Evaluation and comparison of 3D lidar based SLAM algorithms

SLt COC Baptiste Valentin

***Abstract*—This article has two main objectives: first, evaluating the performance of recent 3D lidar based SLAM (Simultaneous Localization And Mapping) algorithms; second, contributing actively to the advancement of knowledge in that field. To that end, it briefly explains the underlying theory and draws up an inventory list of 3D SLAM algorithms available on the ROS platform. Based on this list, it selects the two most promising algorithms compatible with 3D lidar sensors and compares their performance in indoor and outdoor environments. To simplify this evaluation, the metrics assessing SLAM quality that were defined in the literature are summed up and most of them are proven to be correlated. Hence, only some of the metrics need to be analyzed before ranking the two algorithms. The best one is then further investigated: improvements that can be brought to this method are looked for. The different improvements are combined in a brand new algorithm that proves to be a significant step forward compared to previous methods regarding SLAM quality, determinism, and computing heaviness. This new package is finally made available to the community.**

***Index Terms*—SLAM, localization, mapping, lidar, odometry, loop closure.**

I. Introduction

ONE hundred years ago, the word *robot* was used for the first time. It referred to the Czech "robota", meaning "forced labor". At this time, robots were a popular science fiction topic and they were thought of as androids (human-like robots) that would behave like assistants. Over the last years, however, significant progress has been made in the field of Robotics and humans have got a different conception of what a robot can be. Nowadays, it is indeed defined by the Cambridge Dictionary as *a machine controlled by a computer that is used to perform jobs automatically* [1]. This current definition embraces not only androids, but also all computer-controlled devices able to execute tasks without human control. The next step is even to make robots fully autonomous (able to make their own decisions). With this in mind, the SLAM (Simultaneous Localization And Mapping) is a capital skill, aiming to keep track of a robot's location within an unknown environment while building a map of it. Thanks to different sensors, a robot receives a large amount of raw data and needs to be able to extract from it the relevant information that will help it to perform SLAM. In general, the SLAM thus requires means (sensors) and methods (algorithms).

This problem has multiple (military and civilian) practical applications since any robot traveling autonomously in an unknown environment needs to solve it. Hence, autonomous vehicles, military reconnaissance and patrol robots, or even space rovers are all equipped (or will be equipped in the near future) with SLAM systems.

The main objective of this work is twofold: first, evaluating the performance of recent 3D lidar based SLAM algorithms; second, contributing actively to the advancement of knowledge in that field. To that end, the organization of the paper will be as follows. The second section will state the SLAM problem and briefly explain how it can be solved. The third section will focus on the 3D SLAM algorithms already present in the Robot Operating System (ROS[1]). The objectives will be to draw up an inventory of the SLAM algorithms implemented and to distinguish them based on the method they use and on the sensor they work with. In the fourth section, all the tools needed to evaluate the performance of SLAM algorithms will be given. Hence, several performance metrics will be defined and some freely available SLAM datasets will be presented. Besides, a correlation analysis will prove that most metrics are correlated, which eases the SLAM quality evaluation. As a fifth section, two recent 3D lidar based SLAM algorithms will be investigated in detail. The intent of this investigation is to select the best algorithm since the end of the article will focus on it. The sixth section will indeed introduce possible improvements to the algorithm chosen; and the seventh section will combine these improvements in a brand new algorithm created in this work. It will be proven to have a superior performance than previously released algorithms.

II. SLAM: Problem Statement

As it has been stated in the introduction, SLAM is a fundamental skill to build autonomous robots. This acronym stands for "Simultaneous Localization and Mapping". To put it in other words, SLAM addresses two problems at the same time: the localization and the mapping problems. Both are part of a wide class of mathematical problems named the *state estimation problems*. Localization indeed intends to estimate the state of a robot (e.g. its position and orientation), whereas mapping aims to estimate the state of its environment (environment representation). Both problems are, individually, considered to be solved in the literature. However, addressing both of them simultaneously remains challenging, since a map is needed to determine the robot's location and (simultaneously) the robot's location is needed to build a map.

Mathematically, SLAM is addressed with a probabilistic terminology. Therefore, the states are represented by means of PDFs (probability density functions) expressing the probability of the states being in certain intervals. More precisely, SLAM estimates the robot's trajectory (described by the pose vector $\mathbf{x}_{0:t}$, with t a terminal time) and builds its map $\mathbf{m}$ at the

[1]https://www.ros.org/

same time. This estimation is based on available data: the robot's *controls* (also known as *odometry*) $\mathbf{u}_{1:t}$ (provided by introceptive sensors, like e.g. wheel encoders) and the robot's *observations* $\mathbf{z}_{1:t}$ (provided by exteroceptive sensors, like e.g. cameras or lidars). Mathematically, the probability density function of the joint events *robot's locations* and *map* should be determined, given the *observations* and the *controls* (1).

$$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \qquad (1)$$

To solve the SLAM problem (1), three main approaches can be used: the first one assumes that both the robot's location and its map are Gaussian (Kalman Filter approaches), the second one assumes that only the map is Gaussian and represents the robot's location hypotheses by means of particles (Particle Filter approaches), and the third one relies on a graphical representation of the SLAM problem (Graph-Based approaches). Besides, other approaches exist, but are less widespread. For example, another approach divides the tasks of SLAM into two algorithms. One algorithm is then responsible for localization, whereas another takes care of mapping. One of the main advantages of this method is its ability to run both algorithms at different frequencies. That way, it could perform localization in real-time and mapping at a lower frequency.

Whatever approach is used to solve the SLAM problem, three phases can always be distinguished in its resolution process. First of all, the relevant information is extracted from the *point cloud* (the raw data generated by the exteroceptive sensors). In a second phase, the robot's trajectory and its map are incrementally built. Unfortunately, this incremental construction is never perfect and causes (1) errors to build up, (2) the uncertainty of the estimates to increase. The third and last phase consists of the *loop closures*. They enable to reduce the uncertainty and the errors of the previous phase, and they mean that the robot is able to recognize places where it has been in the past.

III. 3D SLAM IN ROS

The Robot Operating System, better known by its acronym ROS, is an open-source consistent set of computer tools, whose purpose is to simplify the development of robust robotic software. Its components can be written either in C++ or in Python and the global ecosystem currently claims tens of thousands of users worldwide.

Since SLAM has been one of the most fashionable topics in Robotics over the last 30 years, many researchers developed new techniques or improved existing methods to solve the SLAM problem. Most of them made their ROS-integrated solutions openly available in the form of *packages*, which has pros and cons. On one side, anyone going into SLAM can easily find open-source implementations of the matter. On the other side, due to the large number of available packages, a novice in SLAM can rapidly get lost in the maze of solutions proposed. The purpose of this section is thus to synthesize the most important 3D SLAM packages available in ROS and to clarify in which circumstances they might work. In Table I, each package is thus followed by: the approach used to solve the SLAM problem, the type of exteroceptive sensor it is optimized for, and the presence of a loop closure module. It is important to stress that the following list isn't exhaustive. Besides, it shouldn't be forgotten that ROS packages are upgradeable. That way, some of their characteristics might be modified over time.

Table I: Main ROS packages for 3D SLAM
(M/S/R stand for mono camera / stereo camera / RGB-D sensor, and EKF/GB mean Extended Kalman Filter / Graph-based).

Package	Method	Sensor	Loop closure
ethzasl_icp_mapping	Other	3D Lidar	No
rtabmap_ros	GB	3D Lidar, S, R	Yes
mrpt_slam	Several	3D Lidar	Yes
cartographer_ros	GB	3D Lidar	Yes
LeGO-LOAM	Other	3D Lidar	Yes
SegMap	GB	3D Lidar	Yes
BLAM!	GB	3D Lidar	Yes
hdl_graph_slam	GB	3D Lidar	Yes
MonoSLAM	EKF	M	Yes
ethzasl_ptam	GB	M	No
ScaViSlam	GB	S, R	Yes
rgbdslam(_v2)	GB	R	Yes
lsd_slam	GB	M	Yes
mcptam	GB	M	Yes
orb_slam(2)	GB	M, S, R	Yes
sptam	GB	S	Yes
maplab	GB	M, S, R	Only offline
VINS-Mono	GB	M, S	Yes

As it has been stated in the introduction, the first objective of this paper is to evaluate the performance of recent 3D lidar based SLAM algorithms. To that end, we decided to select two algorithms from the eight ones of Table I that are compatible with 3D lidars and to compare their performance (section V). To make an informed choice, selection criteria have been defined. First, the chosen algorithms must require only a 3D lidar to provide a solution (no introceptive sensor). Such algorithms are able to estimate the robot's *controls* $\mathbf{u_{1:t}}$ based on its observations $\mathbf{z_{1:t}}$. Second, from the remaining algorithms, the two most popular were selected. After some research, *LeGO-LOAM* (2018) [2] and *hdl_graph_slam* (2019) [3] have been chosen. Interestingly, both packages are recent and don't use the same approach to solve the SLAM problem: whereas hdl_graph_slam implements a classical graph-based approach, LeGO-LOAM splits the localization and mapping problems into two parallel threads.

IV. SLAM QUALITY

Evaluating the performance of SLAM algorithms isn't an easy task. This section deals with this problem by (A) presenting diverse SLAM performance metrics, (B) describing some helpful datasets, and (C) proving that most metrics are correlated.

A. Performance metrics

In the literature, several metrics evaluating the SLAM quality are available. Regarding these metrics, two main cases should be distinguished: evaluating SLAM quality when a *ground truth* (an exact solution) is available and evaluating SLAM quality without knowledge about the ground truth. One can already feel that this second case is trickier since the results provided by the SLAM algorithm cannot be compared to a reference solution.

The ten metrics considered in this work are summarized in Table II. The way to compute them won't be explained, but what they represent will be briefly described. The first three metrics compare the trajectory obtained by the SLAM system to a ground truth. That's why their acronym includes the "E" of *error*. That way, **ATE** stands for Absolute Trajectory Error (also called **APE**: Absolute Pose Error), **RTE** stands in for Relative Translation Error, and **RPE** means Relative Pose Error. The fourth metric compares the map obtained by the SLAM algorithm to a ground truth map (**ADNN** stands for Average Distance to the Nearest Neighbor and **CloudCompare** is the name of a program able to compute the distance between two point clouds). The **visual inspection**, as it name suggests, consists in visually analyzing the map/trajectory to detect aberrations and nonsenses. The sixth metric is called the **relative drift** and evaluates the quality of the incremental build without loop closure (section II). Besides, the **consistency** assesses the behavior of the algorithm when it is applied multiple times in the same circumstances. Finally, the last three metrics (**proportion of occupied space**, **number of enclosed areas**, and **corner count**) try to assess the quality of a map without reference solution. In the literature, it is indeed accepted that the smaller these metrics, the better the SLAM quality. It should be noted that they were in the past only used for 2D maps.

Table II: SLAM performance metrics
(ADNN requires an occupancy grid map, CloudCompare a point cloud map).

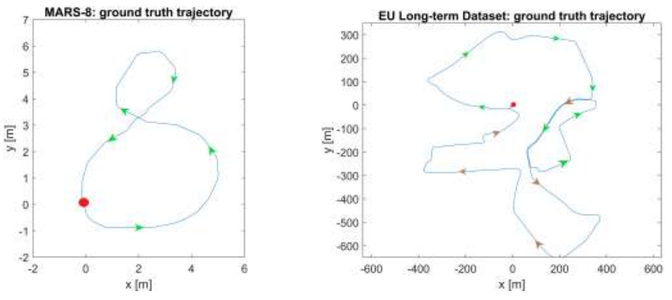
Name	Ground truth?	Evaluated
ATE / APE	YES (traj.)	Traj. quality (absolute)
RTE	YES (traj.)	Traj. quality (relative)
RPE	YES (traj.)	Traj. quality (increments)
ADNN/CloudCompare	YES (map)	Map quality
Visual inspection	No	Map and traj. quality
Relative drift	Yes/No	Traj. quality
Consistency	No	Invariance
Prop. of occ. space	No	Map quality (blurriness)
Num. of encl. areas	No	Map quality (no failure)
Corner count	No	Map quality

B. Datasets

As it is clear from section IV-A, the most trivial way to evaluate a SLAM algorithm is to compare its solution with a ground truth. Hence, two freely available datasets that provide raw measurements acquired by different sensors aboard of a robot, as well as a ground truth will be briefly described. A SLAM algorithm can then be run on the raw measurements, and its solution (map and/or trajectory) can be compared with the reference solution.

The MARS Map dataset [4] is a small indoor dataset, accompanied by a ground truth trajectory and map. The data was recorded in a laboratory by a small wheeled robot equipped with different sensors. In fact, the dataset contains three trajectories, from which MARS-8 ($\approx$ 20 [m] long) has been selected. Its ground truth trajectory is depicted in Fig. 1a, where the red point depicts the start and end positions and the arrows show the direction of motion.

The *EU Long-term dataset* [5] (shortened with EULt) is a large outdoor dataset, accompanied by a ground truth trajectory. The data was gathered in France by a series-produced car equipped with eleven sensors and the total trajectory length is about 5 [km]. The ground truth trajectory is depicted in Fig. 1b, where the red point represents the start and end positions and the arrows show the direction of motion.



(a) MARS-8. (b) EU Long-term.

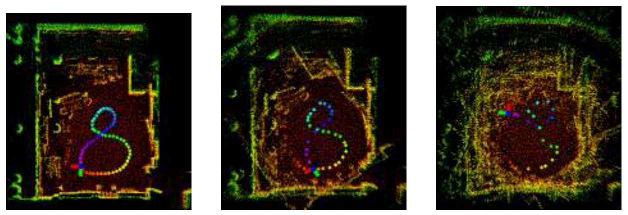
Figure 1: Ground truth trajectories.

C. Correlation

Since most of the metrics of Table II represent related concepts, it seems reasonable that they are correlated in some way. However, this hypothesis has never been confirmed in the literature. This section will thus prove it in a small environment: the MARS-8 dataset. Hence, the performance evaluation of SLAM algorithms will be greatly simplified since fewer metrics will need to be taken into account.

To that end, a parameter able to modify the SLAM quality needed to be found. Experimentally, it has been noticed that the playing rate r of the data of the dataset (e.g. when $r = 2$, the data is played twice as fast as in real-time) influences the SLAM quality, as it is shown in Fig. 2 that depicts top-views of the maps and trajectories obtained.

The next step was to compute the metrics of Table II for different playing rate r and to depict the curves "metric as a function of playing rate" to notice (or not) similar tendencies. It should be noted that this work is the first to generalize the computation of the metrics *proportion of occupied space*, *number of enclosed areas*, and *corner count* to the 3D case. The curves obtained for these three metrics are depicted in Fig. 3, where the trend is clear: the metrics increase with a decreasing SLAM quality. After analysis of the curves of the metrics APE, RPE, relative drift, RTE, and CloudCompare, it can be proven that all these metrics follow the same tendency. Hence, from the ten metrics defined in table II, only two



(a) Playing rate = 1. (b) Playing rate = 10. (c) Playing rate = 15.

Figure 2: MARS-8: SLAM quality modification.

haven't been quantitatively assessed since they're generally not assigned with a value: the visual inspection and the consistency. All the other metrics were proven to increase when the SLAM quality degrades. This proof that the eight numeric metrics are correlated enables to select only one of these metrics and to draw conclusions about SLAM quality based on this single metric.

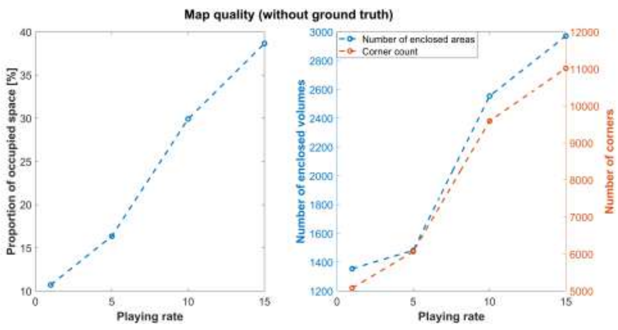


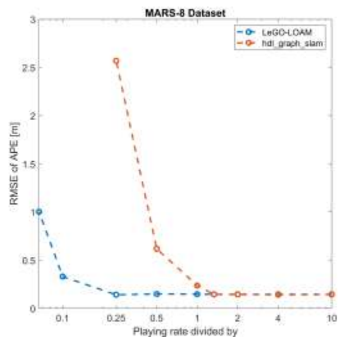
Figure 3: Metrics' correlation.

V. Evaluation

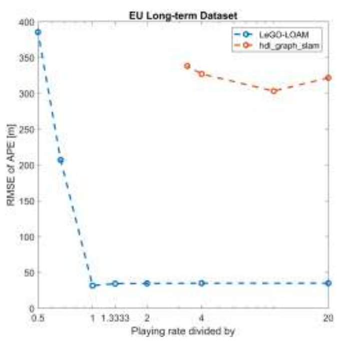
The purpose of this section is to evaluate the performance of LeGO-LOAM and hdl_graph_slam on the two datasets presented in section IV-B while relying on a single numeric metric since it has been proven in section IV-C that they are all correlated. To compare both algorithms, it has been decided to represent the root mean square values of their APE (one of the metrics of Table II) as a function of the reduction of the playing rate of the raw data captured by the sensors. The result is depicted in Fig. 4a (for the indoor dataset) and Fig. 4b (for the outdoor dataset). It should be noted that reducing the playing rate of the data simulates a situation where our computer has more processing power since it receives more time to make computations.

In a nutshell, the conclusions of Fig. 4 are as follows. In the MARS-8 dataset, both LeGO-LOAM and hdl_graph_slam are able of providing solutions of similar quality when the processor has enough time to compute its best solution. However, when the playing rate isn't reduced enough, LeGO-LOAM outperforms hdl_graph_slam. Regarding the EU Long-term dataset, hdl_graph_slam performs worse than LeGO-LOAM whatever the playing rate or parameter setting (this last point has been proven experimentally in our complete master thesis [6]).

All in all, LeGO-LOAM showed more flexibility through our experiments and we can only recommend it. That's why, regardless of the computing power available, LeGO-LOAM is preferred over hdl_graph_slam in both indoor and outdoor environments. Hence, only this algorithm will be further investigated in this article.



(a) MARS-8 dataset.



(b) EU Long-term dataset.

Figure 4: LeGO-LOAM versus hdl_graph_slam.

VI. Improvements

Generally speaking, LeGO-LOAM has been tested in indoor and outdoor environments with satisfactory results. Nevertheless, some behaviors can still be improved.

LeGO-LOAM-BOR [7] (released in 2019) is a first example of a *fork* (a software based on the original LeGO-LOAM) aiming to enhance it. It is available via internet and doesn't fundamentally modify the algorithm. It only adjusts its software implementation with two objectives in mind: improving both the determinism and the efficiency of LeGO-LOAM. After analysis, it can be concluded that LeGO-LOAM-BOR keeps only part of its promises. It was indeed able to increase the determinism and processing speed with respect to LeGO-LOAM. Unfortunately, these improvements come at a cost: a worse quality in a large outdoor dataset (Table III).

SC-LeGO-LOAM [8] (released on the 7th of April 2020) is another fork trying to improve the original LeGO-LOAM. It adds to it a second loop closure detector (called *Scan Context* [9]) that would be able to correct for large drifts. It should be noted that the original loop closure module of LeGO-LOAM wasn't able to do this since it worked according to the principle of *radius-search* (which means that it looked for loop closures only in a sphere around the current pose estimate). To test this new feature, a new situation has been imagined: the dataset EU Long-term can be repeated (executed two times) to simulate a situation with a large drift. This condition is depicted in the "EULt (2x)" column of Table III.

To sum it up, Table III (where LL stands for LeGO-LOAM) presents an overview of the performance of the three presented packages on the different datasets. The last column displays the speed at which the data is processed in order to yield the best solution achievable in the case of the "EULt (2x)" dataset, which means that it represents the efficiency of the algorithms. In that respect, four criteria (the four columns of Table III) should help us to select the best algorithm. To that end, a green (red) color is given to the best (worse) result for each criterion. After analysis, it is clear that the situation is far from perfect: each algorithm is the best one in (at least) one criterion, but also the worse one in (at least) another criterion. Hence, none of the presented packages has found the right balance and no algorithm outperforms the others. That's why the next section will intend to create a new package combining the advantages of SC-LeGO-LOAM and LeGO-LOAM-BOR.

Table III: RMS value of the APE and efficiency (a).

Dataset	MARS-8	EULt	EULt (2x)	Efficiency
LL	0.148 [m]	34.5 [m]	57.8 [m]	1
LL-BOR	0.134 [m]	53.9 [m]	47.0 [m]	2.7
SC-LL	0.154 [m]	39.4 [m]	22.3 [m]	0.5

VII. Our package

After the mixed results of the previous section, the next logical step is to combine both packages SC-LeGO-LOAM and LeGO-LOAM-BOR in a single brand new package SC-LeGO-LOAM-BOR, as it is shown in Fig. 5. This has been made by thoroughly analyzing the C++ code of both packages and by combining right parts of it.

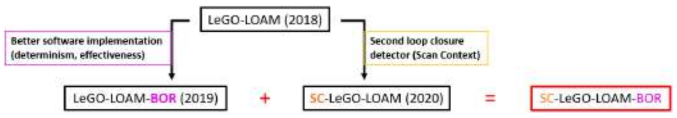

LeGO-LOAM (2018)
Better software implementation (determinism, effectiveness)
Second loop closure detector (Scan Context)
LeGO-LOAM-BOR (2019)
+
SC-LeGO-LOAM (2020)
=
SC-LeGO-LOAM-BOR

Figure 5: Overview of SC-LeGO-LOAM-BOR.

The performance of SC-LeGO-LOAM-BOR are added to Table III to yield Table IV, where a sea green color is assigned to the second best result for each criterion. Whereas Table III didn't enable to select the best algorithm, Table IV certainly does. The brand new SC-LeGO-LOAM-BOR appears indeed to be the best or second best algorithm regarding all criteria. The purpose of this section has thus been achieved since SC-LeGO-LOAM-BOR combines the advantages of LeGO-LOAM-BOR (computing efficiency, determinism, and good performance on the MARS-8 dataset), and the ones of SC-LeGO-LOAM (a second loop detector that improves the SLAM quality on the repeated EU Long-term dataset).

After the creation of a new SLAM algorithm, the tradition generally followed is to evaluate its performance on a dataset that has been used for multiple years. Such a dataset can indeed summarize the results provided by diverse SLAM algorithms over the years, which makes it very easy to compare them. Regarding SLAM evaluation, the best known and most widespread dataset is *The KITTI Vision Benchmark Suite* [10]. Hence, LeGO-LOAM and SC-LeGO-LOAM-BOR were applied on the 11 sequences of it, with resulting mean RTE (Table II) equal to, respectively, 0.73 [%] and 0.66 [%]. This proves on a third different dataset the superior performance of our new algorithm.

Finally, the package SC-LeGO-LOAM-BOR was implemented on a ground robot of the Royal Military Academy (RMA) that is equipped with a Velodyne-HDL32E 3D lidar. The purpose of this final part was to show that the use of this new algorithm on a real robot is straightforward and that its results are as good as expected. For the record, the top view of the map obtained when the robot is driven on the second floor of the Department of Mechanics at the RMA is depicted in Fig. 6, where the white points are the trajectory and one can recognize an office in the top left corner, a class room in the middle, and different corridors.

Table IV: RMS value of the APE and efficiency (b).

Dataset	MARS-8	EULt	EULt (2x)	Efficiency
LL	0.148 [m]	34.5 [m]	57.8 [m]	1
LL-BOR	0.134 [m]	53.9 [m]	47.0 [m]	2.7
SC-LL	0.154 [m]	39.4 [m]	22.3 [m]	0.5
SC-LL-BOR	0.134 [m]	37.0 [m]	24.3 [m]	2.5

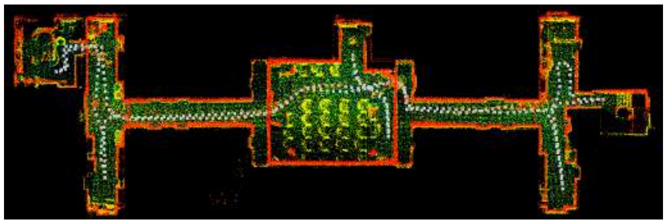


Figure 6: Top-view of the map yielded by SC-LeGO-LOAM-BOR.

VIII. Conclusion

This article had two main objectives: evaluating the performance of recent 3D lidar based SLAM (Simultaneous Localization And Mapping) algorithms, and contributing actively to the advancement of knowledge in this field. SLAM describes the problem of localizing a robot in an unknown environment while building a map of it and has multiple practical applications since any robot traveling autonomously in an unknown environment needs to solve it.

The main contributions of this paper may be summarized as follows: the creation of a list recapitulating the ROS-integrated 3D SLAM packages, the computation of performance metrics that hadn't been generalized yet to the 3D case, the proof that a large number of metrics are correlated, the evaluation of two recent SLAM algorithms, the presentation of possible improvements to the best algorithm (from which one was published in 2020) and the creation of a brand new package combining the advantages of previously released algorithms. This new method (called SC-LeGO-LOAM-BOR) has finally been released online to make it available to the scientific community.

References

[1] The Cambridge English Dictionary, "Robot," https://dictionary.cambridge.org/dictionary/english/robot, accessed: February 26, 2020.
[2] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," 10 2018, pp. 4758–4765.
[3] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419841532, 2019. [Online]. Available: https://doi.org/10.1177/1729881419841532
[4] H. Chen, X. Zhao, J. Luo, Z. Yang, Z. Zhao, H. Wan, X. Ye, G. Weng, Z. He, T. Dong, and S. Schwertfeger, "Towards generation and evaluation of comprehensive mapping robot datasets," 2019.
[5] Z. Yan, L. Sun, T. Krajnik, and Y. Ruichek, "EU long-term dataset with multiple sensors for autonomous driving," *CoRR*, vol. abs/1909.03330, 2019. [Online]. Available: http://arxiv.org/abs/1909.03330
[6] B. Valentin, "Evaluation and comparison of 3d lidar based slam algorithms," Master's thesis, Royal Military Academy, 2020.
[7] D. Faconti, "Lego-loam-bor," https://github.com/facontidavide/LeGO-LOAM-BOR, accessed: March 23, 2020.
[8] G. Kim, "Sc-lego-loam," https://github.com/irapkaist/SC-LeGO-LOAM, accessed: March 23, 2020.
[9] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Oct. 2018.
[10] 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.