LPG-SLAM: a Light-weight Probabilistic Graph-based SLAM

Kathia Melbouci, Fawzi Nashashibi

2020 16th International Conference on
Control, Automation, Robotics and Vision (ICARCV)
December 13-15, 2020. Shenzhen, China

LPG-SLAM: a Light-weight Probabilistic Graph-based SLAM

Kathia Melbouci[1] and Fawzi Nashashibi[1]

Abstract— Most of Current autonomous navigation solutions critically rely on SLAM systems for localisation, especially in GPS-denied environments but also in urban and indoor environments. Their efficacy an d effi ciency thus depend on the ability of the underlying SLAM method to map large-scale environments in a data-efficient ma nner. State of the art systems, while accurate, often require powerful hardware such as GPUs, and need careful tuning of hyper-parameters in order to adapt to the user's needs. In this paper, we propose a light-weight but accurate probabilistic 2D graph-based SLAM system. We validate our approach on sequences from the KITTI dataset as well as on data gathered by our experimental platform.

Index Terms— SLAM, Mapping, 2D Lidar, Graph based optimization, loop closure.

I. INTRODUCTION

Simultaneous Localization And Mapping (SLAM) is critical in urban environments, especially in case of degraded or missing GPS signal. In the past decades, significant progress has been made towards that goal. State of the art solutions rely on fusion between complementary exteroceptive (e.g. LIDAR, camera) and proprioceptive (e.g. IMU, odometry) sensors. While sensor graphs that make use of passive vision as their primary source of geometric information about the environment (e.g. [1], [2], [3]) are inexpensive, they will often fail in poorly textured environments or in scenes with challenging lighting conditions, and will overall have inferior accuracy compared to methods that make use of active vision sensors such as laser sensors. Indeed, the dense and fine-grained geometric information that such sensors provide result in less uncertainty in depth values and robustness to challenging environmental changes. The accuracy and robustness of solutions exploiting 3D sensors (e.g. [4], [5], [6]) however comes at the cost of increased computational complexity, which makes them difficult to de ploy on low-end hardware. In this work, we present a real-time method capable of exploiting 3d data from LIDAR which is much less demanding in terms of computation while retaining state of the art accuracy. In particular, we show that in situations where the main goal is accurate localization, a 2d occupancy grid computed from 3d data is sufficient. Fu rthermore, ou r method has fewer hyper-parameters and requires less fine-tuning during deployement in different s ituations. O ur method is closely related to the work of [7] as they use the same probabilistic framework. However, their solution is less accurate as they use a minimal map-matching, and don't take advantage of loop closure. We validate our approach on sequences from the KITTI dataset as well as on data gathered using our experimental apparatus, described in section IV.

The next section is dedicated to our work's relation with the state of the art. We detail the proposed system in section III. Quantitave and qualitative results are presented and discussed in section IV, and closing remarks and future directions are given in section V.

II. Related work

In this section, we review advances that have been made in the graph-based approach to SLAM [8] in large-scale urban environments based on LIDAR data and position ourselves with respect to recent works.

One of the first graph-based SLAM approches targeting large-scale and unstructured environments is described in [9]. The authors present a framework for view-based SLAM using 2D laser range measurements. They construct local maps of a sequence of aligned scans using a robust iterative scan matching technique incorporated into an extended Kalman filter. To enable the detection of loop closure without depending on prior knowledge, they perform map matching via histogram cross-correlation and entropy sequence projections. Adding the global optimization step to Kalman filter process grows the computation complexity, which makes the approach unfit for high frequencies applications.

Exploiting the local map idea mentioned above, [10] propose a 2D real time graph-based LIDAR SLAM which operates on a filtered point cloud. Pairs of local maps and associated scans are stored in memory, as well as the position of each scan. At fixed time intervals, the scan matcher tries to find the position of a current scan in the stored local maps by correlative-scan-matching ([11]), and adds it as an edge in the graph. Subsequently, the optimizer estimates the best locations for the scans and the associated local maps that reduces to a classical non linear least square error. In order to achieve real-time results on modest hardware, this process follows the sparse pose adjustment method described in [12], implemented in a separate thread. However, the size of each local map constructed by the front-end grows with the number of scans that are inserted into it. Larger numbers of scans in a map lead to more accurate scan-matching, but increase the time that is necessary to

*This work has been supported by the French project CAMPUS, a PIA project funded by the French ADEME (Agence de l'Environnement et de la Maîtrise de l'Energie).

[1]Robotics and Intelligent Transportation Systems (RITS) team, Inria Paris, 56 rue du Charolais, Paris 75012, France

detect loop closures. Here, we make use of their proposed background loop detection process. In contrast to their work however, the aforementioned trade-off between accuracy and efficency resulting from map size is handled dynamically by our system.

In [4], the authors extend the ideas described above to 3D data. Local maps are encoded by surfels [13] associated with a rendered vertex and normal map. Thus the detection of loop closure is no longer done by correlative scan matching but via frame-to-model ICP [14]. Each observed scan is aligned with the known map, trying to minimize a point to plane error with respect to the rendered vertex map. Loop closure is included in the optimization step if the observed scan is consistent with the newly rendered map at the predicted position. While graph optimization is performed on a separate thread, the method still suffers from high time complexity as they process all 3D data.

In order to better approximate the real world and reject incorrect loop closures, different works have focused on data association strategies. In [15], the posterior distribution over the robot's location is modeled as a maximum-mixture of Gaussians. While their approach is useful when dealing with perceptual aliasing and multiple loop closure hypotheses, we found that an M-estimator produced satisfactory results in our experiments.

Since data association is a major source of error accumulation and drift, some works have mainly focused on improving the scan matching process ([15], [16]). For example, IMLS-SLAM [6] forgoes the use of a backend to concentrate on scan-to-model matching, which results in detailed and accurate maps. However, their method is very costly and doesn't run in real-time.

In this work we present a complete solution for graph based probabilistic SLAM using 2D occupency grids which operates in real-time thanks to an efficient map management process, and which dynamically handles the trade-off between accuracy and efficency. In particular, our experimental results demonstrate that accurate localization is possible by using 2d occupency grids instead, which is less expensive than processing the entire 3d point cloud.

III. System overview

An overview of our system is given in figure 1. The input consists of LIDAR scan data, which can be an accumulation of 2D scans or one revolution from 3D velodyne LIDAR. We preprocess the data in order to remove noise, dynamic objects and the road plane. Subsequently, we estimate the most likely current pose via multi-resolution map matching and then construct local submaps. The submaps are subdivided and stored in a memory-efficient manner. Those will then be retrieved by the optimizer, which as part of our backend, is responsible for large-scale corrections.

In the following, we detail the main components of our SLAM.

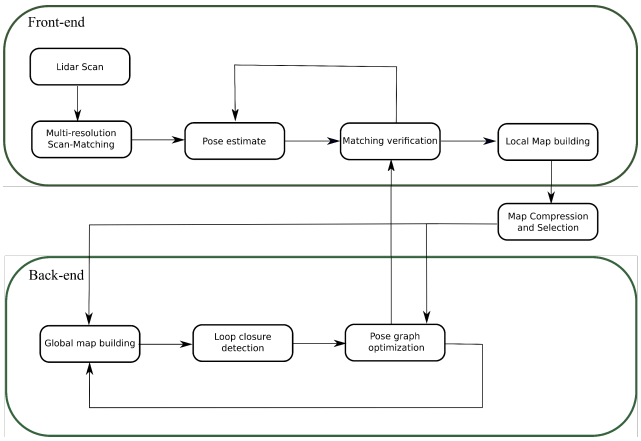

Fig. 1. An overview of our system's main components and their interactions.

A. Map construction

We represent the environment with an occupancy grid map $\mathbf{m}$, which in order to filter out noise and dynamic objects is estimated via computing a reflection map [17].

Given a set of vehicle poses $\mathbf{x}_{0,t}$ and a set of sensor measurements $\mathbf{z}_{1,t}$, the map is estimated by solving

$$\mathbf{m}^* = \underset{\mathbf{m}}{argmax}\ P(\mathbf{m} \mid \mathbf{z}_1, ..., \mathbf{z}_t, \mathbf{x}_0, ..., \mathbf{x}_t) \tag{1}$$

Assuming a uniform prior on the map and following Baye's rule, we obtain

$$\begin{aligned} \mathbf{m}^* &= \underset{\mathbf{m}}{argmax}\ P(\mathbf{z}_1, ..., \mathbf{z}_t \mid \mathbf{m}, \mathbf{x_1}, ..., \mathbf{x_t}) \\ &= \underset{\mathbf{m}}{argmax} \textstyle\prod_{t=1}^{N} P(\mathbf{z_t} \mid \mathbf{m}, \mathbf{x_t}) \\ &= \textstyle\sum_{t=1}^{N} \ln P(\mathbf{z_t} \mid \mathbf{m}, \mathbf{x_t}) \end{aligned} \tag{2}$$

whose numerical solution is given by counting the number of time the laser scan hits and misses each cell [17].

For real time map reconstruction, we need to sample the input point cloud since the the number of points provided by recent 3D sensors (e.g. Velodyne [18]) can reach ~ 200000 points per revolution, which makes processing the entire point-cloud in real-time infeasible on modest hardware. We choose an adaptive point cloud sampling described in [5].

B. Map management

Online matching is critical for navigation, and its failure is difficult to rectify without external information (i.e. additional sensors). Thus, local maps have to be large enough.

Loading large maps in memory continuously is time consuming. To handle this, we adopt a strategy inspired by the work of [7]. As shown in figure 2, we consider five equal-sized cells for each local map. The first one, noted l_0

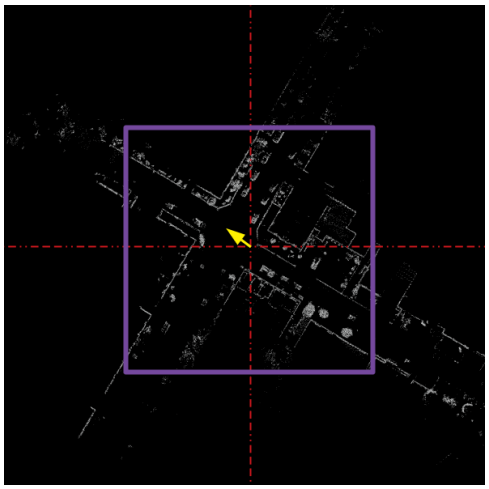

Fig. 2. Map cells used for scan-matching. The purple scan, centered around the vehicle (yellow arrow) is kept for immediate scan matching. The four cells resulting from subdividing the map in a 2×2 grid are compressed and stored along with their orientations with respect to the vehicle.

and shown in purple, is centered on the vehicle's position, and is kept in memory for immediate scan-matching. To obtain the four other cells $l_1, ..., l_4$, we subdevide the local map as a 2×2 grid and compute each cell's orientation o_i with respect to the vehicle (shown as a yellow arrow). This results in four pairs (l_i, o_i) that we compress and store in secondary memory. They will be retrieved only in case of scan-matching failure with l_0, in which case their order of selection will be determined by the precomputed o_i. These cells are only kept temporarily until a new local map is built, at which point they are replaced by its five subdivisions.

C. Pose estimation

The vehicle's pose is estimated from successive laser scan alignment. A scan $\mathbf{S}_j = \{\mathbf{s}_1, ..., \mathbf{s}_n\}$ is a set of n range measurements, with $\mathbf{s}_i = [x_i, y_i, z_i]^t$ expressed in the sensor's referential frame. At time t a scan $\mathbf{S}_t$ is matched with the previous mapped environment m to estimate the current pose x_t. This is done via maximizing the following likelihood:

$$p(\mathbf{x}_{0:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{x}_0) \prod_t [p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}) \; p(\mathbf{x}_t, M \,|\, \mathbf{x}_{t-1}, \mathbf{u}_{1:t})] \quad (3)$$

where $\eta \in \mathbb{R}$ is normalization constant, and where $p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{x}_{1:t-1}, \mathbf{u}_{1:t})$ and $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})$ respectively correspond to the motion and measurement models, which will be detailed in the following subsections.

a) Motion model: Without additional sensors, a constant velocity model is assumed. This assumption is required for lidar based systems with low rotating rate. The frequency of recent 3D lidars is around $10Hz$, a car moving at $100km/h$ travel $\simeq 2.8m$ per $\frac{1}{10}$ second. As the sensor is blind to changes that happen during that distance, we assume that the vehicle follows the constant velocity.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_{t-1} \delta t \quad (4)$$

Initial velocity is estimated aligning two laser scans in the absence of other sensors. Otherwise, it is estimated via an IMU/GPS fusion using Kalman filtering and providing a 3D position estimation including velocity.

b) Measurement model: To avoid convergence to local minima, most works use an exhaustive search over a set of candidates via correlative-scan matching (e.g. [16], [7], [19]). In this work we follow a similar approach and use multi-resolution-scan-to-map matching to compute $p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m})$.

At time step t, the Scan $\mathbf{S}_t$ is projected on a previously estimated part of the environment given by reference map m_r with position and orientation (p_r, θ_r) which encodes the log likelihood of observing a point at a specific location. The pose of the current scan in the global coordinates frame, noted $P_t \in SE(2)$, is computed by solving

$$P_t = \underset{i}{argmin} \sum_{i \in N} f_m(P_t \mathbf{s}_i) \quad (5)$$

where the function f_m finds the nearest occupied cell that lies on the ray starting from p_r and pointing in the direction of θ_r, and then returns its distance to s_i. We cast equation 5 as a non linear least square minimisation. To make the scan-matching more reliable and efficient for online mapping applications, we use multi-resolution scan-matching over three levels of the grid map (one more level than [16]).

D. Pose graph optimization

In our system, loop closure is performed as soon as high-confidence candidates become available. As previously mentioned (§II), we handle the issue of multiple candidates using an M-estimator. Given the current scan S_t at time t, we rank local map subdivisions based on their orientation relative to the vehicle (§III-B) and load them from secondary memory. We then perform many-to-many matching between all the scans from the candidate maps and the scans from S_t. Once a candidate is selected, we optimize

$$\underset{P}{argmin} \sum_k \mathbf{e}_{ij}^T(s) \Lambda_{ij} \mathbf{e}_{ij}(s) \quad (6)$$

using Levenberg-Marquardt's algorithm. The error state $\mathbf{e}_{ij}$ between node i and node j is defined as

$$\mathbf{e}_{ij} = (log_{SE(2)}((P_i^{-1} P_j P_{ij}))^\vee \quad (7)$$

where $\vee : \mathfrak{se}(2) :\rightarrow \mathbb{R}^2$ maps the tangent space of the corresponding Lie group to 2d Euclidean space, and where P_{ij} denotes the relative pose between the two

frames, which for loop closure candidates is given by the matching process.

At the end of the optimization, all local maps in secondary memory are updated.

IV. Experimental results

We have implemented the proposed graph-based SLAM using the C++ programming language, and used the CERES [20] library for bakend optimization needs. All of the presented experiments have been performed on a laptop computer equiped with an Intel Core i5-8350U CPU @ 1.70GHz @ 4 cores, running Windows 10.

We present quantitative and qualitative results on the KITTI odometry dataset [21], as well as qualitative results on the "campus dataset" that we have gathered using with a VLP-16 Puck [1] mounted on top of a car. In all cases, we apply the following preprocessing to the data: 1) we filter the point cloud in order to remove the road and dynamics objects in order to keep the data that correspond to the static structure of the scene. 2) As our SLAM is based on 2D grid-maps in order to be deployable on modest hardware, we remove the z component from the data.

As our method is closely related to the state of the art method PML-SLAM [7], we use their estimations as a baseline for comparison and show that our system consistenly produces more accurate results.

A. Experiments on the KITTI odometry dataset

Results are presented for the challenging sequences $00, 05$ and 07. Corresponding RMSE values of relative translational and rotational errors are respectively reported in tables I and II. We notice that our average translational error in all three sequences is inferior to $2m$, while PML-SLAM's error is on average larger than $4m$. Similarly, our orientation estimation is more accurate ($\sim 3^\circ$ vs $\sim 6^\circ$ on average). Trajectories as estimated by both PML-SLAM and our method are shown in figure 3 for visual appreciation. In all three cases, it can be seen that our estimated trajectory is closer to the ground truth than the estimation produced by PML-SLAM, which is subject to considerable drift as it doesn't do any loop closure. Among all sequences, sequence "00" is particularly challenging due to weaker geometric constraints from the environment. In particular, during initialization from LIDAR data only, the point cloud provides ambiguous constraints in the direction of motion (up to a translation).

method \ sequence	00	05	07
Ours	2.00	01.40	0.45
PML − SLAM	5.01	2.38	1.34

TABLE I
RMSE of relative translational error (in meters) on the KITTI odometry dataset.

method \ sequence	00	05	07
Ours	2.45	3.34	5.15
PML − SLAM	4.00	10.44	5.58

TABLE II
RMSE of relative rotational error (in degress) on the KITTI odometry dataset.

B. Campus dataset

This dataset consists of two sequences campus-A and campus-B that we have gathered using a VLP-16 sensor mounted on top of a car moving at $\sim 30m/s$. Unfortunately, we don't have any reliable ground-truth data for those sequences, so we overlay the estimated trajetories over satellite images from google maps (figures 4 and 5). Both sequences contain loops. The campus-A sequence is $459m$, which consists of a $\sim 285m$ loop around buildings followed by a straight trajectory of $\sim 174m$ where there are numerous dynamic objects (moving cars, pedestrians). The second sequence, campus-B, is ~ 296 meters long. In this loop, the vehicle travels around $40m$ on a straight line and then goes through a loop of $\sim 156m$. Figure 4 shows the comparison between the trajectory that is estimated by our method (yellow) and the trajectory that is estimated by PML-SLAM (red) on the campus-A sequence. As expected, the loop-closures performed by our system corrects the drift that is accumulated by PML-SLAM. Similar results can be observed in figure 5 for campus-B.

C. Running time and memory consumption

All experiments have been performed on an intel Core i5-8350U CPU @ 1.70GHz @ 4 cores, running Windows 10. Each new pass through our entire system (matching, pose and map estimation, loop closure and map update) takes on average $\sim 60ms$, which is significantly faster than the active sensor's frequency. Our system's memory consumption consistenly remains below $1.5Gb$.

V. Conclusion and future works

We have introduced LPG-SLAM, a light-weight probabilistic 2d graph-based SLAM which exploits data from high-frequency 3d sensors such as Velodyne LIDAR. We showed that 2d occupancy grids obtained from projecting 3d point clouds still provide strong geometric constraints that are sufficient for accurate localization on challening benchmarks. Unlike most state of the art methods which require powerful hardware such as GPUs, our formulation results in a lower-dimensional optimization problem which can be efficiently solved on low-end hardware. We saw however in some sequecnes that constraints from LIDAR data can be ambiguous. Future works will therefore include extending our framework into a

[1] https://www.cadden.fr/familles-de-produit/lidar-2d-et-3d/lidar-3d-velodyne

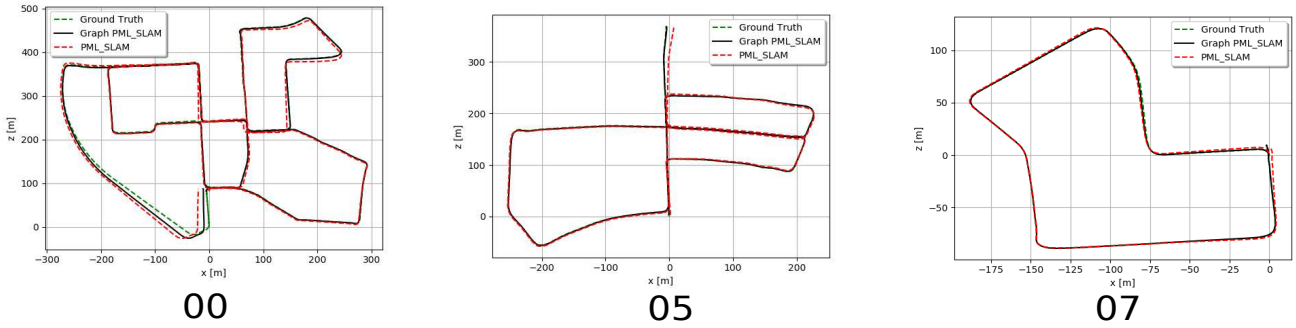

Fig. 3. Results on the KITTI odometry benchmark (sequences 00, 05, 07) : 2-d trajectories as estimated by our method and PML-SLAM.

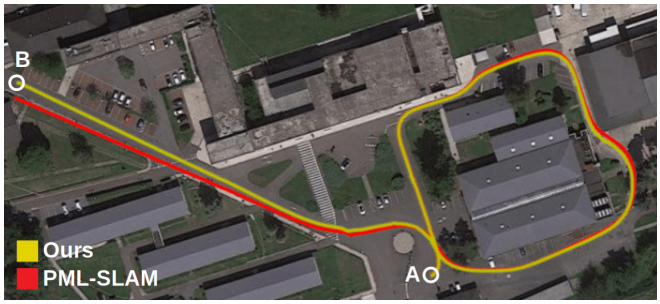

Fig. 4. Estimated trajectories from our method (yellow) and PML-SLAM (red) overlayed on google maps satellite images for the campus-A sequence. The trajectory starts in the area denoted *A* and ends in the area denoted *B*.

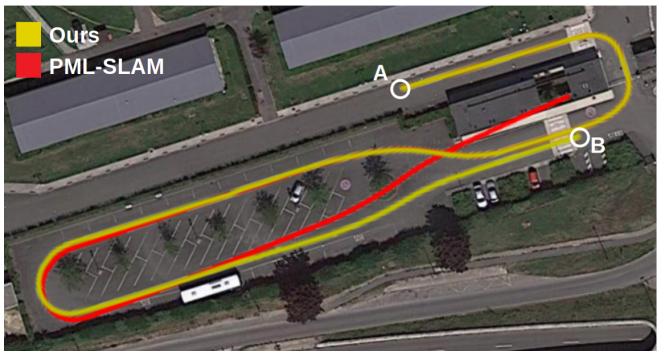

Fig. 5. Estimated trajectories from our method (yellow) and PML-SLAM (red) overlayed on google maps satellite images for the campus-B sequence. The trajectory starts in the area denoted *A* and ends in the area denoted *B*.

more general sensor fusion framework where among other sensors, we take advantage of GPS and odometry.

References

[1] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," The International Journal of Robotics Research, vol. 34, no. 3, pp. 314–334, 2015.

[2] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," Transactions on pattern analysis and machine intelligence, vol. 40, no. 3, pp. 611–625, 2017.

[3] A. Salehi, V. Gay-Bellile, S. Bourgeois, N. Allezard, and F. Chausse, "Large-scale, drift-free slam using highly robustified building model constraints," in International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 1586–1593.

[4] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." Proc. of Robotics: Science and Systems (RSS), 2018.

[5] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in International Conference on Robotics and Automation (ICRA). IEEE, 2016, pp. 1271–1278.

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

[7] Z. Alsayed, G. Bresson, F. Nashashibi, and A. Verroust-Blondet, "Pml-slam: a solution for localization in large-scale urban environments," in Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV). IEEE, 2015.

[8] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31–43, 2010.

[9] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based slam," The International Journal of Robotics Research (IJRR), vol. 27, no. 6, pp. 667–691, 2008.

[10] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in International Conference on Robotics and Automation (ICRA). IEEE, May 2016, pp. 1271–1278.

[11] E. B. Olson, "Real-time correlative scan matching," in International Conference on Robotics and Automation (ICRA). IEEE, 2009, pp. 4387–4393.

[12] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in International Conference on Intelligent Robots and Systems (IROS). IEEE, 2010, pp. 22–29.

[13] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 335–342.

[14] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2011, pp. 127–136.

[15] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," The International Journal of Robotics Research (IJRR), vol. 32, no. 7, pp. 826–840, 2013.

[16] E. Olson, "M3rsm: Many-to-many multi-resolution scan matching," in International Conference on Robotics and Automation (ICRA). IEEE, may 2015.

[17] W. Burgard, C. Stachniss, M. Bennewitz, G. Grisetti, and K. Arras, "Introduction to mobile robotics," Probalistic Sensor Models, University of Freiburg, pp. 2–3, 2011.

[18] V. Lidar, "High definition real-time 3d lidar," https://velodynelidar.com/products/hdl-64e/, April 2020.

[19] J. Xie, F. Nashashibi, M. Parent, and O. Garcia-Favrot, "A real-time robust slam for large-scale outdoor environments," 2010.
[20] S. Agarwal, keir Mierle, and Others, "Ceres solver," http://ceres-solver.org.
[21] 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). IEEE, 2012.