

Lidar Positioning for Indoor Precision Navigation

Max Holmberg, Oskar Karlsson*, Michael Tulldahl
Swedish Defence Research Agency (FOI), Linköping, Sweden

Abstract

Lidar based simultaneous localization and mapping methods can be adapted for deployment on small autonomous vehicles operating in unmapped indoor environments. For this purpose, we propose a method which combines inertial data, low-drift lidar odometry, planar primitives, and loop closing in a graph-based structure. The accuracy of our method is experimentally evaluated, using a high-resolution lidar, and compared to the state-of-the-art methods LIO-SAM and Cartographer. We specifically address the lateral positioning accuracy when passing through narrow openings, where high accuracy is a prerequisite for safe operation of autonomous vehicles. The test cases include doorways, slightly wider reference passages, and a larger corridor environment. We observe a reduced lateral accuracy for all three methods when passing through the narrow openings compared to operation in larger spaces. Compared to state-of-the-art, our method shows better results in the narrow passages, and comparable results in the other environments with reasonably low usage of CPU and memory resources.

1. Introduction

Autonomous navigation in indoor environments such as offices, warehouses, and industrial buildings is a challenging task. The problem is especially difficult when dealing with autonomous unmanned aerial or ground vehicles (UAVs and UGVs) in combination with narrow passages such as doorways and openings between large or small objects. This requires precision navigation which in turn requires accurate positioning relative to the environment. In this study we propose and evaluate a lidar SLAM (Simultaneous Localization And Mapping) method for application in such scenarios, where a known map is inexistent or an existing map or blueprint is too time consuming to adapt for the purpose of the navigation. Our method is based on data from inertial sensors and a high-resolution rotating time-of-flight (ToF) lidar.

Autonomous operation of UAVs or UGVs can be divided into the main tasks: path planning, navigation, and control [3]. Path planning considers the high-level task of defining waypoints to visit in the simultaneously created map. Autonomous path planning is a vast research area that is not within the scope of our present study. Navigation and control involve subtasks such as 3D path following, hover control, attitude control [28], obstacle avoidance, and autonomous takeoff and landing [21]. Within these subtasks, we specifically address the problem of precise vehicle positioning in the map, to enable safe navigation through narrow passages without hitting or bumping into any physical objects previously detected and located in the map building process. Safe passage through narrow openings can be obtained with accurate control algorithms, but these algorithms also require accurate positioning. Especially for UAVs, high accuracy is needed both in terms of absolute and relative positioning which are needed as feedback parameters into the attitude (roll, pitch, yaw) control [28]. The UAV ego positioning should also have a reasonably high update rate and low latency. Before addressing lidar positioning accuracy we will briefly discuss strategies for passing through narrow openings. We exemplify with a passage through a doorway from one room or a corridor into another room. This is the scenario evaluated experimentally in Section 4.1. Ideally, the path planner gives waypoints located on each side of the doorway such that the UAV, when following a straight line between these waypoints, has equal clearance on both sides. If the path planner waypoints are less accurate, an obstacle avoidance algorithm is needed to give intermediate waypoints [18] which results in sufficient clearance to both doorposts. Regardless of which algorithm generated the path or waypoints, an important challenge is the fact that a ToF lidar usually has a minimum range which makes the sensor blind to close objects, in our example the doorposts. Thus, during the time interval from an obstacle is last seen to the actual passage of the obstacle, the relative position accuracy is highly important. Another strategy for passing near obstacles is to pass with relatively high speed and thus reducing the time interval when the obstacles are unseen by the onboard sensors, which also requires highly accurate position (relative to obstacles) and speed feedback

*Corresponding author: oskar.karlsson@foi.se

to the controller. This was demonstrated in impressive aggressive UAV indoor flights using an external positioning system [28] and also using only onboard sensors (downward facing camera and IMU - inertial measurement unit) and computing [27] in indoor environments with known obstacles.

Our proposed SLAM method is a realtime, graph-based SLAM using plane and edge features from lidar data. It utilizes an IMU and plane features directly in the graph optimization. A precision lidar odometry step, using plane and edge features, is used to minimize drift over shorter time periods and loop closing over longer time periods.

In summary, this paper has the following key contributions: (1) A lidar SLAM method for accurate indoor localization is presented. The method is designed to run in realtime onboard a UAV or UGV with relatively low computing power. (2) Experimental data is collected specifically for evaluating the accuracy when passing through narrow openings. Data is collected with the sensor on a roller table to achieve reliable ground truth (GT) data in the lateral dimension. (3) With the experimental data, our SLAM method is evaluated and compared to the methods Cartographer [17] and LIO-SAM [34].

2. Related work

UAV and UGV positioning systems are usually equipped with an IMU and additional sensors for localization relative to external global or local references. Data from the IMU and additional sensors are fused *e.g.* with Extended Kalman Filters (EKF) or other algorithms such as factor graphs to obtain pose (position and orientation) estimates.

Indoor positioning implicitly requires other reference systems than Global Navigation Satellite Systems (GNSS). In addition to lidar, there exist several possible sensor types, which combined with algorithms, achieve such reference systems. Commonly used sensors are stereo cameras [1, 6, 9, 26, 30] or monocular cameras [1, 6, 11] combined with extraction of references, called features or landmarks, from the images. A downward facing camera can be used to estimate the vehicle speed from optical flow data extracted from the images [1, 2]. Apart from difficulties in low-textured scenes, a drawback with passive camera sensors are that they rely on ambient light and therefore are sensitive to lighting conditions and shadows, including shadowing by the own vehicle. Adaptation to low-light conditions can be made using onboard headlights, which however give limited imaging range [1]. Sonars or ultrasonic sensors can be used for localization [4, 35] and as proximity sensors for obstacle detection [16]. Ultra-wideband (UWB) positioning was demonstrated based on the principle of range measurement from one onboard tag to eight external anchors [36].

Concerning lidar based SLAM methods, a large body of

work exists and many efficient lidar SLAM algorithms have been proposed in recent years. Scan-to-scan, and scan-to-map matching methods have been popular. In [17] the background for an algorithm using scan-to-map matching with realtime loop closure was presented. The papers by Zhang and Sing [37, 38] regarding real time lidar odometry are still relevant and provided impressive results. Further improvements and optimization of lidar odometry have been proposed in [33]. By including the IMU in the optimization via a factor graph framework, Shan *et al.* [34] further built on lidar odometry SLAM. Lidar SLAM using surfels has been proposed in *e.g.* [5] and in [8] the algorithm was further developed using semantic information. Our SLAM algorithm adapts two specific methods from [37], namely the feature extraction and the mapping algorithms. A novelty in our proposed solution compared to the mapping algorithm [37] is that we introduce feature age as a weight factor, obtaining a low-drift algorithm we call precision lidar odometry.

In contrast to the lidar odometry algorithms previously mentioned, our proposed SLAM-algorithm utilizes a plane representation directly in the graph optimization. In order to utilize planes directly in the estimation formulation a good parametrization of the planes is required. A spherical parametrization can be used as in [25], where a method for indoor mapping with a RGB-D sensor was presented. Another minimal representation called closest point is suggested in [15], where infinite plane primitives were used and evaluated using simulated 3D-lidar data. In [22] a plane representation suitable for least squares optimization was presented and evaluated with a RGB-D camera. This representation was further used by Hsiao *et al.* [20] [19], creating a keyframe based dense SLAM with a RGB-D camera. While full 3-DOF lidar planes have been used in graph SLAM in previous works, our algorithm applies the concept to real-world data with a 3D-lidar. By combining the features and lidar odometry from [37], the loop closing algorithm from [34], and planar feature representation from [22], we have created a novel full 3D-SLAM algorithm combining an IMU, precision lidar odometry, loop closing, and full 3-DOF planar primitives in a graph-based structure. This is to our knowledge a SLAM method that has not been presented in any prior works.

Several previous lidar SLAM studies have evaluated and compared the accuracy between different methods including recent both outdoor [13, 29] and indoor [39] applications. To our knowledge, no previous study has addressed or quantified the specific accuracy performance or degradation when passing through narrow openings and we thus consider our study a relevant contribution also in this aspect. For the accuracy evaluation through narrow passages we employ an error metric proposed by Kümmerle [24], where the error is calculated as the average of all relative relations at a fixed distance along the trajectory. The metric was fur-

ther employed by Geiger [14] and Hess [17]. This method of quantifying errors has the advantage of consistently estimating error values regardless where the error occurred or in which order the data is processed.

3. Methods

3.1. Hardware overview

The system on which the proposed SLAM method is validated consists of an Ouster OS1-16 lidar. The OS1-16 lidar outputs a 360° point cloud consisting of 16x1024 points at 10 Hz. The 6-axis 100 Hz IMU included in the OS1-16 lidar is also incorporated in the SLAM-algorithm. By mounting the lidar at a 9° angle as shown in Figure 1 we make sure that the resulting point cloud contains information in all degrees of freedom, including points from horizontal planes such as the floor. Since our SLAM-method is to be executed in realtime on a small UGV or UAV, it is crucial that the onboard computer is both lightweight and power efficient. For this application we use the Raspberry Pi 4 8G, due to its ease of use, low cost and relatively high performance to weight and power ratios.

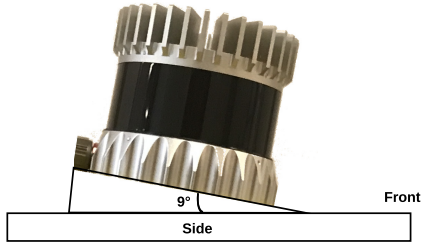


Figure 1. Schematic sketch of the system hardware.

3.2. Software overview

Our algorithm consists of 4 main stages as can be seen in the flowchart in Figure 2. The stages are: a) Feature extraction and association, which we denote frontend, b) Data fusion of IMU and infinite plane features, which we denote backend, c) Precision lidar odometry, and d) Loop closing. These stages will be explained in Sections 3.2.1 to 3.2.5.

3.2.1 Feature extraction

Due to the constraints of running in realtime on a small platform and the relatively large amount of data that is generated by the lidar, it is crucial to abstract the lidar data to a more manageable form at an early stage in the processing. For this purpose we use features extracted from the lidar data instead of the whole point cloud. The features are extracted and selected with the same method as in

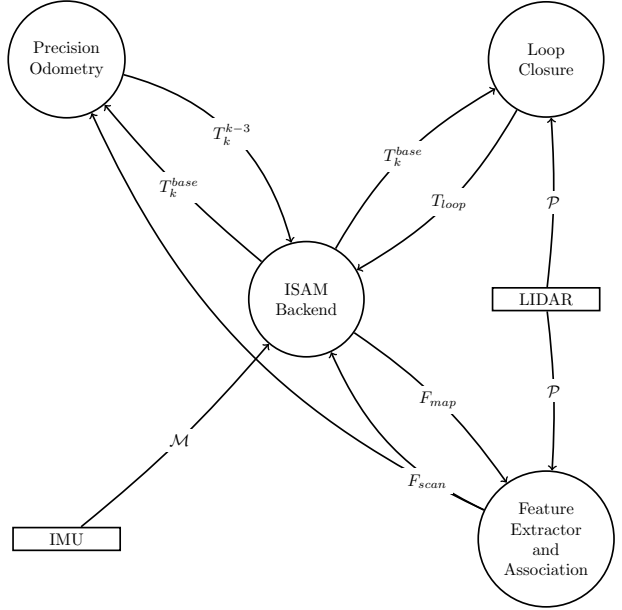


Figure 2. Flowchart of the system software. \mathcal{P} is a point cloud scan, \mathcal{M} is IMU data, T_j^i is the transform from i to j , F_{map} is the feature map and F_{scan} is the features extracted from the most recent scan.

LOAM [37], and consists of edges and planes. For the extracted plane features, lidar points are fitted to planes and the feature with the lowest point-to-plane RMSE distance is selected. For edges a similar selection is made except that the lowest point-to-line RMSE distance is used.

3.2.2 Feature association

The feature association in our algorithm is kept simple in order to minimize the computational complexity. For each new feature in a lidar-frame a Nearest Neighbor (NN)-search to the current map created in the backend step Section 3.2.3 is done. We have complemented each plane in the map with a center point and use this in the nearest neighbor search. The NN-search is done through kd-trees and in addition to the Cartesian distance between center points, the normal direction of the planes is used.

3.2.3 ISAM backend

The backend is built using the sensor fusion library Georgia Tech Smoothing And Mapping (GTSAM) [10]. To be able to work on data in realtime, the methods for Iterative Smoothing And Mapping (ISAM) [23] available in the GTSAM-library are used. By using IMU preintegration as presented in [12] and [7], along with a minimal representation of our plane features as presented in [22] we can achieve fast (10Hz) and accurate positioning over a short

time span. The extra computational complexity that arise from having infinite planes in the factor graph is a challenge. As the map of features in the graph grows larger, it becomes impossible to fulfill the imposed time and computation constraints. The factor graph is therefore pruned when it grows to large, keeping only the most recent features as part of the ISAM optimization. This will of course lead to a growing error over longer time spans, since we cannot connect our most current measurements with older parts of the map. We solve this problem by having the precision lidar odometry handle drift over longer time spans. The precision stage of the algorithm operates at a slower pace than the rest of the algorithm and feeds odometry measurements to the ISAM backend at a rate of 2.5 Hz. The precision stage therefore functions as a short term loop closing algorithm, and is further described in Section 3.2.4. The final step is to include real, long-term loop closing. The factor graph framework is excellent for doing this and our algorithm is further described in Section 3.2.5. In total, we use four different factors in the graph: IMU Preintegration factor, Infinite plane factors (called Oriented3DPlane in GT-SAM), precision lidar odometry, and a loop closing factor. An overview can be seen in Figure 3.

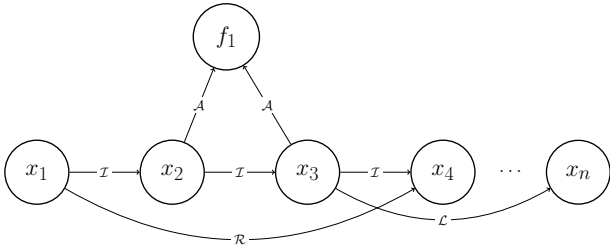


Figure 3. Example of the graph-structure. x_i are pose values, f_i are feature values, \mathcal{T} are Preintegrated IMU factors, \mathcal{A} are Oriented3DPlane factors, \mathcal{R} are precision odometry between factors and \mathcal{L} are loop closing between factors.

3.2.4 Precision lidar odometry

The precision step is used to reduce drift in the x-,y-,z-directions, and around the yaw axis. Features from the four latest scans are registered using the odometry estimate from the ISAM backend and merged together. These are then associated, using NN-search, to a feature map consisting of previously merged features. The corresponding poses are then optimized using a rigid transform with the matches between merged features and the previous map and the result is fed back into the ISAM backend. The method is similar to the mapping algorithm in LOAM [37] with the difference that we have added a further improvement by including feature age as a weight factor to the map features. The precision lidar odometry improves the accuracy in two ways.

First, it reduces the pose error caused by errors in individual lidar range measurements by combining multiple scans, in this case four. Second, it reduces accumulation of errors over time by applying higher weight to older features on the map. The weight assigned to a map feature is a linear weight to its age, and the maximum weight is set to a fixed value of 10, *i.e.* all features older than 10 s have equal age-weights.

3.2.5 Loop closing

Whilst the precision lidar odometry handles drift over longer time spans, it will still build up error over time. When creating larger maps we have found it crucial to correct long-term drift by searching for possible loop closing opportunities. This part of the SLAM method works in its own thread at a variable speed that is typically at slower rate than the precision lidar odometry. The algorithm first lists all scans that are both closer and older than respective thresholds. After a suitable previous pose has been found we use Generalized Iterative Closest Point (ICP) [32] to do individual scan to scan matching. If a match between the current scan and a previous scan is found, *i.e.* the resulting error from the ICP-algorithm is below a certain threshold we feed the calculated transform back to the GTSAM backend to handle the correct map updates. In this part, we work on a subsampled version of the point cloud frame, not the extracted features.

4. Evaluation and results

The accuracy of our SLAM method is evaluated with data from two main experimental test cases. The data from both test cases are also evaluated and compared to the two representative state-of-the-art SLAM methods Cartographer [17] and LIO-SAM [34]. Additionally, the CPU load and memory usage are evaluated for the three SLAM solutions. The purpose of the first experimental case is to evaluate the lateral positioning error during passage through narrow openings. The purpose of the second test case is to evaluate the translation and rotation errors over sequences with common start and stop poses.

In order to evaluate LIO-SAM and Cartographer on the same hardware and under the same conditions, some modifications were made. Since LIO-SAM in its native form does not work with a 6-axis IMU, an estimate of orientation must be calculated in advance if the Ouster IMU is used. We implemented a simple algorithm for estimating roll and pitch based on gyro and accelerometer data as described in [31]. For the heading, we fed back the estimate that LIO-SAM outputs from the last timestep. Cartographer in its original form does not output a velocity estimate, however it uses an internal velocity estimate based on the derivative of the calculated poses. We altered the Cartogra-

pher code to output this velocity, however it will naturally be more noisy than the velocity estimates from the other algorithms, which should be kept in mind in evaluation of lateral speed accuracy in Section 4.1.

4.1. Accuracy in narrow passages

The evaluation of passage through narrow openings is performed with data from an office environment. Data were collected with the lidar mounted on a roller table along trajectories with start and stop positions as indicated in the schematic in Figure 4. This figure also shows photographs of the environment. The purpose of using the roller table was to minimize the movement of the lidar in the lateral (sideway) direction. The trajectories were divided into four parts: A, B, T and R, where A and B represent passages through doorways along a straight line. Passage A is slightly narrower than passage B. Passage T represents a passage through a doorway immediately after a 90-degree turn. Passage R is used as a reference in the less narrow corridor for comparison to the results from the narrow passages A, B, and T. To examine the influence from the presence of additional objects in a cluttered scene, chairs were placed in the corridor environment and plastic bags were introduced in the scene in Room 2 as shown in the right part of Figure 4. These passages are labeled A_c , B_c , T_c , and R_c .

Examples of trajectories from the SLAM methods in the reference passage R_c and the cluttered passage B_c are shown in Figure 5. The data from all three methods were analyzed at an output rate of 100 Hz. The start and stop positions of each passage were defined by manually marking, respectively, a position 0.5 m in front of the doorway and a position 1 m after passing the doorway. The corresponding sample times t_1 (start) and t_2 (stop) were then used for the evaluation of each passage. The forward movement direction was estimated by drawing a straight line between the centroid positions of the start and stop points, where the centroids were calculated from trajectory points 50 samples (0.5 s) before and after t_1 and t_2 . In Figure 5, the movement along the straight line is represented by the y-axis value (lateral position) 0 cm. Thus, the y-axis values show the offset positions perpendicular to the direction of movement of the roller table. Figure 5 also shows estimated quantities of lateral errors, denoted absolute and maximum errors. The absolute error is calculated as the average of all relative relations at a fixed distance along the passage as proposed by Kümmerle [24]. This error quantity was also used by *e.g.* by Hess [17], both as absolute and squared values. In our study we only report the absolute error values. The error quantity consistently estimates the error values regardless where the error occurred along a passage. In our case, this error metric also is invariant to possible inaccuracies in the estimated start and stop centroid positions. As a complement and possibly more intuitive error quantity, we also calculate the ab-

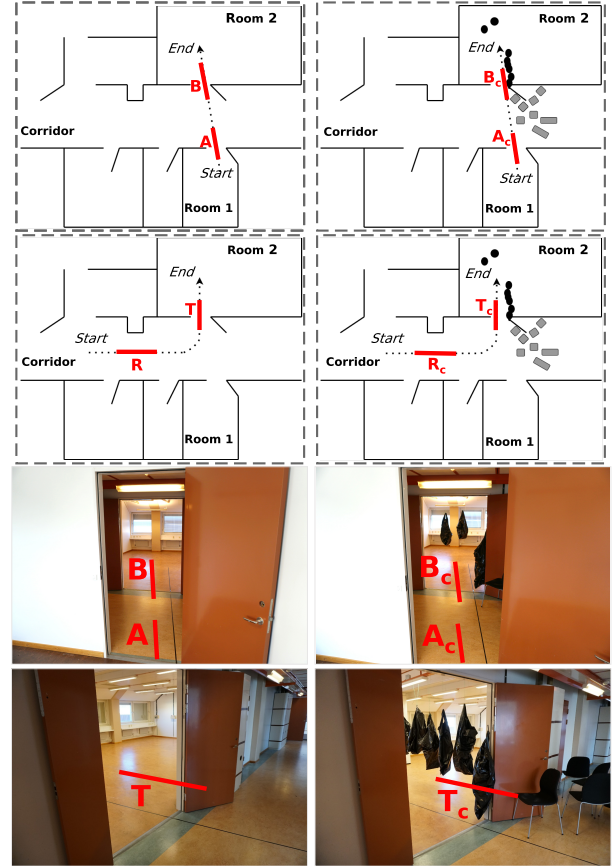


Figure 4. Schematic illustrations and photographs of the environment for evaluation of accuracy in narrow passages.

solute maximum lateral offset position from the straight line from the start and stop centroid positions (denoted Max. err. in Figure 5). An observation of the data in Figure 5 is that Cartographer and LIO-SAM processed data have larger lateral errors in the narrow passage B_c than in the reference passage R_c . Figure 6 shows examples of the lateral speed and a lateral speed error metric for the same raw data as shown in Figure 5. The lateral speed is calculated as the speed perpendicular to the straight line between the start and stop centroid positions. The RMSE lateral speed is calculated with the assumption of a true lateral speed equal to 0 cm/s.

The evaluation results through narrow passages with forward speed 0.2 m/s are presented in Table 1 and summarized in Table 2 for forward speed 0.5 m/s. The values presented for speed 0.2 m/s are the average and standard deviation from three experimental walks through each passage. Both tables show the values for all cluttered (9 walks), uncluttered (9 walks), and reference passages (6 walks). On each row and for each error metric, the smallest values are indicated in bold face.

Table 1. Quantitative lateral errors over narrow passages at average forward speed 0.2 m/s.

Passage	Cartographer			LIO-SAM			Ours		
	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]
A	0.84 ± 0.06	7.6 ± 0.52	8.5 ± 0.67	0.62 ± 0.42	11.5 ± 9.43	7.2 ± 4.39	0.24 ± 0.01	4.1 ± 0.30	2.1 ± 0.17
B	0.73 ± 0.05	5.5 ± 1.04	7.3 ± 0.45	0.56 ± 0.15	10.1 ± 4.60	7.6 ± 2.95	0.22 ± 0.02	4.7 ± 0.84	2.9 ± 0.53
T	0.54 ± 0.01	4.6 ± 0.88	5.0 ± 0.34	0.24 ± 0.02	3.2 ± 1.15	3.4 ± 1.10	0.25 ± 0.05	6.9 ± 0.67	3.0 ± 0.29
A _c	0.80 ± 0.05	5.7 ± 1.44	7.9 ± 0.82	0.51 ± 0.30	8.4 ± 6.00	7.5 ± 3.30	0.27 ± 0.04	4.1 ± 1.70	2.6 ± 0.77
B _c	0.69 ± 0.04	6.2 ± 2.20	7.4 ± 0.66	0.32 ± 0.07	5.5 ± 1.61	4.7 ± 3.13	0.21 ± 0.03	5.0 ± 2.25	2.8 ± 0.29
T _c	0.60 ± 0.02	5.2 ± 0.65	6.1 ± 0.45	0.22 ± 0.04	3.1 ± 1.43	3.6 ± 0.65	0.27 ± 0.04	5.5 ± 1.12	3.1 ± 0.58
R	0.51 ± 0.07	4.2 ± 0.51	5.1 ± 0.55	0.18 ± 0.01	2.8 ± 0.87	2.4 ± 1.35	0.19 ± 0.02	4.3 ± 1.43	2.3 ± 0.75
R _c	0.55 ± 0.16	4.8 ± 1.44	5.4 ± 1.11	0.13 ± 0.03	2.0 ± 0.21	2.2 ± 0.66	0.17 ± 0.03	2.3 ± 0.27	1.3 ± 0.10
Passages									
A, B, T	0.71 ± 0.14	5.9 ± 1.51	6.9 ± 1.58	0.47 ± 0.30	8.2 ± 6.52	6.1 ± 3.36	0.24 ± 0.03	5.2 ± 1.38	2.7 ± 0.56
A _c , B _c , T _c	0.70 ± 0.09	5.7 ± 1.42	7.1 ± 0.99	0.35 ± 0.20	5.7 ± 3.91	5.3 ± 2.87	0.25 ± 0.05	4.8 ± 1.68	2.8 ± 0.55
R, R _c	0.53 ± 0.11	4.5 ± 1.03	5.3 ± 0.80	0.16 ± 0.03	2.4 ± 0.74	2.3 ± 0.96	0.18 ± 0.03	3.3 ± 1.41	1.8 ± 0.75

Table 2. Quantitative lateral errors over narrow passages at average forward speed 0.5 m/s.

Passages	Cartographer			LIO-SAM			Ours		
	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]	Absolute lateral errors [cm]	Maximum lateral errors [cm]	RMSE lateral speed errors [cm/s]
A, B, T	0.71 ± 0.09	4.7 ± 0.87	7.3 ± 0.67	0.39 ± 0.17	4.4 ± 2.18	5.7 ± 2.73	0.35 ± 0.08	3.6 ± 0.85	4.0 ± 1.26
A _c , B _c , T _c	0.87 ± 0.21	5.5 ± 1.57	8.8 ± 2.27	0.64 ± 0.55	6.4 ± 6.44	8.6 ± 6.42	0.42 ± 0.04	4.6 ± 1.86	6.0 ± 1.10
R, R _c	0.55 ± 0.15	4.5 ± 1.13	6.1 ± 1.38	0.19 ± 0.06	1.6 ± 0.49	2.2 ± 1.41	0.24 ± 0.06	2.5 ± 0.82	2.6 ± 1.05

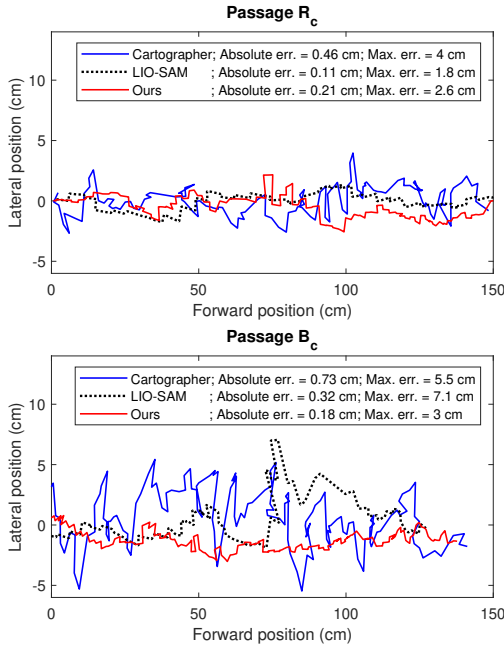


Figure 5. The trajectories in forward and lateral directions produced by Cartographer, LIO-SAM, and our SLAM solution at a forward speed of about 0.2 m/s over passages R_c and B_c.

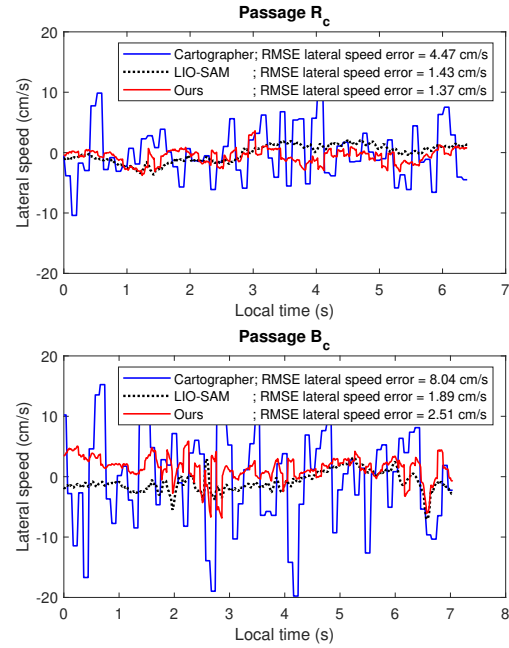


Figure 6. The lateral speed produced by Cartographer, LIO-SAM, and our SLAM solution at a forward speed of about 0.2 m/s over passages R_c and B_c.

For all three SLAM solutions and both forward speeds, the lateral position and speed errors are significantly larger during narrow passages compared to the reference passages in the corridor. It is noted that the lateral position and speed errors in the reference passages R , R_c do not significantly change between low and high forward speeds.

At forward speed 0.2 m/s, neither of the solutions show any significant difference in lateral accuracy for the cluttered compared to the uncluttered scene. However, at 0.5 m/s all three methods have much larger lateral errors in the cluttered passages, both in speed and either in absolute error (Cartographer, LIO-SAM) or in maximum error (ours).

Table 1 indicates that Cartographer and LIO-SAM exhibit the largest lateral errors in the narrowest passage A (both uncluttered and cluttered). Passage T, immediately after a 90-degree turn, gives similar errors for our SLAM method compared to a straight passage B. Cartographer and LIO-SAM produces significantly smaller errors after a turn T compared to a straight passage B. A possible explanation for the similar or better accuracy after a turn might be the fact that more parts of Room 2 (Figure 4) are exposed to the lidar sensor during the turn and thus more features in Room 2 can be detected before actually entering Room 2.

An overall comparison of all three SLAM methods at both forward speeds gives that LIO-SAM has the smallest lateral errors in the corridor reference passages and that our method has the smallest lateral errors in the narrow passages.

4.2. Absolute pose accuracy

As a complement to the evaluation of narrow passages in Section 4.1 an evaluation of the absolute error that builds up over time was also performed. Data from two different environments are analyzed, a large indoor corridor scene and a small cluttered kitchen environment, see Figures 7 and 8.



Figure 7. Photographs from the environments for evaluation of absolute pose accuracy.

In both datasets, the trajectory is a loop, making sure that start and end poses are identical. Since we want to evaluate the error that builds up over time, we show the results both with and without loop closing algorithms. Both translational and rotational absolute errors between start and end poses are presented in Table 3. All three algorithms provide

Table 3. Position and rotation errors between start and end poses for two different environments. The presented values are the average and the standard deviation over three separate runs with similar trajectories.

Large indoor corridors (297m)		Translation error [m]	Rotation error [°]
Without loop closing	Cartographer	5.36 \pm 0.33	3.78 \pm 0.28
	LIO-SAM	8.31 \pm 5.18	7.52 \pm 2.30
	Ours	8.79 \pm 3.92	7.10 \pm 5.56
With loop closing	Cartographer	0.35 \pm 0.15	1.72 \pm 0.57
	LIO-SAM	0.69 \pm 0.97	3.70 \pm 4.88
	Ours	0.08 \pm 0.09	0.68 \pm 0.62
Small cluttered kitchen (17m)		[m]	[°]
Without loop closing	Cartographer	0.21 \pm 0.09	1.5 \pm 0.73
	LIO-SAM	0.85 \pm 0.74	10.83 \pm 10.06
	Ours	0.04 \pm 0.03	1.37 \pm 1.32
With loop closing	Cartographer	0.18 \pm 0.05	1.41 \pm 0.48
	LIO-SAM	0.80 \pm 0.25	7.36 \pm 2.03
	Ours	0.01 \pm 0.01	1.26 \pm 1.33

excellent results when their respective loop closing algorithms are used. Cartographer seems however to have the lowest drift over time, when no loop closing is allowed.

4.3. Computation requirements

Since the focus of our SLAM-algorithm is to run in real-time on a small platform it is essential to examine the computation requirements on the intended hardware. The three algorithms were run on the target hardware. The CPU load and memory usage are displayed in Figure 9. Only the CPU processes connected to each respective algorithm was taken into consideration. Cartographer and our algorithm both provided identical results on the Raspberry Pi as on a desktop computer, LIO-SAM struggled somewhat with the limited computational resources and thus the quality of the result was somewhat degraded. CPU usage increases drastically in the beginning as the maps and graphs are built, and all three algorithms plateau after a certain amount of time. The evaluation was performed on the large indoor environment presented in Section 4.2, and all algorithms ran without loop closing.

5. Conclusion

A novel lidar SLAM method was presented which combines lidar odometry, loop closing, and full 3-DOF planar primitives in a graph-based structure. The method was experimentally evaluated and compared to the state-of-the-art methods Cartographer and LIO-SAM. Our method is adapted for, and was demonstrated to run on low computing resources. The approach was developed for deployment on small autonomous vehicles operating in unknown or previously unmapped indoor environments.

In the experimental evaluation, we specifically addressed

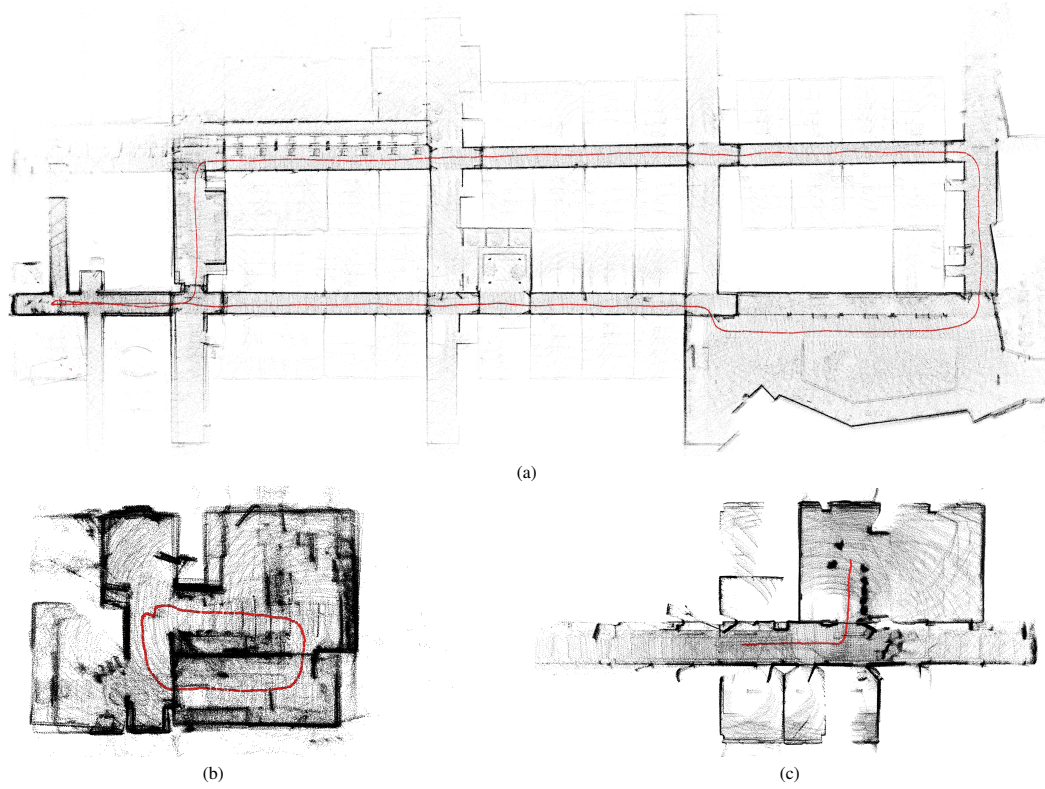


Figure 8. Point clouds of three different environments. a) is the large indoor corridor dataset, b) is the cluttered kitchen dataset, and c) is an example of a dataset from the narrow passage evaluation set.

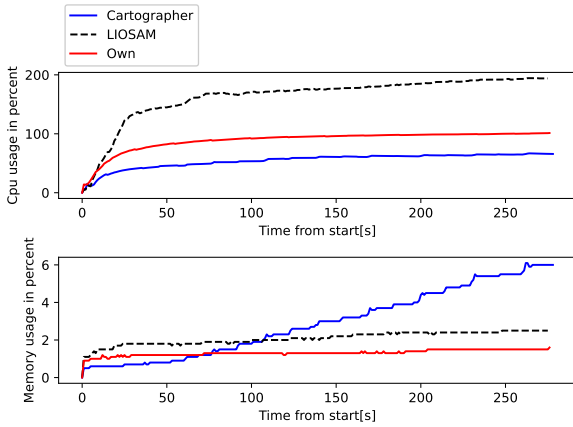


Figure 9. Computational performance on a Raspberry Pi 4 8G. 100% corresponds to one full CPU core in the top plot, and corresponds to 8GB of RAM in the bottom plot.

and quantified the lateral positioning accuracy when passing through narrow openings such as doorways, a situation when high accuracy is a prerequisite for safe navigation. An important result from the experimental evaluation is that

a decrease in lateral accuracy occurs for all three SLAM methods when passing through the narrow openings compared to operation in larger spaces such as corridors. In the comparison, LIO-SAM produced the lowest lateral errors in the corridor environment while our proposed method gave the smallest lateral errors in the narrow doorway passages. In experiments over longer sequences without loop closure, Cartographer produced the lowest long-term drift. All three methods provided excellent results when their respective loop closing algorithms were used.

6. Acknowledgments

This work has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement No. 883345, INTREPID (“Intelligent Toolkit for Reconnaissance and assessment in Perilous Incidents”), and from the European Union’s Horizon 2020 Research and Innovation Programme and the Korean Government under Grant Agreement No 883345, INGENIOUS. Content reflects only the authors’ view, and the Research Executive Agency (REA) and the European Commission are not responsible for any use that may be made of the information it contains.

References

- [1] Mohammad O. A. Aqel, Mohammad H. Marhaban, M. Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *Springer-Plus*, 5(27843754):1897–1897, Oct. 2016. [2](#)
- [2] Shahrukh Ashraf, Priyanka Aggarwal, Praveen Damacharla, Hong Wang, Ahmad Y Javaid, and Vijay Devabhaktuni. A low-cost solution for unmanned aerial vehicle navigation in a global positioning system-denied environment. *International Journal of Distributed Sensor Networks*, 14(6):1550147718781750, 2018. [2](#)
- [3] Ahmad Azar, Anis Koubaa, Nada Mohamed, Habiba Ibrahim, Zahra Fathy, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa Khamis, Ibrahim Hameed, and Gabriella Casalino. Drone deep reinforcement learning: A review. *Electronics*, 10, 04 2021. [1](#)
- [4] F Azhari, S Kiely, C Sennersten, C Lindley, M Matuszak, and S Hogwood. A comparison of sensors for underground void mapping by unmanned aerial vehicles. In M Hudyma and Y Potvin, editors, *UMT 2017: Proceedings of the First International Conference on Underground Mining Technology*, pages 419–430. Australian Centre for Geomechanics, 2017. [2](#)
- [5] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, page 59, 2018. [2](#)
- [6] Bruno Bodin, Harry Wagstaff, Sajad Saecdi, Luigi Nardi, Emanuele Vespa, John Mawer, Andy Nisbet, Mikel Lujan, Steve Furber, Andrew J. Davison, Paul H. J. Kelly, and Michael F. P. O’Boyle. Slambench2: Multi-objective head-to-head benchmarking for visual slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3637–3644, 2018. [2](#)
- [7] Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4290–4297. IEEE, 2014. [3](#)
- [8] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. [2](#)
- [9] Tianxiao Cui, Chen Guo, Yi Liu, and Zeyu Tian. Precise landing control of uav based on binocular visual slam. In *2021 4th International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 312–317, 2021. [2](#)
- [10] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. [3](#)
- [11] Isaac Deutsch, Ming Liu, and Roland Siegwart. A framework for multi-robot pose graph slam. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572, 2016. [2](#)
- [12] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, 2015. Georgia Institute of Technology, 2015. [3](#)
- [13] Matteo Frosi and Matteo Matteucci. Art-slam: Accurate real-time 6dof lidar slam. *IEEE Robotics and Automation Letters*, 7(2):2692–2699, April 2022. [2](#)
- [14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. [3](#)
- [15] Patrick Geneva, Kevin Eickenhoff, Yulin Yang, and Guoquan Huang. Lips: Lidar-inertial 3d plane slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 123–130. IEEE, 2018. [2](#)
- [16] Gerard Gibbs, Huamin Jia, and Irfan Madani. Obstacle detection with ultrasonic sensors and signal analysis metrics. *Transportation Research Procedia*, 28:173–182, 2017. IN-AIR 2017. [2](#)
- [17] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016. [2](#), [3](#), [4](#), [5](#)
- [18] Stefan Hrabar. Reactive obstacle avoidance for rotorcraft uavs. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4967–4974, 2011. [1](#)
- [19] Ming Hsiao, Eric Westman, and Michael Kaess. Dense planar-inertial slam with structural constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6521–6528. IEEE, 2018. [2](#)
- [20] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5110–5117. IEEE, 2017. [2](#)
- [21] Youeyun Jung, Hyo choong Bang, and Dongjin Lee. Robust marker tracking algorithm for precise uav vision-based autonomous landing. *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, pages 443–446, 2015. [1](#)
- [22] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, 2015. [2](#), [3](#)
- [23] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. [3](#)
- [24] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387, 2009. [2](#), [5](#)
- [25] Tae-kyeong Lee, Seungwook Lim, Seongsoo Lee, Shounan An, and Se-young Oh. Indoor mapping using planes extracted from noisy rgb-d sensors. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1727–1733. IEEE, 2012. [2](#)
- [26] Xiaodong Li, Nabil Aouf, and Abdelkrim Nemra. Estimation analysis in vslam for uav application. In *2012 IEEE Interna-*

tional Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pages 365–370, 2012. 2

- [27] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2017. 2
- [28] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012. 1, 2
- [29] Qingyu Meng, Hongyan Guo, Xiaoming Zhao, Dongpu Cao, and Hong Chen. Loop-closure detection with a multiresolution point cloud histogram mode in lidar odometry and mapping for intelligent vehicles. *IEEE/ASME Transactions on Mechatronics*, 26(3):1307–1317, 2021. 2
- [30] Yasir Mohd Mustafah, Amelia Wong Azman, and Fajril Akbar. Indoor uav positioning using stereo vision sensor. *Procedia Engineering*, 41:575–579, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012). 2
- [31] William Premerlani and Paul Bizard. Direction cosine matrix imu: Theory. *Diy Drone: Usa*, 1, 2009. 4
- [32] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. 4
- [33] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018. 2
- [34] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020. 2, 4
- [35] Xiangqian Shu, Lingyu Yang, Xiaoke Feng, and Jiansong Zhang. An imu/sonar-based extended kalman filter for mini-uav localization in indoor environment. *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, pages 1–6, 2018. 2
- [36] Janis Tiemann, Florian Schweikowski, and Christian Wietfeld. Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, 2015. 2
- [37] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 2, 3, 4
- [38] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 02 2017. 2
- [39] Lipu Zhou, Daniel Koppel, and Michael Kaess. Lidar slam with plane adjustment for indoor environment. *IEEE Robotics and Automation Letters*, 6(4):7073–7080, Oct 2021. 2