans $\mathcal{P}^{D_{t-M:t}}$ captured at times $t - M$ until t are individually aligned to the model in parallelized threads. The parameter L controls the frequency of TMOs and M controls the length of the sequence used for the candidate evaluation in the next step.

The scan-to-model alignment follows an approach based on ICP. Point correspondences $\mathcal{C} = \{c_0, c_1, \ldots, c_n\}$ are estimated between the isolated scan points and the point cloud of the 3D model. It is formulated as a nonlinear least-squares optimization problem with the total cost J:

$$J(\boldsymbol{q}, \boldsymbol{t}) = \sum_{c \in \mathcal{C}} \rho(\|f(c, \boldsymbol{q}, \boldsymbol{t})\|^2). \tag{3}$$

$\boldsymbol{q}$ and $\boldsymbol{t}$ are the quaternion and translational vector being optimized, respectively. The cost function f returns the Euclidean residual for each point correspondence. It is wrapped inside a Huber loss function ρ and is solved with a Levenberg-Marquardt trust-region algorithm.

Before the alignment, we uniformly downsample the isolated scan points to 500 to cap the computational complexity. We use $1\,\mathrm{m}$ maximum correspondence distance to be more robust against outliers, that survived the isolation. Correspondence estimation and optimization are repeated 100 times to retrieve fully model-converged scans with (possibly) very low pose error. We set the maximum convergence time to $2\,\mathrm{s}$ after which the converged pose until then is used.

The ultimate output of the module are the refined poses ${}^{W}\tilde{\boldsymbol{T}}_{D_{t-M:t}}$ with ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$ being the candidate. ${}^{W}\tilde{\boldsymbol{T}}_{D_{t-M:t}}$ are sent to the EKF for evaluation, if the MSE of ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$ after scan-to-model alignment is smaller than 0.001. Note that some model-aligned poses may have a large error despite of a small MSE. If the alignment in the sequence worked well, the last aligned pose will pass the motion prior filtering step described in the following section.

D. Candidate Evaluation

To find out if the scan-to-model alignment of the sequence worked well and if ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$ is a good candidate for TMO, we leverage a motion prior based on ${}^{W}\tilde{\boldsymbol{T}}_{D_{t-M:t-1}}$ using an EKF. It follows the common structure of a prediction and correction step. The predicted state $\boldsymbol{x}_{K|K-1}$ and state covariance $\boldsymbol{P}_{K|K-1}$ are obtained with

$$\begin{aligned}\boldsymbol{x}_{K|K-1} &= f(\boldsymbol{x}_{K-1|K-1})\\ \boldsymbol{P}_{K|K-1} &= \boldsymbol{F}_{K-1}\boldsymbol{P}_{K-1|K-1}\boldsymbol{F}_{K-1}^T + \boldsymbol{Q}_{K-1}.\end{aligned} \tag{4}$$

$f(\cdot)$ is the state transition function, $\boldsymbol{F}_{K-1} = \frac{\partial f}{\partial \boldsymbol{x}}|_{\boldsymbol{x}_{K-1|K-1}}$ is its Jacobian and $\boldsymbol{Q}_{K-1}$ is the constant process noise covariance. The correction step first computes the Kalman gain:

$$\boldsymbol{K}_K = \frac{\boldsymbol{P}_{K|K-1}\boldsymbol{H}_K^T}{\boldsymbol{H}_K\boldsymbol{P}_{K|K-1}\boldsymbol{H}_K^T + \boldsymbol{R}_K} \tag{5}$$

$\boldsymbol{H}_K$ is the measurement model matrix and $\boldsymbol{R}_K$ is the measurement noise covariance. The Kalman gain is then used to update the state $\boldsymbol{x}_{K|K}$ and state covariance $\boldsymbol{P}_{K|K}$:

$$\begin{aligned}\boldsymbol{x}_{K|K} &= \boldsymbol{x}_{K|K-1} + \boldsymbol{K}_K(\boldsymbol{z}_K - \boldsymbol{H}_K\boldsymbol{x}_{K|K-1})\\ \boldsymbol{P}_{K|K} &= (\boldsymbol{I} - \boldsymbol{K}_K\boldsymbol{H}_K)\boldsymbol{P}_{K|K-1}(\boldsymbol{I} - \boldsymbol{K}_K\boldsymbol{H}_K)^T\\ &\quad + \boldsymbol{K}_K\boldsymbol{R}_K\boldsymbol{K}_K^T\end{aligned} \tag{6}$$

The model-aligned poses ${}^{W}\tilde{\boldsymbol{T}}_{D_{t-M:t-1}}$ are used as measurement input $\boldsymbol{z}_K$. For each measurement, a prediction and correction step are performed. We take the predicted state $\boldsymbol{x}_{K|K-1}$ as motion prior ${}^{W}\hat{\boldsymbol{T}}_{D_t}$ for the last model-aligned pose ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$. If the Euclidean distance of the translational vector between the two transformations is $< 0.05\,\mathrm{m}$ and the rotational distance is $< 0.5^\circ$, then ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$ is following the motion model of the previous scan-to-model alignments and is therefore considered accurate enough to be a good pose for TMO. Note that for each input sequence ${}^{W}\tilde{\boldsymbol{T}}_{D_{t-M:t-1}}$, the state of the EKF is reinitialized. ${}^{W}\tilde{\boldsymbol{T}}_{D_t}$ is then sent to the PGO module for trajectory optimization.

E. Pose Graph Optimization (PGO)

The PGO allows for correcting the trajectory since the last TMO. With the corrected poses, the map of the SLAM algorithm can then be corrected.

Incoming map-optimized poses ${}^{W}\boldsymbol{T}_D$ from the mapping module are inserted as new nodes to the pose graph. The pose graph $\mathcal{G} = \{n_0, n_1, \cdots\}$ consists of nodes $n = \langle \boldsymbol{q}, \boldsymbol{t} \rangle$. The nodes are equal to a transformation ${}^{W}\boldsymbol{T}_D$ from the mapping module before the optimization. The nodes are connected by edges $\mathcal{E} = \{e_{0,1}, e_{1,2}, \cdots\}$. An edge $e_{i,j} = \langle \boldsymbol{\Omega}_{i,j}, \boldsymbol{q}_{i,j}, \boldsymbol{t}_{i,j} \rangle$ represents a graph constraint connecting the nodes n_i and n_j. $\boldsymbol{q}_{i,j}$ and $\boldsymbol{t}_{i,j}$ describe the relative pose between the two nodes and $\boldsymbol{\Omega}_{i,j}$ is the 6×6 information matrix describing the confidence of the relative pose. We use the implementation of the Ceres Solver [25] for the optimization of the pose graph. It is modeled as a nonlinear least-squares optimization problem with the total cost J:

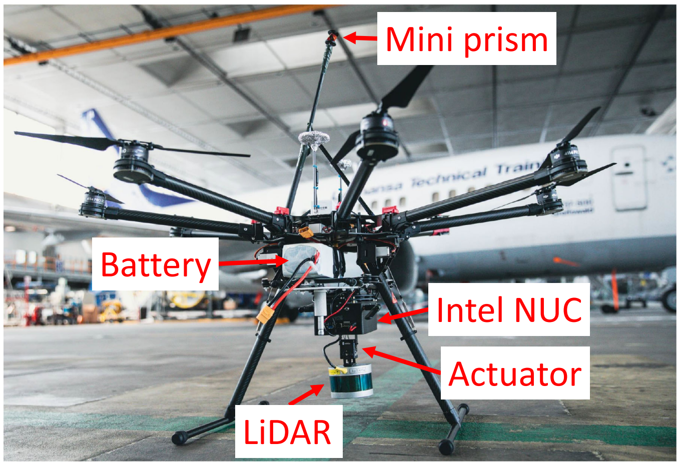
$$\begin{aligned}J(\hat{\boldsymbol{q}}_{i,j}, \hat{\boldsymbol{t}}_{i,j}) &= \sum_{e \in \mathcal{E}} \rho(||f(e_{i,j}, \hat{\boldsymbol{q}}_{i,j}, \hat{\boldsymbol{t}}_{i,j})||^2),\\ \text{with} \quad f(e_{i,j}, \hat{\boldsymbol{q}}_{i,j}, \hat{\boldsymbol{t}}_{i,j}) &= \hat{\boldsymbol{\Omega}}_{i,j}\begin{pmatrix}\Delta \boldsymbol{t}_{i,j}\\ 2\Delta \vec{\boldsymbol{q}}_{i,j}\end{pmatrix},\\ \text{with} \quad \Delta \boldsymbol{t}_{i,j} &= \boldsymbol{t}_{i,j} - \hat{\boldsymbol{t}}_{i,j}\\ \text{and} \quad \Delta \boldsymbol{q}_{i,j} &= \boldsymbol{q}_{i,j}\hat{\boldsymbol{q}}_{i,j}^*\end{aligned} \tag{7}$$

$f(\cdot)$ represents the cost function returning one residual and is wrapped inside a Huber loss function ρ. $\hat{\boldsymbol{q}}_{i,j}$ and $\hat{\boldsymbol{t}}_{i,j}$ are optimized iteratively and computed from the respective nodes n_i and n_j. $\Delta\vec{\boldsymbol{q}}_{i,j}$ is the vector part of $\Delta\boldsymbol{q}_{i,j}$. $\boldsymbol{q}_{i,j}$ and $\boldsymbol{t}_{i,j}$ are the relative pose constraints from the edge $e_{i,j}$. $\hat{\boldsymbol{\Omega}}_{i,j}$ is the lower triangular matrix of the Cholesky-decomposed information matrix $\boldsymbol{\Omega}_{i,j}$.

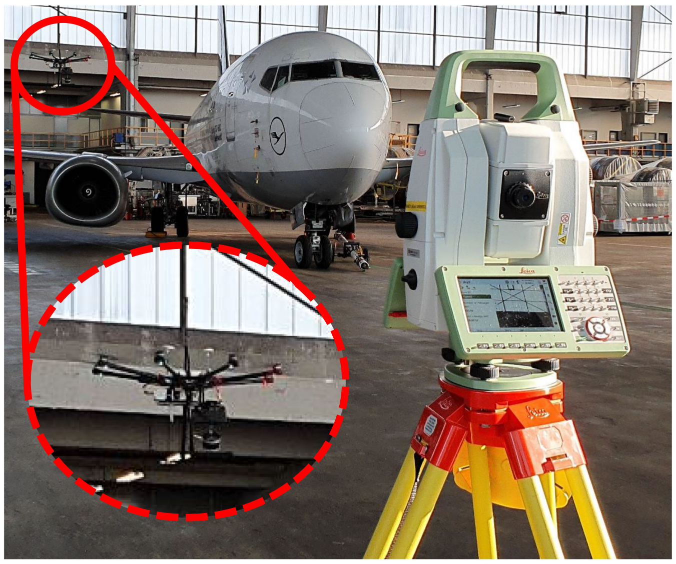
After the verified pose ${}^W\tilde{\boldsymbol{T}}_{D_t}$ has been received, a new subgraph is created with graph nodes from the map-optimized poses ${}^W\boldsymbol{T}_{D_{t-X:t}}$. The previous TMO was at time $t-X$ and the most recent at time t. Subsequently, edge constraints are created by computing the relative poses between the nodes. The relative pose between a pose at time a and a pose at time b can be determined with ${}^{D_a}\boldsymbol{T}_{D_b} = {}^W\boldsymbol{T}_{D_a}^{-1}\,{}^W\boldsymbol{T}_{D_b}$. For each relative pose, we use an identity information matrix $\boldsymbol{\Omega} = \boldsymbol{I}$, which treats each edge equally during the optimization. Finally, the relative pose between the previous and current TMO poses is computed as ${}^{D_{t-X}}\boldsymbol{T}_{D_t} = {}^W\boldsymbol{T}_{D_{t-X}}^{-1}\,{}^W\boldsymbol{T}_{D_t}$ and the constraint is added with a high confidence value in the information matrix, e.g., $\boldsymbol{\Omega}_{t-X,t} = 4000 * \boldsymbol{I}$. We set the first node of the subgraph, which is the pose of the previous TMO, as constant, so the optimizer can not change it. We use the trust-region Levenberg-Marquardt solver for PGO. Since the first node is constant and a highly weighted constraint is added for the current TMO pose, only the nodes in between will be adjusted during the optimization process. The output of the PGO module is ${}^W\bar{\boldsymbol{T}}_{D_{t-X:t}}$, which are the PG-optimized poses with ${}^W\bar{\boldsymbol{T}}_{D_{t-X}} \widehat{=} {}^W\tilde{\boldsymbol{T}}_{D_{t-X}}$ and ${}^W\bar{\boldsymbol{T}}_{D_t} \widehat{=} {}^W\tilde{\boldsymbol{T}}_{D_t}$.

F. Map Correction

With the PG-optimized poses ${}^W\bar{\boldsymbol{T}}_{D_{t-X:t}}$, the map can be corrected to reduce drift since the last PGO, i.e., by reinserting previous LiDAR scans with the PG-optimized poses. The map correction module is an interface between our reference object-based TMO extension and the LiDAR-SLAM algorithm. Note that this is the only module in our pipeline, which is tightly coupled to the LiDAR-SLAM algorithm since it is highly dependent on the map structure of the mapping module.



(a) Octocopter UAV



(b) Leica ground-truth system

Fig. 3. Illustration of the UAV platform (a) and the Leica ground-truth system during dataset recording (b).

IV. Experimental Setup

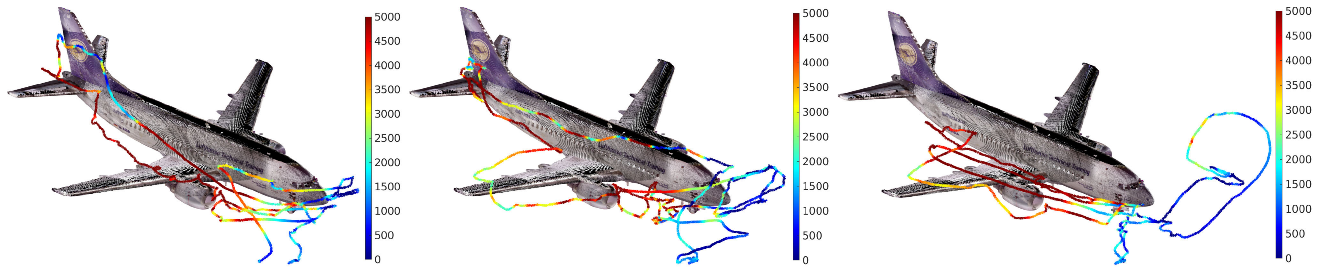
A. Dataset

We use an octocopter UAV platform to carry a Velodyne VLP-16 3D LiDAR mounted on a Dynamixel actuator. It allows us to continuously rotate the LiDAR sensor throughout the flight to increase the FoV even when the platform is static [1], [9], [26], [27]. The LiDAR provides an update rate of 10 Hz. The actuator rotates the LiDAR around the roll axis between $\pm 40^\circ$. An Intel NUC saves the LiDAR data on an SSD drive for offline evaluation.

The UAV is controlled by an operator inside a hangar on one side of a B737 airplane along suitable trajectories for visual surface inspection. A Leica Nova MS60 MultiStation measures the ground-truth 3-DoF position (no rotation) of a low-weight Leica mini prism mounted on top of a rod for good visibility throughout the flight. The scanning functionality of the Leica system is used to generate a highly accurate 3D point cloud model of the B737 airplane (see Fig. 1). It consists of over $1.6\,\mathrm{M}$ points and has a mean Euclidean distance to the nearest neighbor of 1.4 cm. Fig. 3(a) shows the UAV platform and (b) shows the Leica ground-truth system in operation during a dataset recording.

Fig. 4 shows the trajectories of three generated datasets and the airplane model as a reference object. The color indicates the number of scan points on the reference object. Dataset 2 has the longest trajectory with $211\,\mathrm{m}$ and over 7000 LiDAR scans. Dataset 3 also includes some scans from the other side of the airplane. Especially at the front of the airplane, sometimes no scan points belong to the reference object. This makes a pure localization approach without SLAM impossible.

Time synchronization between the Leica and SLAM trajectories was achieved by finding the minimum MSE between the aligned trajectories. For this, the time offset was adjusted in a sliding window approach.



(a) Dataset 1
#scans: 5076, avg. vel: 0.33 m/s,
length: 162 m

(b) Dataset 2
#scans: 7034, avg. vel: 0.30 m/s,
length: 211 m

(c) Dataset 3
#scans: 5712, avg. vel: 0.39 m/s,
length: 196 m

Fig. 4. Illustration of the trajectories for each dataset along a B737 airplane used as a reference object. The colors indicate the number of scan points on the reference object captured with the LiDAR sensor at the corresponding position.

B. Implementation

We use the A-LOAM implementation[2] of LOAM [1], R-LOAM [2] and ROS melodic[3] for our experiments. We use the robot localization package[4] of ROS for the EKF. For the measurement noise covariance, we use $\boldsymbol{R}_K = 0.01 * \boldsymbol{I}$. The other parameters and matrices are left as default.

The relative pose between the reference object and LiDAR coordinate systems is estimated with a scan-to-model alignment method (see Section III-C) using the accumulated first 20 scans of the respective dataset, which were acquired when the UAV and LiDAR were still static. The relative pose estimation is required before each exploration. By determining the relative pose, the position of the airplane in the map frame is known. The RMSE amounts to < 3 cm for each of the three datasets. After the relative pose estimation, the proposed method is initialized. The experiments were conducted on a server with 32 Intel Xeon CPUs E5-2690 @ 2.90 GHz and 132 GB memory.

V. Results

Our reference object-based TMO extension is decoupled from the LiDAR-SLAM algorithm and can therefore be used for other methods also. In this section, we present the results of our extension for LOAM [1] and R-LOAM [2], denoted as LOAM + RO and R-LOAM + RO, respectively. We compare the results with LOAM and R-LOAM without our extension. All experiments were performed online, i.e., with 10 Hz LiDAR scans.

Table I shows the percentage of scan-to-model aligned poses below a certain Absolute Pose Error (APE), using the map-optimized poses of the LOAM algorithm [1] as an initial guess. The results show that only few scan-to-model aligned poses actually qualify for TMO with APE < 10 cm. This shows that a pure reference object-based localization without a SLAM algorithm would fail, also considering that the reference object is not visible in every scan. If a scan-to-model alignment works well, highly depends on the current scan of the reference object due to ambiguities. Hence, the MSE of the scan-to-model alignment is not suitable as covariance. It is the task of the EKF to evaluate

TABLE I
Percentage of Scan-to-Model Aligned Poses Below a Certain APE. The Map-Optimized Poses of the LOAM Algorithm [1] Were Used as Initial Guess

APE	Scan-to-model aligned poses (%)		
	Dataset 1	Dataset 2	Dataset 3
< 10 cm	9	16	7
< 50 cm	32	41	25
< 100 cm	50	59	39

TABLE II
Experimental Results for Variations of the Parameter M on Dataset 1 for LOAM + RO With $L = 50$. $M + 1$ Scans Are Used for the Scan-to-Model Alignment. The Best Results Are Marked in **Bold**

M	APE mapping (cm)				#TMOs
	Max	Mean	Median	RMSE	
4	79.5	11.7	9.0	16.8	18
9	94.9	**9.9**	**6.3**	16.6	**25**
19	96.7	13.7	9.1	20.4	19
29	**58.5**	11.5	8.9	**15.7**	20

with the motion prior if the TMO candidate has a possibly low error.

The parameter M controls the number of scans used for the scan-to-model alignment and influences the quality of the motion model inside the EKF. We performed experiments varying $M = [4, 29]$ and analyzed the APE for $L = 50$ on Dataset 1 with LOAM + RO. Table II shows that a value of $M = 9$ results in the lowest mean and median APE while having the most successful TMOs. $M = 4$ results in a shorter input sequence for the EKF. Since the state transition function and $\boldsymbol{P}_{K-1|K-1}$ are reinitialized for each sequence, several prediction and correction steps are required to output a reliable motion prior suitable for TMO candidate evaluation. $M > 9$ makes wrongly converged scan-to-model alignments in the sequence more likely, which reduces the chance of the motion prior to being close to the last model-aligned pose. In both cases, the number of TMOs decreases. Hence, considering the state transition function to be unknown, we will use $M = 9$ for the following experiments.

[2][Online]. Available: https://github.com/HKUST-Aerial-Robotics/A-LOAM
[3][Online]. Available: http://wiki.ros.org/melodic
[4][Online]. Available: http://wiki.ros.org/robot_localization

TABLE III
EXPERIMENTAL RESULTS FOR VARIATIONS OF THE PARAMETER L ON DATASET 1 FOR LOAM + RO WITH $M = 9$. A SCAN-TO-MODEL ALIGNMENT IS TRIGGERED EVERY L SCANS. THE BEST RESULTS ARE MARKED IN **BOLD**

L	APE mapping (cm)				#TMOs
	Max	Mean	Median	RMSE	
15	**84.2**	**8.9**	**6.5**	**13.7**	**62**
100	96.3	17.0	11.6	24.4	12
200	91.9	17.3	13.3	23.0	6
300	92.6	24.4	17.4	31.0	4

Table III shows the influence of the TMO frequency on the APE by varying the parameter $L = [15, 300]$ for Dataset 1 with LOAM + RO. A higher value for L buffers more isolated scans before triggering a scan-to-model alignment. The results show that a lower value for L leads to more successful TMOs and hence, also reduces the APE.

The results of the three datasets for $M = 9$ and $L = 15$ can be seen in Table IV. Since the results are not fully reproducible in online mode, we run each of the experiments with the same parametrization 5 times and show the mean values. The mapping module of LOAM [1] performs frame-skipping if the rate of map optimization is lower than the rate of incoming scans. *APE mapping* shows the errors of the map-optimized poses of the mapping module and *APE TMO* shows the errors of the poses used for TMO only. Note that the *APE mapping* is only indirectly influenced by our extension and is improved solely due to the increased map quality and corrected drift from the map correction process.

R-LOAM shows a significantly reduced error compared to LOAM. The latter shows a large drift due to wrong scan matching for all three datasets, which can be prevented by using the reference model. The results show an improvement in mean and median APE for all three datasets when activating our TMO extension for LOAM and R-LOAM. Note that R-LOAM uses the same airplane model in its joint optimization as we use in our extension. The main advantage is the offloaded and parallelized process, where the scan-to-model convergence can run for many more iterations compared to R-LOAM. This allows our extension to further reduce the APE of R-LOAM. The EKF-based evaluation module effectively filters out bad TMO candidates, resulting in over 50 successful TMOs with a maximum APE of < 30 cm and a median APE of < 5 cm for all datasets.

To summarize the results, activating our extension reduces the APE for LOAM and R-LOAM at nearly no additional computational cost on the onboard computer, when running in a remote SLAM setup.

Finally, Fig. 5 shows an example of the final SLAM map when running the R-LOAM + RO algorithm on Dataset 1 and the dense airplane model as a reference object. It gives an impression of the ratios between the whole environment used by the SLAM algorithm and the reference object. Despite the UAV flying close to the airplane, the majority of map points belong to the hangar and interior.

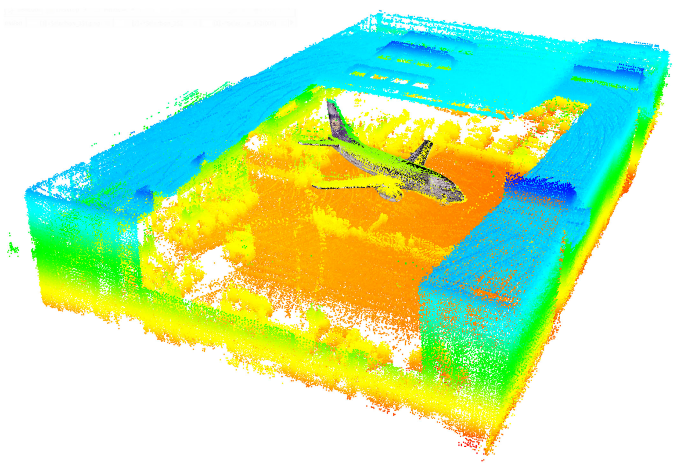


Fig. 5. Illustration of the final feature map of R-LOAM + RO for Dataset 1 and the dense airplane model as a reference object. Parts of the hangar roof are removed for illustrative purposes.

VI. Limitations

We want to point out that the proposed method can be used as a plug-in extension for other LiDAR-SLAM algorithms. It may even be employed complementarily to relocalization or loop closure methods. The proposed method is not a standalone LiDAR-SLAM algorithm and we would like to highlight the relative benefit of leveraging a known 3D model of a reference object for TMO.

Similar to R-LOAM, the proposed approach is relying on the accurate initial relative pose estimation of the robot to the reference object as described in Section IV-B. Without initial guess, global registration methods can be employed in combination with ICP refinement. However, any error will have a direct negative impact on the following trajectory and map optimization performance. Deviations between the model and the actual geometry of the reference object will also increase the error during scan-to-model alignment. Outlier rejection methods can be used and the correspondence distance threshold adjusted to deal with this case.

VII. Conclusion

In this letter, we proposed a reference object-based trajectory and map optimization extension for 3D LiDAR-SLAM algorithms termed RO-LOAM, capable of running online on an edge cloud. We use the well-known LOAM framework [1] as an example LiDAR-SLAM system to introduce our extension. Our approach leverages the known geometry and location of a 3D reference object to perform trajectory and map optimization. For this, a sequence of consecutive LiDAR scans is aligned to the 3D model and the aligned poses are evaluated by an EKF motion prior. If successful, the trajectory is optimized and the map of the LOAM algorithm [1] is corrected. We evaluated the approach with data from a visual airplane inspection scenario inside a hangar with a 3D LiDAR sensor mounted on a UAV. We activated our extension for the LOAM [1] and R-LOAM [2] algorithms and found consistent improvements in terms of Absolute Pose Error (APE) for three datasets. Future work may investigate the

TABLE IV
EXPERIMENTAL RESULTS FOR THE THREE DATASETS. THE BEST RESULTS ARE MARKED IN **BOLD**

	Method	**APE mapping** (cm)				**Freq.** (Hz)	**APE TMO** (cm)				**#TMOs**
		Max	Mean	Median	RMSE	mapping	Max	Mean	Median	RMSE	
Dataset 1	LOAM [1]	466.3	134.6	71.1	182.7	**3.3**	-	-	-	-	-
	LOAM + RO	**84.2**	**8.9**	**6.5**	**13.6**	3.0	20.3	5.2	4.2	6.4	62
	R-LOAM [2]	**50.7**	12.6	11.5	14.4	**3.1**	-	-	-	-	-
	R-LOAM + RO	57.8	**8.1**	**5.9**	**10.5**	**3.1**	22.9	5.2	4.1	6.7	62
Dataset 2	LOAM [1]	423.2	117.1	67.2	164.7	**2.9**	-	-	-	-	-
	LOAM + RO	**89.7**	**9.6**	**6.4**	**15.7**	2.6	18.2	4.0	3.3	5.2	62
	R-LOAM [2]	**58.4**	10.3	9.6	12.4	**2.8**	-	-	-	-	-
	R-LOAM + RO	65.0	**7.2**	**5.2**	**10.4**	2.6	16.8	4.1	3.6	5.1	64
Dataset 3	LOAM [1]	354.8	123.8	80.6	154.8	**3.2**	-	-	-	-	-
	LOAM + RO	**88.4**	**13.2**	**6.8**	**21.8**	2.9	28.1	4.6	3.6	6.5	54
	R-LOAM [2]	69.7	16.9	15.3	19.4	**3.0**	-	-	-	-	-
	R-LOAM + RO	**54.8**	**9.3**	**6.7**	**12.2**	**3.0**	23.9	5.3	3.9	7.2	58

benefit of adding IMU data to the EKF for further improvement of the candidate evaluation.

ACKNOWLEDGMENT

We want to express our gratitude to Jan-Christopher Knufinke at Lufthansa Technik for the organization of all the hangar visits.

REFERENCES

[1] J. Zhang and S. Singh, “LOAM: LiDAR odometry and mapping in real-time,” *Auton. Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[2] M. Oelsch, M. Karimi, and E. Steinbach, “R-LOAM: Improving lidar odometry and mapping with point-to-mesh features of a known 3D reference object,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2068–2075, Apr. 2021.

[3] M. Schadler, J. Stuckler, and S. Behnke, “Multi-resolution surfel mapping and real-time pose tracking using a continuously rotating 2D laser scanner,” in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, 2013, pp. 1–6.

[4] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D LiDAR SLAM,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1271–1278.

[5] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, and A. Savvaris, “LIDAR-inertial integration for UAV localization and mapping in complex environments,” in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2016, pp. 649–656.

[6] K. Pathak, A. Birk, N. Vaškevičius, and J. Poppinga, “Fast registration based on noisy planes with unknown correspondences for 3-D mapping,” *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 424–441, Jun. 2010.

[7] G. Ajay Kumar, A. K. Patil, R. Patil, S. S. Park, and Y. H. Chai, “A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification,” *Sensors (Switzerland)*, vol. 17, no. 6, 2017, Art. no. 1268.

[8] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, and S. Behnke, “Multilayered mapping and navigation for autonomous micro aerial vehicles,” *J. Field Robot.*, vol. 33, no. 4, pp. 451–475, 2016.

[9] D. Droeschel, M. Schwarz, and S. Behnke, “Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner,” *Robot. Auton. Syst.*, vol. 88, pp. 104–115, 2017.

[10] D. Droeschel and S. Behnke, “Efficient continuous-time slam for 3D LiDAR-based online mapping,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5000–5007.

[11] J. Lin and F. Zhang, “Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3126–3131.

[12] S. W. Chen *et al.*, “SLOAM: Semantic LiDAR odometry and mapping for forest inventory,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 612–619, Apr. 2020.

[13] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[14] Y. Zhu, B. Xue, L. Zheng, H. Huang, M. Liu, and R. Fan, “Real-time, environmentally-robust 3D LiDAR localization,” in *Proc. IEEE Int. Conf. Imag. Syst. Techn.*, 2019, pp. 1–6.

[15] T. Sandy, M. Giftthaler, K. Dörfler, M. Kohler, and J. Buchli, “Autonomous repositioning and localization of an in situ fabricator,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 2852–2858.

[16] M. Mielle, M. Magnusson, H. Andreasson, and A. J. Lilienthal, “SLAM auto-complete: Completing a robot map using an emergency map,” in *Proc. 15th IEEE Int. Symp. Safety, Secur. Rescue Robot., Conf.*, 2017, pp. 35–40.

[17] M. Mielle, M. Magnusson, and A. J. Lilienthal, “The auto-complete graph: Merging and mutual correction of sensor and prior maps for SLAM,” *Robotics*, vol. 8, no. 2, 2019, Art. no. 40.

[18] M. Mielle, M. Magnusson, and A. J. Lilienthal, “URSIM: Unique regions for sketch map interpretation and matching,” *Robotics*, vol. 8, no. 2, 2019, doi: 10.3390/robotics8020043.

[19] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, “Robust LiDAR-based localization in architectural floor plans,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3318–3324.

[20] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, “A pose graph-based localization system for long-term navigation in CAD floor plans,” *Robot. Auton. Syst.*, vol. 112, pp. 84–97, 2019.

[21] A. Gawel *et al.*, “A fully-integrated sensing and control system for high-accuracy mobile robotic building construction,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2300–2307.

[22] E. Héry, P. Xu, and P. Bonnifait, “Pose and covariance matrix propagation issues in cooperative localization with LiDAR perception,” in *Proc. IEEE Intell. Veh. Symp.*, 2019, pp. 1219–1224.

[23] E. Héry, P. Xu, and P. Bonnifait, “LiDAR based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape,” in *Proc. 10th Workshop Planning, Perception Navigation Intell. Vehicles*, 2018.

[24] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[25] S. Agarwal, K. Mierle, and T. C. S. Team, “Ceres solver,” [Online]. Available: http://ceres-solver.org

[26] L. Pfotzer, J. Oberlaender, A. Roennau, and R. Dillmann, “Development and calibration of karola, a compact, high-resolution 3D laser scanner,” in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2014, pp. 1–6.

[27] M. Karimi, M. Oelsch, O. Stengel, E. Babaians, and E. Steinbach, “LoLa-SLAM: Low-latency LiDAR SLAM using continuous scan slicing,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2248–2255, Apr. 2021.