2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

R-LOAM: Improving LiDAR Odometry and Mapping with Point-To-Mesh Features of a Known 3D Reference Object

Martin Oelsch, Mojtaba Karimi and Eckehard Steinbach

***Abstract*— LiDAR-based odometry and mapping is used in many robotic applications to retrieve the robot's position in an unknown environment and allows for autonomous operation in GPS-denied (e.g. indoor) environments. In this paper, we extend the well-known LOAM framework by leveraging prior knowledge about a reference object in the environment to further improve the localization accuracy using a 3D LiDAR sensor. This requires a known 3D model of the reference object and its known position in a global coordinate frame. Instead of only relying on the point features in the mapping module of LOAM, we also include mesh features extracted from the 3D triangular mesh of the reference object in the optimization problem. Essentially, our approach not only makes use of the previously built map for absolute localization in the environment, but also takes the relative position to the reference object into account. We generated datasets using the Gazebo simulation environment in exemplary visual inspection scenarios of an airplane inside a hangar and the Eiffel Tower. An actuated 3D LiDAR sensor is mounted via a 1-DoF gimbal on a UAV capturing 360° scans. We benchmark our approach against the state-of-the-art open-source LOAM framework. The results show that the proposed joint optimization using both point and mesh features yields a significant reduction in Absolute Pose Error (APE) and therefore improves the map and 3D reconstruction quality during long-term operations.**

I. INTRODUCTION

In many application scenarios for autonomous robots, a global positioning system such as GPS is not available, e.g., during extraterrestrial exploration or indoor navigation. Knowing the position in an environment is crucial for position holding, e.g. for a UAV during flight or for autonomous navigation along a pre-defined path. Usually no previous knowledge about the environment is available, so that robots have to create their own map for localization using measurements from sensor data. Various algorithms and frameworks exist for Simultaneous Localization and Mapping (SLAM).

In this paper, we leverage the knowledge of the exact position and geometry of a reference object in the environment to improve the localization accuracy of 3D LiDAR-SLAM. To maximize the performance, the object should be scanned as often as possible. The trajectory of the robot can be inside, go around, or be above the reference object.

*This work was supported by the space agency of the German Aerospace Center with funds from the Federal Ministry of Economics and Technology on the basis of a resolution of the German Bundestag under the reference 20X1707C.

All authors are with the Chair of Media Technology of the Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany
Email: martin.oelsch@tum.de, mojtaba.karimi@tum.de, eckehard.steinbach@tum.de

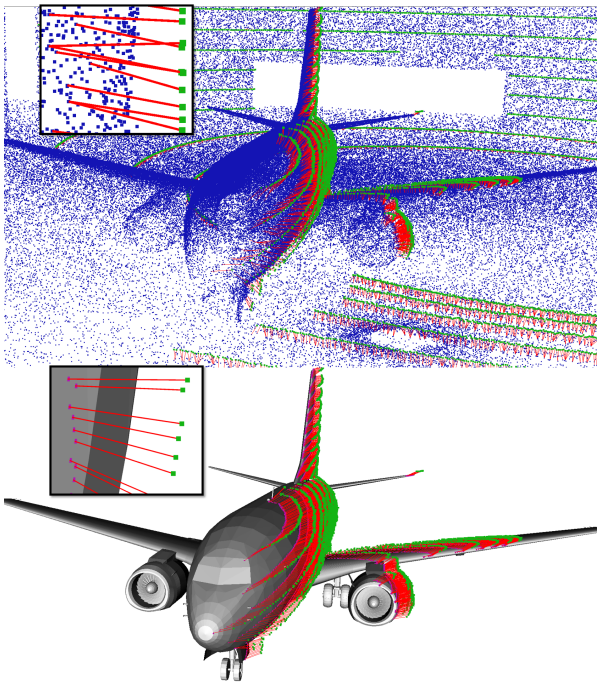


Fig. 1: Scan alignment for LiDAR-based 6-DoF pose estimation using point-to-point correspondences between the map (blue) and the current scan point cloud (green) in the LOAM framework (upper image). Our extension can be seen in the lower image using additional point-to-mesh correspondences between the triangular mesh surface and the current scan point cloud. In both images, the feature correspondences are depicted as red lines.

We extend the well-known LOAM framework [1], which uses point features extracted from the raw LiDAR scans, by additionally incorporating features from the triangular mesh of the reference object in the optimization problem. We term our approach Reference-LOAM (R-LOAM). Mesh features are virtual points sampled on the mesh surface, which are closest to the points of the current scan. These point-to-mesh feature correspondences are then inserted into an additional cost function, which we add to the conventional map optimization step of LOAM.

While the proposed approach may be used for any environment with proximity to a known object, in this paper we take as an example three simulated visual inspection scenarios using a UAV. The first two scenarios follow a trajectory around an airplane inside a hangar, once using the

airplane itself as the reference object and once using a van as a smaller reference object. The third scenario is outdoors, following a trajectory around the lower part of the Eiffel Tower as an alternative large reference object. Furthermore, we assume that the exact models of the reference objects are known and available as a 3D CAD model, e.g., as a triangular mesh file. In this work, we assume that the exact position of the reference objects in a global coordinate frame relative to the UAV is known or can be estimated before takeoff. Figure 1 shows an example of point-to-point correspondences (top) and point-to-mesh correspondences (bottom), which are both used in our LOAM extension. We include the latter in the map optimization step of the LOAM algorithm, yielding a higher accuracy in the localization and an improved 3D reconstruction of the environment.

The contributions of this paper can be summarized as follows:

1) A joint optimization problem is proposed using mesh features of a known object together with point features for LOAM.
2) The existing LOAM algorithm [1] is extended by incorporating an additional cost function term for mesh features.
3) Results of three simulated scenarios using a Velodyne VLP-16 and an Ouster OS1-128 LiDAR sensor, indoors around an airplane and outdoors at the Eiffel Tower, using our joint optimization approach are presented and compared with the ground-truth data from the simulation, and with the state-of-the-art LOAM.

The rest of this paper is structured as follows: We first elaborate on related LiDAR-SLAM systems and previous works using prior information to improve localization accuracy in Section II. In Section III we first review the original LOAM framework and then present our improved LOAM architecture in detail. In Section IV we give details about our simulated dataset and implementation. In Section V we show the results of our improved LOAM framework before discussing the limitations of our approach in Section VI. Section VII concludes this paper.

II. Related Work

A. LiDAR-SLAM

Many researchers have focused recently on improving the accuracy of LiDAR-based localization algorithms in GPS-denied environments. Traditionally, the Iterative Closest Point (ICP) algorithm has been used for localization with laser rangefinders. An important progress was achieved by Hess et al. [2] with the Cartographer approach for 2D LiDAR-SLAM in the year 2016. The software is one of the first open-source frameworks to perform high-accuracy real-time localization with loop closure for 2D LiDAR sensors and is still widely used. Kumar et al. [3] developed a framework using two low-cost rangefinders and the fusion with an Inertial Measurement Unit (IMU) to compute the 6-DoF position of a UAV during flight operations. The authors use a scan matching method in polar coordinates for the primary LiDAR and line segment extraction for the secondary LiDAR for position estimation. A similar approach was followed by Opromolla et al. [4] who used line features for scan matching.

With decreasing prices and advancing technologies, there was a transition to 3D LiDAR sensors, while still maintaining real-time capability and high precision for the localization. Zhang et al. [1] proposed LOAM, a real-time capable algorithm using a 3D LiDAR sensor. It uses point features to compute the odometry between consecutive scans and creates a 3D point cloud map for localization. However, it does not perform loop closure to increase accuracy when visiting previously seen areas of the environment. Its source code is publicly available and can still be considered as state-of-the-art. Droeschel et al. [5] proposed a continuous-time SLAM algorithm, which uses local maps and a hierarchical pose graph to refine scan alignments. IMLS-SLAM [6] aims at reducing the drift by using an Implicit Moving Least Squares (IMLS) surface representation for scan-to-model matching. However, as of today[1] the traditional LOAM algorithm [1] still performs best among the pure 3D LiDAR-based localization methods on the KITTI benchmark [7].

Also, various SLAM systems, which use a combination of LiDAR and cameras to compute the 6-DoF pose, have been proposed [8]–[10]. These methods try to combine the strengths of both sensor types to overcome their individual limitations. E.g. while cameras have a clear disadvantage in difficult lighting conditions, LiDAR sensors are mostly unaffected. On the other hand, cameras can help to get a better scene understanding in the FOV due to the higher information (pixel) density compared to a 360° 3D LiDAR. We refer the interested reader to the review of visual-LiDAR SLAM systems by Debeunne et al. [11].

B. Prior Information for Pose Estimation

A common method to improve a built map is to perform a loop closure with a pose graph optimization when visiting a previously visited area. However, prior information about parts of the environment can also be leveraged to improve the accuracy of pose estimation. Most related works in this field make use of pre-existing maps of the environments, known landmarks or floor plans. Parsley et al. [12] argue that prior information may not necessarily be a known map, but can also be modeled as constraints between landmarks in a SLAM system. Other works make use of aerial images in order to improve outdoor positioning [13]–[16]. Outdoors GPS signals can be used as the primary source for absolute positioning with the support of prior information in case of a GPS failure, while indoors other sensors such as LiDAR or camera have to be used. In the area of autonomous driving, methods for relative localization of the ego vehicle to a vehicle driving in front using LiDAR data and a geometric car model have been proposed [17], [18].

Shan et al. [19] leverage the existence of a ground plane for LiDAR-based localization (LeGO-LOAM). They perform

[1]March 2021

a scan segmentation followed by planar and edge feature extraction. These features are then used to estimate the odometry and build a map. The resulting transforms from odometry and mapping are then integrated to retrieve the final pose estimate. Other works use architectural floor plans as prior information using either RGB-D cameras [20], additionally WiFi [21] or LiDAR sensors. Boniardi et al. [22] have used a CAD-based architectural floor plan for 2D LiDAR-based localization. They proposed to use a pose graph SLAM system with Generalized ICP (GICP) for scan matching. The CAD floor plan is first converted to a 2D binary image and afterwards augmented with the resulting map for robot localization in a static factory-like scenario. Subsequently, they improved their framework by also enabling long-term localization in dynamic environments [23]. A similar approach was followed by Mielle et al. [24]. They merged the sensor map with an emergency map by feature matching to improve SLAM performance. Herbers et al. [25] used a Microsoft HoloLens as an augmented reality device to acquire point clouds and compute the current position inside a building by matching them to a 2D distance map of the building via template matching.

Few works were presented that leverage 3D prior information. Sandy et al. [26] developed an In situ Fabricator, which is an autonomous platform with a 6-DoF robotic arm and an end-effector to perform simple tasks in a construction site. The platform and end-effector are equipped with LiDAR sensors. For the localization on the construction site, they use 2D LiDAR and IMU measurements. Once the platform reaches the place of operation, it performs a scan of the workpiece while being stationary by moving the end-effector with the laser rangefinder. The position relative to the workpiece is then estimated by matching the acquired scan to the 3D model of the workpiece and construction site, which consist of planes or multiple simple primitives, i.e. polygons. Gawel et al. [27] developed a similar robotic platform. They perform ray-tracing into the 3D CAD building model to achieve high positioning accuracy for their end-effector.

In this paper, we consider the presence of a *complex* object represented as a 3D triangular mesh consisting of up to hundreds of thousands of triangles and vertices. In contrast to previous works, we do not assume the robot to be static, but in constant motion during scan acquisition. For each 360° LiDAR scan we compute the 6-DoF pose using a novel joint optimization formulation incorporating point and mesh features by extending the well-known LOAM framework [1].

III. Methodology

In this section, we first explain the basic structure of the conventional LOAM pipeline and then give an overview of our proposed extension. Figure 2 shows the conventional LOAM pipeline (blue) and our extension (green). Only the optimization part of the conventional LOAM pipeline inside the mapping module has been modified to a joint optimization. The other green boxes are extensions of the mapping process of LOAM, reducing the changes done to the original LOAM algorithm to a minimum.

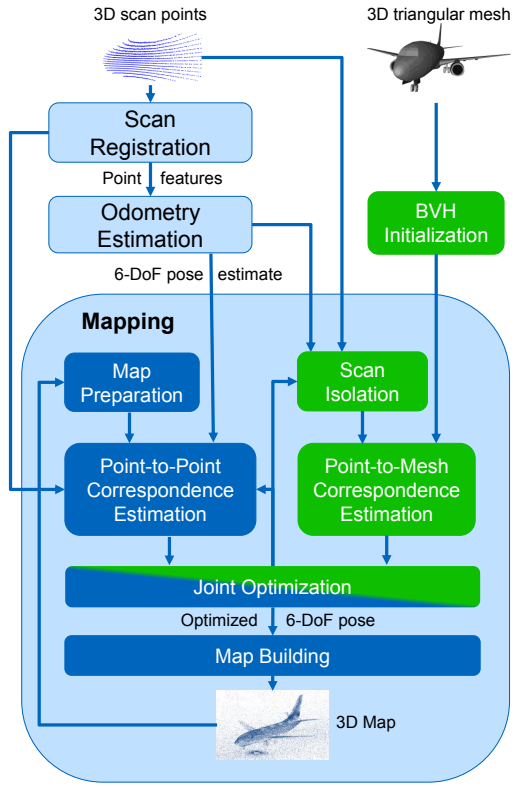


Fig. 2: System overview of the extended LOAM framework. Light blue boxes are the modules of the state-of-the-art LOAM algorithm, and dark blue are the sub-modules inside the mapping node. Our extension is illustrated by the green boxes. In addition to the traditional point features, mesh features are used in the proposed joint optimization formulation.

A. State-of-the-art LOAM

The LOAM algorithm [1] consists of three steps: Scan registration, odometry estimation, and mapping. In the scan registration module, the algorithm extracts point features from the raw scan points. Point features are either corner points or surface points. After the feature extraction, the points are sent to the odometry module where the relative transform, i.e., relative pose change to the previous scan is computed by feature matching and an optimization step. The odometry estimation yields a 6-DoF pose estimate, which is used as an initial pose estimate for the mapping module. The first step inside the mapping module is the map preparation. Here the LOAM algorithm sorts and stores the feature points of the map in a cube structure, which is used to find feature correspondences between the features of the current scan points and the features in the map. We denote corner correspondences as $c^{\mathcal{C}}$ and surface correspondences as $c^{\mathcal{S}}$ with $c = \{p_1, p_2\}$, where p_1 is a point of the current scan and p_2 is a point of the map, respectively. Once the point-to-point correspondences are found, residual blocks can be formed using the corresponding cost functions for corner and surface features, $f^{\mathcal{C}}$ and $f^{\mathcal{S}}$, respectively. The outputs of the cost functions are squared residuals, which are

weighted by the loss function ρ. Residuals are the Euclidean distances between the points forming the correspondence. Residual blocks are the cost functions wrapped inside a loss function. Finally, the resulting residual blocks are added to the optimization function J. The state-of-the-art map optimization problem of LOAM is defined as

$$J(q,t) = \sum_{c^{\mathcal{S}} \in C^{\mathcal{S}}} \rho(\|f^{\mathcal{S}}(c^{\mathcal{S}}, q, t)\|^2) + \sum_{c^{\mathcal{C}} \in C^{\mathcal{C}}} \rho(\|f^{\mathcal{C}}(c^{\mathcal{C}}, q, t)\|^2) \quad (1)$$

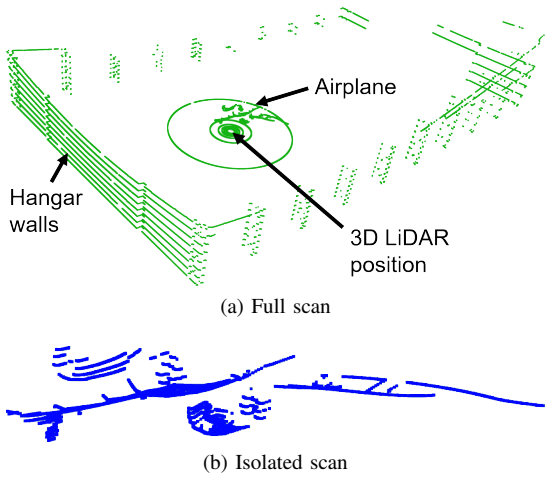
where $C^{\mathcal{C}}$ and $C^{\mathcal{S}}$ are the sets of point feature correspondences for corner and surface features, respectively. $C^{\mathcal{C}}$ is defined as $C^{\mathcal{C}} = \{c_1^{\mathcal{C}}, c_2^{\mathcal{C}}, ..., c_n^{\mathcal{C}}\}$ and $C^{\mathcal{S}}$ is defined as $C^{\mathcal{S}} = \{c_1^{\mathcal{S}}, c_2^{\mathcal{S}}, ..., c_m^{\mathcal{S}}\}$ with n and m being the number of corner and surface feature correspondences, respectively. q is the quaternion and t is the translation vector, which transform the feature points of the scan from the LiDAR frame to the map frame during the optimization iterations. $||.||^2$ denotes the squared Euclidean distance, which is used as residual. With the optimized transform, a second iteration of point-to-point correspondence estimation with a subsequent optimization step is performed, which is the default in the conventional LOAM algorithm. After the two iterations, the feature points of the current scan are inserted into the map using the final optimized transform, which is illustrated by the map building module in Figure 2. Subsequently, the updated map can be used to estimate the 6-DoF pose for the next LiDAR scan.

B. Improved LOAM using Mesh and Point Features

In this section, we give an overview of our improved LOAM pipeline using mesh and point features. Furthermore, we explain the extension modules shown in green in Figure 2 in detail.

In the following, we assume that the model of the reference object is available as a 3D triangular mesh. In addition, we assume that its location is known. First, the triangular mesh is transformed from its local frame to the map frame using its known location. Note that the local frame is the reference system for the vertices of the mesh. In order to speed up mesh feature computation later on, we borrow a space division technique from computer graphics. AABB-Trees are a common method to perform real-time collision detection between 3D objects. Once transformed into the map frame, we initialize the AABB-Tree, a hierarchical balanced binary tree with each leaf node containing one face of the mesh. Using an AABB-Tree is optional and can be replaced by any other Bounding Volume Hierarchy (BVH) method. However, using a BVH is highly recommended since it significantly speeds up mesh feature extraction.

1) Scan Isolation: Incoming raw point clouds are processed in the scan isolation module as illustrated in Figure 2. Here the raw scan points are first transformed from the LiDAR frame to the map frame using the initial pose estimate from the odometry module. Secondly, the scan points belonging to the reference object need to be isolated from the rest of the points. For this we use a simple bounding box cropping with the size of the object plus some additional buffer, i.e. 2 m larger than the object's bounding box. This step assumes that no other objects are located in the area of the bounding box of the object. Other cropping techniques may also be used, however, it shall be noted that tighter isolation requires a good initial LiDAR pose estimate and is a compromise between removing too many or too few points. Ideally, only scan points belonging to the reference object will remain after scan isolation. Figure 3 illustrates the scan cropping process from the raw full scan point cloud (a) to the isolated point cloud (b).



(a) Full scan

(b) Isolated scan

Fig. 3: Examples of a full 360° 3D LiDAR scan (a) and an isolated scan with only scan points belonging to the airplane surface (b).

2) Mesh Feature Extraction: With the help of the AABB-Tree, point-to-mesh correspondences can be efficiently computed between the isolated scan points and the triangular mesh. We define a point-to-mesh correspondence between a point of the current isolated scan and a virtual point on the mesh surface. The virtual point on the mesh surface we consider as a mesh feature. For each scan point passing the scan isolation process, we compute a virtual point on the triangular mesh surface, which is closest to the corresponding scan point. The surface is consisting of faces, which are implicitly defined by the enclosing vertices. Particularly we are looking for the triangle of the mesh with a virtual point on the plane of the triangle which has the smallest Euclidean distance to the scan point. For each scan point, we first need to find the triangle containing the closest virtual point. Note that the naïve method is a brute-force approach, which is intractable due to extremely high computational complexity if the number of faces of a mesh is high, i.e. over 200k faces. Leveraging the axes parallelism, traversing through the AABB-Tree is very efficient since only the extreme points defining the bounding boxes need to be considered in order to find the closest triangle. Once found, we can use basic

3D geometry to find the distance and the virtual point on the plane of the triangle which is closest to the scan point. Computing the virtual points on the mesh surface for all scan points yields all the required point-to-mesh correspondences.

3) Joint Optimization: Coming back to our pipeline in Figure 2, we also create one residual block for each point-to-mesh correspondence using the Huber loss function and a cost function $f^{\mathcal{M}}$, which is defined as

$$f^{\mathcal{M}}(c^{\mathcal{M}}, q, t) = (qp_1q^{-1} + t) - p_2 \text{ , with } p_1, p_2 \in c^{\mathcal{M}} \quad (2)$$

where p_1 is the scan point of the point-to-mesh correspondence $c^{\mathcal{M}}$, which is transformed from the LiDAR frame to the map frame using q and t. p_2 is the virtual point on the mesh in the map frame, which is closest to the scan point. We add normalization terms for point features in Eq. 1 and extend it by the cost function defined in Eq. 2. The normalized joint optimization problem can then be defined as

$$\begin{aligned} J(q,t) = &\frac{1}{\bar{\bar{C}}^{\mathcal{S}}} \sum_{c^{\mathcal{S}} \in C^{\mathcal{S}}} \rho(\|f^{\mathcal{S}}(c^{\mathcal{S}}, q, t)\|^2) \\ &+ \frac{1}{\bar{\bar{C}}^{\mathcal{C}}} \sum_{c^{\mathcal{C}} \in C^{\mathcal{C}}} \rho(\|f^{\mathcal{C}}(c^{\mathcal{C}}, q, t)\|^2) \\ &+ \frac{\lambda}{\bar{\bar{C}}^{\mathcal{M}}} \sum_{c^{\mathcal{M}} \in C^{\mathcal{M}}} \rho(\|f^{\mathcal{M}}(c^{\mathcal{M}}, q, t)\|^2) \text{ ,} \end{aligned} \quad (3)$$

where $\bar{\bar{C}}^{\mathcal{S}}$, $\bar{\bar{C}}^{\mathcal{C}}$ and $\bar{\bar{C}}^{\mathcal{M}}$ denote the cardinality of the surface, corner and point-to-mesh correspondence sets. λ is the Lagrange multiplier, which enables us to control the influence of point-to-mesh residuals in contrast to the point-to-point residuals. $\lambda = 1$ gives an equal weight to corner, surface and mesh features, whereas $\lambda = 2$ gives an equal weight for point and mesh features. If less than 100 mesh features are extracted we set $\bar{\bar{C}}^{\mathcal{M}} = 100$ so that the mesh residuals do not get too much weight in the optimization problem. Just as for the original LOAM implementation, we run at least two iterations of correspondence estimation and joint optimization before retrieving the final optimized 6-DoF pose. Intuitively, the joint optimization problem minimizes the residuals between the scan points and the map points, but also the residuals of the isolated scan points to the mesh surface. This effectively aligns the scan to the mesh surface, giving an absolute reference in the current environment and reducing the long-term drift, while also taking the built map into account. The map building process remains unchanged by our extension.

IV. Experimental Setup

A. Dataset

We use the Gazebo simulation environment[2] to generate 3D LiDAR data. The simulation also provides the ground-truth 6-DoF pose of the LiDAR sensor in the global coordinate frame. We simulate two 3D LiDAR sensors following the technical specifications of a Velodyne VLP-16 and an Ouster OS1-128. For the VLP-16 we add zero-mean Gaussian noise with $\sigma = 0.03$ and for the OS1-128 with $\sigma = 0.05$. The VLP-16 provides up to 30 000 points per scan, distributed on 16 scan lines, while the OS1-128 provides up to 262 144 points per scan, distributed on 128 scan lines. Both operate at 10 Hz. One 3D LiDAR sensor is mounted at a time on a 1-DoF gimbal carried by a quadcopter UAV. The gimbal allows us to rotate the LiDAR sensor during the flight to capture a wider area, even when the drone is static. Figure 4 shows the quadcopter with the LiDAR tilting range from -0.6 rad (a) to $+0.6$ rad (b). Throughout the flight, the LiDAR is constantly actuated back and forth in this range.

[2] http://gazebosim.org/

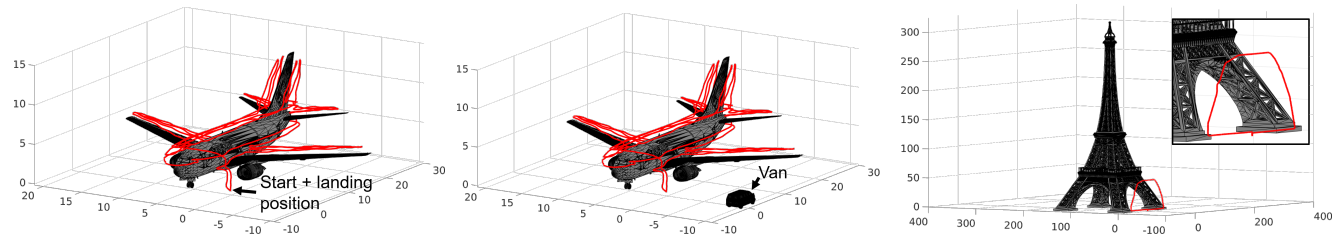
(a) -0.6 rad ($\approx -34.38°$) (b) $+0.6$ rad ($\approx +34.38°$)

Fig. 4: Visualization of the tilting range of the actuated 3D LiDAR sensor mounted on a 1-DoF gimbal on a quadcopter.

TABLE I: Metadata for the generated datasets. For each scenario, two datasets were generated. One using a VLP-16 and one using an OS1-128 LiDAR sensor.

Dataset	Scenario	LiDAR	#Scans	Avg. vel.	length
1	1	VLP-16	15111	0.35 m/s	514 m
2	1	OS1-128	9905	0.48 m/s	474 m
3	2	VLP-16	9718	0.49 m/s	474 m
4	2	OS1-128	9726	0.49 m/s	474 m
5	3	VLP-16	5674	0.51 m/s	291 m
6	3	OS1-128	5762	0.51 m/s	291 m

We created three scenarios as illustrated in Figure 5. In the scenarios 1 and 2 the drone follows a trajectory inside a hangar (37.6 x 150 x 90 meters) around the upper part of the airplane. In scenario 1 we use the airplane of type B737 (13 x 31 x 30.8 meters) as a known 3D model and in scenario 2 we use a van (1.8 x 4.8 x 2.3 meters) as a known object. Scenario 1 shows a manually flown trajectory and scenario 2 shows the same trajectory, executed by an autonomous flight controller. Figure 5 (c) shows the trajectory at the lower part of the Eiffel Tower (324 x 146 x 146 meters). Here we used the Eiffel Tower itself as a known reference object for R-LOAM. In total, six datasets were generated, two for each scenario using a VLP-16 and OS1-128. Table I shows an overview of all the datasets. In dataset 1 a trajectory was flown with manual control, while the datasets 2-4 followed this trajectory with an autonomous flight controller. The maximum velocity for the autonomous control was set to 0.5 m/s. Due to a more efficient control, the total flight time and therefore number of scans decreased for the datasets 2-4. It shall be noted that due to varying LiDAR rotations from the gimbal, the LiDAR data is never the same, despite an almost identical trajectory.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 5: Illustration of the flight trajectory of the quadcopter for three scenarios. (a) uses the airplane as reference, (b) uses the van as reference, (c) uses the Eiffel Tower as reference.

B. Implementation

We use the open-source Advanced LOAM implementation (A-LOAM[3]) and extend it with our approach to perform a joint optimization incorporating point and mesh features. A-LOAM is split into three Robot Operating System (ROS) nodes for scan registration, odometry and mapping for which we use the ROS melodic framework[4]. We incorporate all extensions into the mapping node. A-LOAM has the functionality to abort map optimization when a new scan arrives before the previous optimization has finished in order to ensure real-time capability. Here only the pose estimate of the odometry module is used as final pose. We disabled this functionality to enforce an optimized pose from the mapping module for each scan for all of our experiments.

For the scan isolation we crop the scans with a bounding box, which has the size of the used 3D model plus an additional buffer of $2\ m$ ($200\ m$) in the X, Y and Z directions for scenarios 1+2 (scenario 3) with respect to a right-handed coordinate system. We cut off at $1\ m$ height for scenario 1 and $0.5\ m$ height for scenarios 2+3 in order to remove the ground-plane. The additional buffer reduces the number of accidentally cropped points of the object surface due to wrong initial pose estimates from the odometry module and previous map optimizations. In order to find the virtual point and closest distance between the query scan point and a triangle on the mesh, we make use of the IGL library [28].

For the joint optimization we use the Ceres Solver [29] optimization framework and also use the Levenberg-Marquardt trust region algorithm, which is the same as in the original A-LOAM implementation. For all the remaining parameters, we use the default configuration of LOAM. Additionally, we adapted the point feature extraction in the scan registration module to support LiDAR scans from the Ouster OS1-128 sensor.

We found that the weight λ in Eq. 3 for mesh features should be low in the first iteration and continuously increase with higher iterations. We chose a logarithmic increase from 1 to 35 iterations for λ. Hence, we compute the log_{10} function in the range 1 to 35 and then scale the resulting log-values to the range $0.1 - 40$ to achieve a log-shaped curve for the first 35 iterations. E.g. in iteration 1 we use 0.1 and at iteration 35 we use 40 for λ. The values have been empirically determined by conducting several experiments using scaled sigmoid, linear and log functions.

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

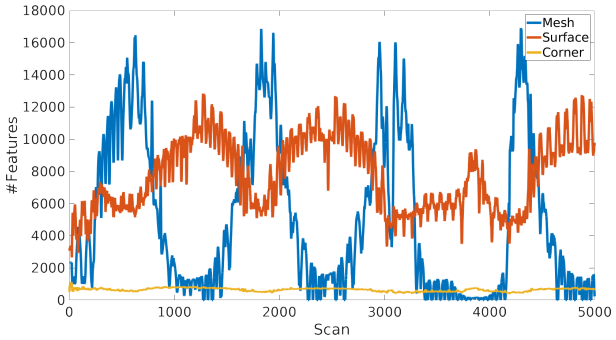


Fig. 6: Illustration of the number of mesh, surface and corner features for the first $5\,000$ LiDAR scans of dataset 1.

V. Results

First, we discuss an example of the varying number of detected features depending on the environment. Figure 6 shows the number of mesh, surface and corner features for the first $5\,000$ LiDAR scans of dataset 1. In every scan, a similar amount of corner features were detected. However, the number of corner features is low compared to mesh and surface features. The high frequency variations for the mesh and surface features are due to the actuation of the LiDAR sensor. More or fewer parts of the airplane and hangar walls are in the LiDAR's FOV depending on the rotation. Sometimes no mesh features are detected. In this case only the first two terms in Eq. 3 are effective, using only point features. The same features as in conventional LOAM are used, but with normalized residuals in the optimization.

We compare the Absolute Pose Error (APE) and Rotational Error (RE), with the ground-truth for the state-of-the-art LOAM (Eq. 1) and our extended LOAM using mesh features (Eq. 3). For each method we conducted between 2 (default) and 35 correspondence estimation and optimization iterations.

Table II shows the results of our performance evaluation for scenario 1 using a VLP-16 and OS1-128. In this scenario, we use the airplane itself as a 3D reference object. For the

TABLE II: Experimental results for scenario 1 with varying number of iterations. The best results are marked in **bold**.

	Method	#Iter	APE in cm			RE in deg		
			max	mean	median	max	mean	median
VLP-16 (Dataset 1)	LOAM [1]	2 (def)	102.4	55.8	50.5	2.98	2.19	2.20
		5	116.9	64.7	57.7	3.13	2.55	2.56
		15	101.0	55.5	51.1	3.14	2.11	2.12
		25	69.6	35.9	30.9	3.14	1.58	1.59
		35	71.2	36.8	31.9	3.14	1.61	1.61
	R-LOAM	2 (def)	26.7	11.6	10.5	1.36	0.41	0.40
		5	22.6	6.9	6.6	1.83	0.32	0.31
		15	19.5	3.3	3.1	1.67	0.26	0.25
		25	**17.7**	3.0	2.8	1.38	0.11	0.11
		35	19.4	**2.1**	**2.0**	**1.17**	**0.09**	**0.09**
OS1-128 (Dataset 2)	LOAM [1]	2 (def)	20.8	12.7	13.2	0.29	0.10	0.09
		5	20.9	13.0	13.5	0.29	0.09	0.08
		15	20.9	13.0	13.5	0.29	0.09	0.08
		25	20.9	13.0	13.5	0.29	0.09	0.08
		35	20.9	13.0	13.5	0.29	0.09	0.08
	R-LOAM	2 (def)	18.3	2.9	2.6	0.44	0.12	0.13
		5	13.8	2.1	1.9	0.32	0.09	0.09
		15	10.0	1.6	1.4	0.29	0.08	0.08
		25	7.8	1.4	1.3	0.29	**0.07**	**0.07**
		35	**6.3**	**1.3**	**1.2**	**0.28**	**0.07**	**0.07**

TABLE III: Experimental results for scenario 2 with varying number of iterations. The best results are marked in **bold**.

	Method	#Iter	APE in cm			RE in deg		
			max	mean	median	max	mean	median
VLP-16 (Dataset 3)	LOAM [1]	2 (def)	37.7	22.2	19.0	1.46	0.84	0.83
		5	136.7	72.7	65.0	3.24	3.08	3.08
		15	60.5	28.9	24.0	3.52	1.65	1.63
		25	59.9	28.6	23.8	3.52	1.63	1.62
		35	60.4	28.9	23.9	3.52	1.64	1.62
	R-LOAM	2 (def)	39.7	20.0	17.9	**1.35**	0.82	0.81
		5	16.4	7.5	7.3	1.44	0.44	0.44
		15	**15.3**	**7.0**	**6.3**	1.42	**0.37**	**0.37**
		25	23.5	13.2	13.5	1.41	0.38	**0.37**
		35	23.6	13.2	13.5	1.41	**0.37**	**0.37**
OS1-128 (Dataset 4)	LOAM [1]	2 (def)	23.9	14.5	14.8	0.46	0.08	0.07
		5	24.2	15.0	15.3	0.41	0.08	0.06
		15	24.1	15.1	15.4	0.41	0.07	0.06
		25	24.1	15.1	15.4	0.41	0.07	0.06
		35	24.1	15.1	15.4	0.41	0.07	0.06
	R-LOAM	2 (def)	32.6	6.3	6.3	0.73	0.10	0.09
		5	15.2	5.2	4.9	0.40	0.07	0.06
		15	**12.3**	4.8	4.3	**0.39**	**0.06**	**0.05**
		25	13.5	**4.7**	**4.2**	**0.39**	**0.06**	**0.05**
		35	14.2	**4.7**	**4.2**	**0.39**	**0.06**	**0.05**

VLP-16, the lowest APE using the state-of-the-art LOAM can be seen at around 25 iterations with a maximum value of 70 cm and a median of 31 cm. A higher number of iterations did not improve the APE further. R-LOAM shows the lowest median APE at 35 iterations with 2 cm. Also, the RE is lower with 0.09 deg compared to the lowest RE of LOAM with 1.59 deg. Interestingly, additional iterations for LOAM using the OS1-128 did not result in further improvements. The lowest median APE is at the default of 2 iterations with a median APE of 13.2 cm. However, for R-LOAM the error further reduces until 35 iterations to a median APE of 1.2 cm.

Table III shows the results of our performance evaluation for scenario 2. For R-LOAM we only used the van as known 3D reference model, resulting in significantly fewer mesh features. The van is only visible in the LiDAR scans with specific gimbal rotations and is completely occluded when the UAV is on the other side of the airplane. The lowest median APE of LOAM using a VLP-16 sensor can be seen at 2 iterations with 19 cm and for R-LOAM at 15 iterations with 6 cm median APE. The median APE for LOAM amounts to 15 cm at 2 iterations using the OS1-128. Our R-LOAM approach peaks at 25 iterations with a bit more than 4 cm median APE and an RE of 0.05 deg. Using an OS1-128 improved the RE in both scenarios for LOAM and R-LOAM. These results show that the more scan points are detected on the 3D reference object, the lower is the pose error of our R-LOAM approach. An analysis showed that in dataset 3 only 22% of the scans have more than 100 mesh features, in contrast to 45% in dataset 4 due to a higher point density in each scan. Despite the low visibility, R-LOAM improved the pose error in both datasets.

TABLE IV: Experimental results for scenario 3 with varying number of iterations. The best results are marked in **bold**.

	Method	#Iter	APE in cm			RE in deg		
			max	mean	median	max	mean	median
VLP-16 (Dataset 5)	LOAM [1]	2 (def)	1512.2	495.0	499.3	10.09	3.66	2.47
		5	443.4	327.0	380.2	5.35	4.38	4.08
		15	1797.5	1244.7	1420.9	17.56	17.25	17.28
		25	1696.2	1166.6	1326.4	16.61	16.36	16.40
		35	1800.5	1218.6	1338.5	18.42	18.17	18.21
	R-LOAM	2 (def)	1869.5	226.0	31.5	17.01	2.73	0.29
		5	88.2	5.1	2.4	1.60	0.07	0.04
		15	72.9	1.2	0.5	**0.32**	**0.02**	0.02
		25	61.5	0.7	**0.3**	0.40	**0.02**	0.02
		35	**48.9**	**0.5**	**0.3**	0.34	**0.02**	**0.01**
OS1-128 (Dataset 6)	LOAM [1]	2 (def)	2472.0	1109.7	1058.2	12.77	6.12	4.97
		5	1063.1	218.2	105.0	9.23	2.70	1.49
		15	47.8	10.2	9.3	1.01	0.14	0.14
		25	45.7	10.8	10.5	0.94	0.12	0.10
		35	39.9	11.5	11.3	0.40	0.11	0.11
	R-LOAM	2 (def)	94.1	11.1	6.4	1.42	0.19	0.12
		5	42.9	3.5	2.4	0.69	0.10	0.08
		15	13.1	1.6	1.2	**0.25**	0.08	**0.07**
		25	8.5	1.3	1.1	**0.25**	0.08	**0.07**
		35	**7.3**	**1.2**	**1.0**	**0.25**	**0.07**	**0.07**

Table IV shows the results of our performance evaluation for scenario 3. LOAM fails in this scenario with a VLP-16, while R-LOAM achieves a sub-centimeter median APE at 35 iterations. Due to the structure of the Eiffel Tower, it seems that the extracted point features are insufficient to reliably estimate the pose, while the addition of mesh features give a significant improvement. With at least 15 iterations LOAM also achieves a high accuracy in dataset 6 with an OS1-128. For this scenario too, R-LOAM outperforms LOAM by a large margin.

To summarize our results, with our joint optimization formulation we are able to achieve in our simulation-based experiments an average reduction in median APE of over 92% for scenario 1, over 69% for scenario 2 and over 94% for scenario 3 compared to conventional LOAM. Apart from dataset 3, R-LOAM significantly benefits from an increased number of map optimization iterations, e.g. 15-35. Also, mesh features are a valuable addition to point features as seen in dataset 5, where LOAM fails. The OS1-128 proved to be superior and beneficial for the localization accuracy due to the higher number of scan points (see datasets 5+6).

Instead of using a mesh, a reference object converted

to a point cloud may also be used. However, a sparsely sampled model will decrease accuracy and an extremely densely sampled object might have a higher correspondence estimation complexity than the estimation of point-to-mesh correspondences.

VI. Limitations

In our experiments we assume a perfect 3D mesh of the reference object and the knowledge of its exact location in the map frame. Any imperfections in the mesh which deviate from reality will have a negative influence on R-LOAM. In practice, devices from Building Information Modeling (BIM) could generate a 3D model with sub-millimeter accuracy. Also, any error in the pose estimation of the reference object in the map frame will have a direct impact on R-LOAM. This means that a large error might cause a worse performance of R-LOAM than LOAM, since the point-to-mesh formulation directly depends on the relative localization to the reference object. Hence, a low error in the 3D mesh, and reference object localization, is crucial for R-LOAM.

VII. Conclusion

In this paper we proposed an extension to the existing LOAM algorithm [1] in order to improve the localization accuracy by leveraging prior knowledge about a reference object in the environment. We extracted mesh features from a 3D triangular mesh in addition to the traditional point features from a 3D LiDAR scan. Point-to-mesh correspondences were then added to the novel joint optimization problem using an additional cost function. We simulated three scenarios of visual inspection scenarios, indoors and outdoors, using a VLP-16 and an OS1-128. We evaluated our approach by means of APE and RE for a varying number of iterations. Results showed that our R-LOAM approach using mesh features showed a reduction of median APE in all three scenarios, even when using a small reference object (e.g. van), resulting in an improved 3D reconstruction.

References

[1] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[2] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 1271–1278, 2016.

[3] 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*, 2017.

[4] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, and A. Savvaris, "LIDAR-inertial integration for UAV localization and mapping in complex environments," *2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016*, pp. 649–656, 2016.

[5] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," *Proceedings - IEEE International Conference on Robotics and Automation*, no. May, pp. 5000–5007, 2018.

[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] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," *Proceedings - IEEE International Conference on Robotics and Automation*, 2015.

[9] J. Graeter, A. Wilczynski, and M. Lauer, "LIMO: Lidar-Monocular Visual Odometry," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7872–7879, 2018.

[10] Y. Seo and C. C. Chou, "A tight coupling of vision-lidar measurements for an effective odometry," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2019-June, no. Iv, pp. 1118–1123, 2019.

[11] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors*, 2020.

[12] M. P. Parsley and S. J. Julier, "Exploiting prior information in Graph-SLAM," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2638–2643, 2011.

[13] T. Korah and C. Rasmussen, "Probabilistic contour extraction with model-switching for vehicle localization," in *IEEE Intelligent Vehicles Symposium, 2004*, 2004, pp. 710–715.

[14] K. Y. K. Leung, C. M. Clark, and J. P. Huissoon, "Localization in urban environments by matching ground level video images with an aerial image," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 551–556.

[15] G. Conte and P. Doherty, "An integrated uav navigation system based on aerial image matching," 04 2008, pp. 1 – 10.

[16] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Autonomous Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[17] E. Héry, P. Xu, and P. Bonnifait, "Pose and covariance matrix propagation issues in cooperative localization with LiDAR perception," *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019.

[18] E. Héry, P. Xu, and P. Bonnifait, "Lidar based relative pose and covariance estimation for communicating vehicles exchanging a polygonal model of their shape," 10 2018.

[19] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," *IEEE International Conference on Intelligent Robots and Systems*, 2018.

[20] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, and W. Burgard, "Accurate indoor localization for RGB-D smartphones and tablets given 2D floor plans," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 3138–3143, 2015.

[21] S. Ito, F. Endres, M. Kuderer, G. Diego Tipaldi, C. Stachniss, and W. Burgard, "W-RGB-D: Floor-plan-based indoor global localization using a depth camera and WiFi," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 417–422, 2014.

[22] F. Boniardi, T. Caselitz, R. Kummerle, and W. Burgard, "Robust LiDAR-based localization in architectural floor plans," *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[23] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "A pose graph-based localization system for long-term navigation in CAD floor plans," *Robotics and Autonomous Systems*, vol. 112, pp. 84–97, 2019.

[24] 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. [Online]. Available: https://www.mdpi.com/2218-6581/8/2/40

[25] P. Herbers and M. König, "Indoor Localization for Augmented Reality Devices Using BIM , Point Clouds , and Template Matching," 2019.

[26] T. Sandy, M. Giftthaler, K. Dorfler, M. Kohler, and J. Buchli, "Autonomous repositioning and localization of an in situ fabricator," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 2852–2858, 2016.

[27] A. Gawel, H. Blum, J. Pankert, K. Krmer, L. Bartolomei, S. Ercan, F. Farshidian, M. Chli, F. Gramazio, R. Siegwart, M. Hutter, and T. Sandy, "A fully-integrated sensing and control system for high-accuracy mobile robotic building construction," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2300–2307.

[28] A. Jacobson, D. Panozzo *et al.*, "libigl: A simple C++ geometry processing library," https://libigl.github.io/, 2018, (accessed Mar. 17, 2021).

[29] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org, (accessed Mar. 17, 2021).