2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Real-time 3D Map Update at Point-level for LiDAR-based Localization in Changing Environments

Yuhei Oshikubo[1], Sarthak Pathak[2], Yonghoon Ji[3], and Kazunori Umeda[2]

***Abstract*— In this paper, a novel framework for real-time localization through pre-built map update by 3D LiDAR mounted on a robot is proposed. Autonomous mobile robots usually use high-precision 3D maps (i.e., HD maps) built in advance. However, as time passes, the environmental map changes from the actual environment, adversely affecting autonomous navigation. In addition, it is costly to recreate an HD map every time the environment changes. Therefore, it is necessary to update environmental maps simply and frequently. We propose a method for real-time map updating in the form of point clouds and validate the results of integrating map update with localization.**

I. INTRODUCTION

3D point cloud maps are essential for the localization of autonomous mobile robots. Point cloud localization is typically performed by scan matching of a point cloud acquired by a sensor (e.g., LiDAR, RGB-D camera) with a pre-built point cloud map. Conventional localization assumes that there are no significant differences between the respective point clouds. Since changes in the environment are unavoidable during long-term operations, a difference between the environmental map and the actual environment may be judged unstable due to low matching even though the pose estimation is accurate as shown in Fig. 1. Therefore, pre-built point cloud maps need to be updated regularly. However, the cost of manually building environmental maps every time a change occurs is large. There have been many studies on updating environmental maps in both 2D and 3D to solve this problem. The main methods for updating the point cloud map are pose graphs [1], [2], geometric methods [3], occupancy grids [4], [5], and a combination of them [6], [7]. Update based on pose graphs or geometric methods may result in fault in measurements that include occlusions. In other words, most of these methods assume that it is possible to obtain data for a large area of the map to be managed every time. On the other hand, it is necessary to consider a framework to update the map during the daily operation of the robot to reduce the cost of map updating. Therefore, we consider the occupancy probability of the point cloud map and the light path from the LiDAR to ensure robustness in measurements with occlusion. The idea is similar to OctoMap [8], a type of occupancy grid. We extend the method to the point level to maintain the point cloud format. Updating the map at the point level not only improves the accuracy of localization, but also can be expected to be used for a variety of purposes such as measurement and construction management.

The contributions of this study are as follows:

- We propose a new framework to update 3D point clouds easily and in real time and use them for localization. By considering the confidence score known as occupancy probability at the point level, it is robust to occlusion and dynamic environments and can be updated in detail. In addition, we achieved real-time updating of point cloud maps.
- We achieve robust localization in changing environments by the proposed method.

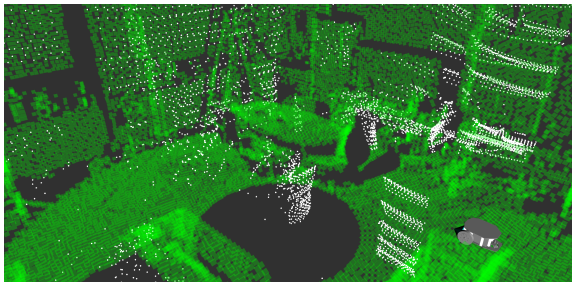


Fig. 1. Degradation of a pre-built point cloud map. The green points are pre-built point cloud map and the white points are from LiDAR mounted on a robot. Some objects were moved, deleted, or added compared to when the map was built in advance.

II. MAP UPDATE AND LOCALIZATION

A. Overview

Fig. 2 shows an overview of the proposed framework in a single session. Our framework can be classified into two sections: localization and point cloud map update. This study focuses on map update and assumes that localization is performed correctly and that the initial pose is known.

Point cloud map update has three phases: initialization, front-end processing and back-end processing. Initialization gives a confidence score to the point cloud with only geometric information. The front-end process detects changes and updates the confidence score of each point as shown in the Fig. 3. In the back-end processing, the updated point cloud map is passed to the localization side. Localization is performed using the updated point cloud map. The last updated point cloud for a given session is used as the pre-built point cloud map for the next session.

[1]Precision Engineering Course, Graduate School of Science and Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, Japan. oshikubo@sensor.mech.chuo-u.ac.jp

[2]Department of Precision Mechanics, Faculty of Science and Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, Japan. {pathak, umeda}@mech.chuo-u.ac.jp

[3]Graduate School for Advanced Science and Technology, Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi-shi, Ishikawa, Japan. ji-y@jaist.ac.jp

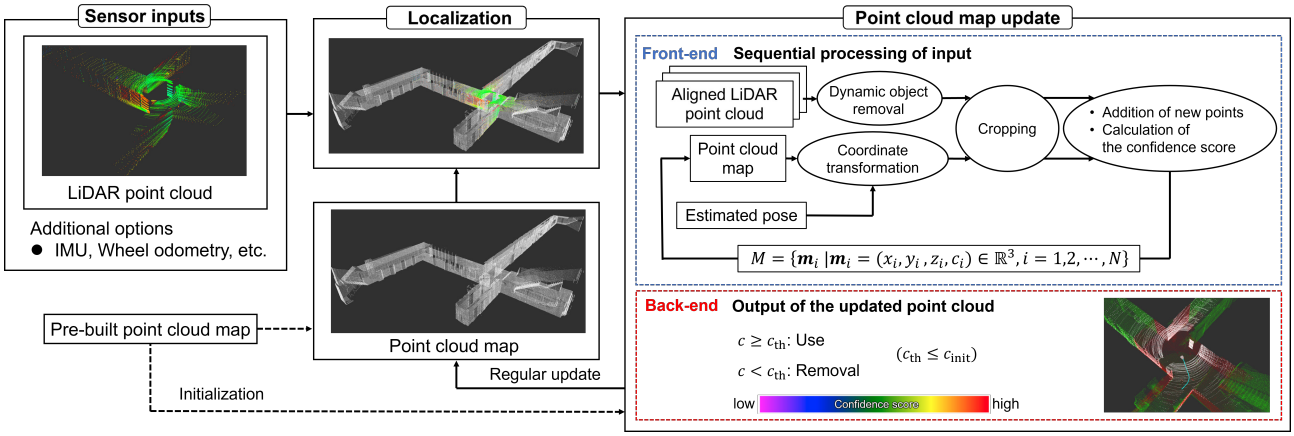


Fig. 2. Overview of localization and point cloud map update in a single session

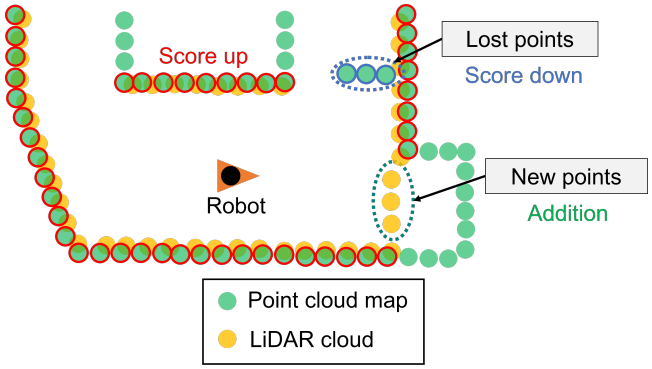
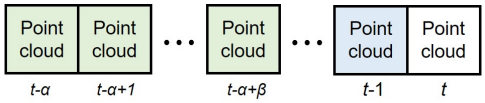
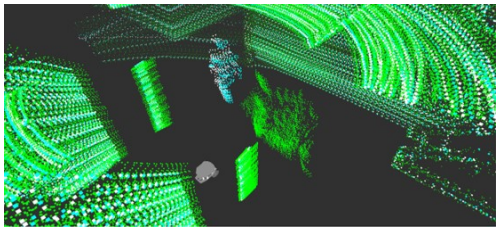


Fig. 3. Sequential processing of input



(a) Buffers of sensor point cloud



(b) Visualization results for buffers of LiDAR point clouds

Fig. 4. Differential detection of LiDAR point clouds

B. Initialization of point cloud map

If the number of points in the point cloud map $\mathcal{M}$ is N and the point with confidence score c_i is $\boldsymbol{m}_i$, the map can be expressed as follows:

$$\mathcal{M} = \{\boldsymbol{m}_i \mid \boldsymbol{m}_i = (x_i, y_i, z_i, c_i) \in \mathbb{R}^3, i = 1, 2, \ldots, N\} \tag{1}$$

Here, $0 \leq c_i \leq 1$, c_{init} is the initial value, and c_{th} is the threshold value that determines the update. Note that $c_{\mathrm{th}} \leq c_{\mathrm{init}}$. This assumption prevents unwanted erasure of point clouds in occlusion areas. In this study, c_{th} and c_{init} are set as 0.5.

C. Dynamic object removal in LiDAR point clouds

In this subsection, we describe the process of removing dynamic points from a LiDAR point cloud. When a robot performs its daily activities, it may be surrounded by dynamic objects such as humans. Generally, dynamic objects are not suitable for localization; thus, it is necessary to remove them from the point cloud map. In this study, we use a differencing method to detect dynamic points. However, difference detection is difficult because the number of point clouds in a frame obtained from a LiDAR is sparse, and the changes between consecutive frames are small. Therefore, as shown in Fig. 4(a), the point cloud from the previous frame α to the current frame t are stored, and the green point cloud accumulated for $\beta + 1$ frames based on the previous frame α are used for difference detection from the point cloud in the current frame shown in white. The blue points are the points from one frame prior to the current frame and have changed little compared to the white points. Fig. 4(b) shows the result of visualizing each point cloud. We used the Octree [9] structure of the Point Cloud Library (PCL) [10] for difference detection. Here, the LiDAR point clouds with no overlap are removed as dynamic points. The LiDAR point cloud in the overlapped area is treated as a candidate to be added to the point cloud map explained in the next subsection.

D. Addition of new points to the point cloud map

This subsection describes the process of adding a new point cloud to the point cloud map. In the candidate sensor point cloud to be added to the map point cloud obtained in the previous subsection, points that already exist in the point cloud map do not need to be added. If the pose estimation results are approximately correct, it can be determined by comparing the point cloud map and the sensor point cloud. Specifically, when there is no overlap in both point clouds, it is considered as a new emergent point. We use Octree to compare the point cloud map converted to the sensor

coordinate system and the sensor point cloud with dynamic points eliminated. Furthermore, the overlapping points are used for the next process explained in the next subsection.

E. Calculation of the confidence score of each point in the point cloud map

When updating the map point cloud, points that have not changed should remain unchanged. First, we discuss how to find points that increase the confidence score in the map point cloud. As mentioned above, if the pose estimation results are almost correct, the same area can be identified as unchanged by measuring it in both the map and sensor point clouds. Using this phenomenon, we increase the confidence score of each overlapping point cloud map based on the binary Bayes filter [11].

Next, we describe a method for finding missing points. It is important to consider occlusion when determining the lost points. In general, it is possible to use volumetric voxel grids to construct maps that are robust to occlusion; however, the processing is known to be heavy [12]. Hence, we consider the path of the rays from the LiDAR without volumetric voxel grids. In an ideal environment, if LiDAR points are exsisted beyond the point cloud map in the region along the ray cast from the LiDAR, it indicates a loss of the point cloud map. Using this phenomenon, the points of the point cloud map are determined to be unoccupied by comparing the centroid of each point cloud at arbitrary angles in a spherical coordinate system. This allows for fast update without changing the format of the point cloud map. Fig. 5 is a schematic of the comparison made in the sensor coordinate system to determine the point cloud map of disappearance. The method consists of the following steps:

1) Each point cloud (i.e., the point cloud map converted to the sensor coordinate system and the sensor point cloud with dynamic points eliminated) is trimmed.
2) Each point cloud is stored for each arbitrary angle θ, ϕ.
3) The centers of gravity of the point cloud of the map and the sensor point cloud for each arbitrary angle are calculated.
4) The distances r_{map} and r_{sensor} from the sensor to each center of gravity are obtained.
5) When $r_{\mathrm{sensor}} - r_{\mathrm{map}}$ is higher than r_{th}, the point cloud of map is considered to have disappeared.

The formula to update the confidence score c_i^t that the i-th point in frame t is as follows:

$$c_i^t = p(\boldsymbol{m}_i \mid o_i^t) = 1 - \left(\frac{1}{1 + \exp(l_i^t)} \right) \tag{2}$$

$$l_i^t = l_i^{t-1} + \ln \left(\frac{p(\boldsymbol{m}_i | o_{i,t})}{1 - p(\boldsymbol{m}_i | o_{i,t})} \right) \tag{3}$$

where o_i^t indicates whether the i-th point exists up to frame t, $o_{i,t}$ indicates whether the i-th point in frame t exists or not and l_i^t is log odds of the probability that the i-th point exists in frame t.

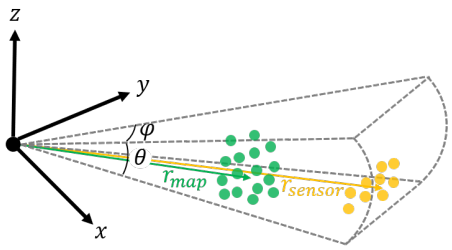


Fig. 5. Comparison of map point clouds and sensor point clouds at arbitrary angles in a spherical coordinate system

F. Output of the updated point cloud

The point cloud map is published for the localization side by considering the confidence score c of each point. Namely, points that are higher than the c_{th} threshold are kept, and points that are lower than the c_{th} threshold are removed. As described in subsection II-B, by setting c_{th} to a value less than or equal to c_{init}, the original point cloud map information can be retained for areas where no measurements were taken. The timing of updating the point cloud map can be changed as necessary. In this study, the map point cloud used for pose estimation is updated every 10 times the confidence score is calculated.

G. Localization

In this study, the point cloud map is regularly updated by customizing a package [13] in which normal distributions transform(NDT) scan matching [14] is implemented for localization.

III. EXPERIMENTS

A. Experimental Setup

To verify the effectiveness of the proposed method, the point cloud map was updated by running the robot at locations where changes have occurred since the pre-built point cloud map was created, as shown in Fig. 6. The pre-built point cloud map was created by FARO Focus M70. In experiments, we evaluated whether the point cloud map was updated properly by the proposed method, its real-time performance and effects on localization under the two conditions listed in the Tbl. I. The updated point cloud map was evaluated qualitatively. The effects of real-time update on localization was evaluated quantitatively using the absolute pose error(APE) by evo [15] and transformation probability(TP) calculated by NDT scan matching. APE evaluates the accuracy of localization by comparing the estimated position to the ground truth. TP measure the likelihood whether the computed transformation between two point clouds or scans is accurate. A higher TP indicates a higher confidence in the estimated transformation, suggesting that the two scans have been matched well. This indicator is used to verify whether accurate localization is being achieved, and if this value is not sufficiently high, the robot may stop for operational safety even if the pose is correct. Note that high value does not guarantee that correct pose estimation is being peformed. As shown in Fig. 7, a robot equipped LiDAR was

used in the experiment. A Megarover Ver. 3.0 from Vstone and a Livox mid-360 were used for the robot and LiDAR. This was controlled using a laptop computer mounted on the robot. The motion capture system was used to obtain ground truth pose. The motion capture camera was an OptiTrack V120 Trio and the acquired pose were communicated with the laptop using the ROS driver for the NatNet protocol [16].

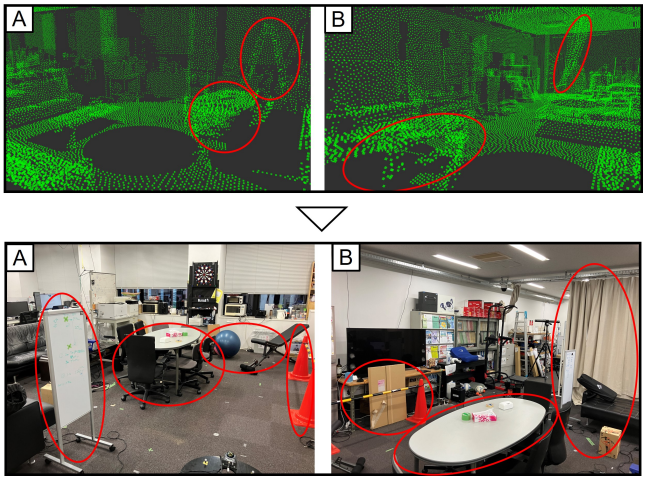


Fig. 6. Two snapshots of the changing environments. Red circles indicate where changes occurred. For example, changes include opening a curtain, repositioning a desk, removing a ladder, and adding a vertical panel.

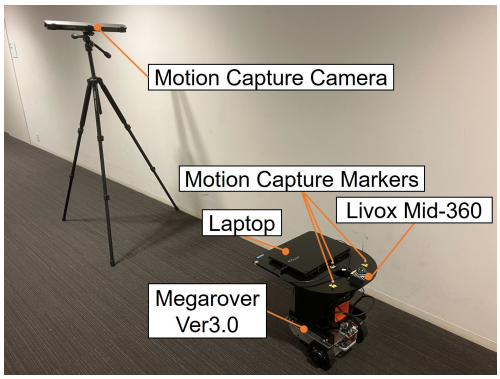


Fig. 7. Experimental devices

B. Results

Fig. 8 is the result of updating the 3D point cloud. The map in the figure was downsampled using a 0.05 m voxel grid. The results show that the point cloud can be added and removed approximately. Fig. 9 shows the transform probability in the proposed method and the conventional method (without map update) while the robot is navigating with different parameters. It indicates that the transform probability of the proposed method increases a few seconds after the robot starts running. Therefore, it can be seen that real-time updating of the point cloud map improves the robustness of the estimation. Fig. 10 shows the APE for each method. This indicates that the proposed method performed slightly better overall. Due to the fact that there were no significant changes in this experiment. In other words, in an environment with a sufficient amount of static geometric features, updates some of the changes contribute little to the accuracy of localization, but lead to robustness. Furthermore, the average computation times per cycle for map updates using the Intel Core i7-12700H CPU were as follows: Condition 1, 151.25 ms; Condition 2, 115.49 ms. This indicates that real-time performance was achieved.

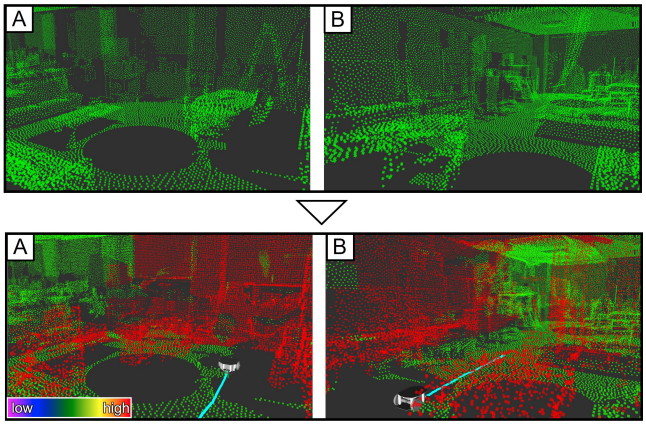
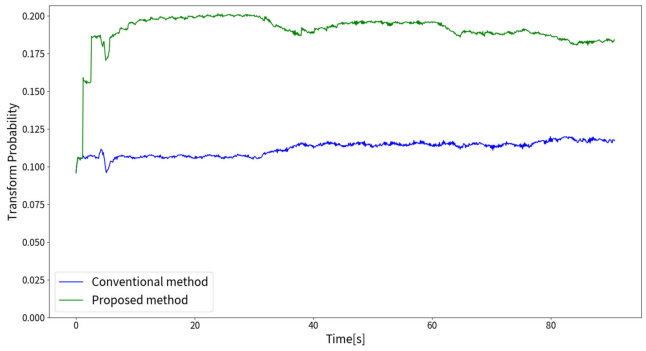
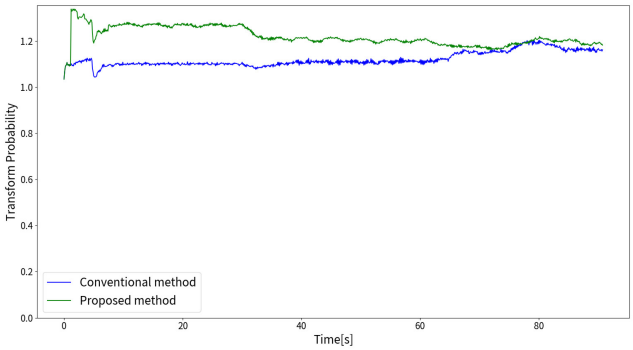


Fig. 8. Results of point cloud map update. The color of the points represents the confidence score.The red points indicate that LiDAR measurements were taken in this experiment and that the confidence score is high. Points with confidence score c lower than c_{th} were removed. The light blue trajectory is the estimated path of the robot.



(a) Condition 1

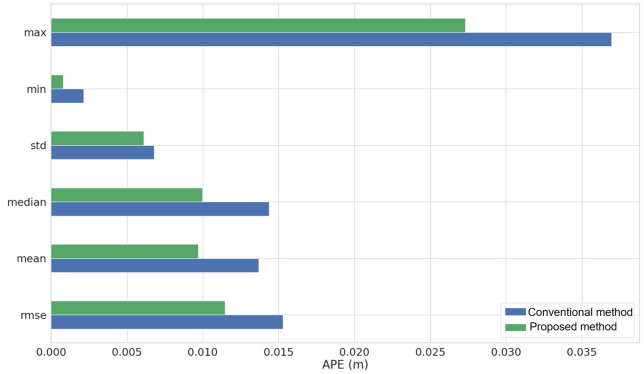


(b) Condition 2

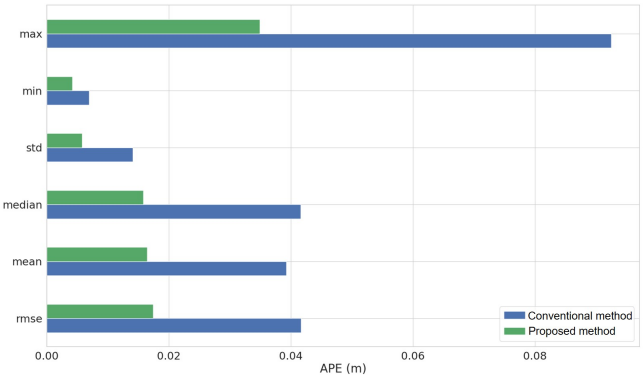
Fig. 9. Transform probability. Each condition is described in Tbl. I

Condition	Voxel Size for Downsampling (Map)	ND Voxel Size (NDT Scan Matching)	Downsampling for LiDAR cloud
1	0.05 m	0.2 m	None
2	0.1 m	0.5 m	None

TABLE I Parameters for each of the conditions



(a) Condition 1



(b) Condition 2

Fig. 10. The evaluation of absolute pose error (APE) using evo [15]. Each condition is described in Tbl. I

IV. CONCLUSIONS

In this paper, we proposed a framework for updating 3D point cloud maps in real-time using 3D LiDAR mounted on a robot and verified by experiments. Specifically, we made it possible to update the point cloud map by considering the occupancy probability of each point in the map point cloud. Our results show that the proposed method correctly updates the map in real time and contributes to improving the reliability of localization. In the future, we will consider making the system work in environments where there are larger changes.

REFERENCES

[1] Min Zhao, Xin Guo, Le Song, Baoxing Qin, Xuesong Shi, Gim Hee Lee, and Guanghui Sun, "A General Framework for Lifelong Localization and Mapping in Changing Environment", *Proc. of International Conference on Intelligent Robots and Systems*, pp. 3305-3312, 2021.

[2] Bin Peng, Hongle Xie, and Weidong Chen, "ROLL: Long-Term Robust LiDAR-based Localization With Temporary Mapping in Changing Environments", *Proc. of International Conference on Intelligent Robots and Systems*, pp. 2841-2847, 2022.

[3] Kun Liu, Jan Boehm, and Christian Alis, "Change Detection of Mobile LiDAR Data Using Cloud Computing", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. LI-B3, pp. 309-313, 2016.

[4] Georgios Tsamis, Ioannis Kostavelis, Dimitrios Giakoumis, and Dimitrios Tzovaras, "Towards Life-long Mapping of Dynamic Environments Using Temporal Persistence Modeling", *Proc. of International Conference on Pattern Recognition*, pp. 10480-10485, 2020.

[5] Liudi Yang, Sai Manoj Prakhya, Senhua Zhu, and Ziyuan Liu, "Lifelong 3D Mapping Framework for Hand-held & Robot-mounted LiDAR Mapping Systems", in *Robotics and Automation Letters*, 2024.

[6] Erik Einhorn and Horst-Michael Gross, "Generic 2D/3D SLAM with NDT Maps for Lifelong Application", *Proc. of European Conference on Mobile Robots*, pp. 240-247, 2013.

[7] Giseop Kim and Ayoung Kim, "LT-mapper: A Modular Framework for LiDAR-based Lifelong Mapping", *Proc. of International Conference on Robotics and Automation*, pp. 7995-8002, 2022.

[8] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees", *Autonomous Robots*, vol. 34, no. 3, pp. 189-206, 2013.

[9] Donald Meagher, "Geometric Modeling Using Octree Encoding", *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129-147, 1982.

[10] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)", *Proc. of International Conference on Robotics and Automation*, pp. 1-4, 2011.

[11] John Mullane, Martin D. Adams and Wijerupage Sardha Wijesoma, "Robotic Mapping Using Measurement Likelihood Filtering", *International Journal of Robotics Research*, vol. 28, no.2, 2009.

[12] Heajung Min, Kyung Min Han, and Young J. Kim, "OctoMap-RT: Fast Probabilistic Volumetric Mapping Using Ray-Tracing GPUs", *Robotics and Automation Letters*, vol. 8, no. 9, pp. 5696-5703, 2023.

[13] AbangLZU, "A simple, clean NDT licalization ROS package", `https://github.com/AbangLZU/ndt_localizer`

[14] Peter Biber and Wolfgang Strasser, "The Normal Distributions Transform: A New Approach to Laser Scan Matching", *Proc. of International Conference on Intelligent Robots and System*, vol.3, pp. 2743-2748, 2003.

[15] Michael Grupp, "evo: Python package for the evaluation of odometry and SLAM", `https://github.com/MichaelGrupp/evo`, 2017.

[16] Aarsh Thakker, "Seamless Integration of Optitrack Motion Capture with ROS" , *Conférence annuelle des développeurs et utilisateurs ROS*, `https://github.com/L2S-lab/natnet_ros_cpp`, 2022.