Dynamic Environment Robust 3D LiDAR Based Monte Carlo Localization

1st Junhyung Kim
Mechanical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
nuninu98@kaist.ac.kr

2nd Junwoo Park
Mechanical Engineering
Korea Advanced Institute of Science and Technology
Daejeon, Republic of Korea
junoop@kaist.ac.kr

***Abstract*—This paper represents the methodology and test result of Dynamic Environment Robust 3D LiDAR Based Monte Carlo Localization. We applied IMU preintegration motion model and 3D grid submap for 3D pose tracking using the particle filter. The proposed method showed robustness even in the case of deformed environment, which does not fit exactly with the pre-built map.**

I Introduction

Localization is essential for Autonomous Mobile Robot (AMR)'s operation. For the last few decades, localization algorithm has been under research. Methods using sensors such as LiDAR, camera has been one of the most common methods for AMR's localizatiom. 3D LiDAR provides laser scanned environmental data. Due to it's data density and high accuracy, scan matching based 3D LiDAR based localization has been widely used for AMR's 3D pose tracking. Shan et al. (2018) represented LeGO-LOAM [1], which uses feature extraction and matching from the 3D LiDAR data. It extracts planar and edge feature for scan matching. Koide et al.(2021) suggests VGICP algorithm [2], which is based on the Generalized Iterative Closest Point (GICP) matching. However, these scan matching based localization might be degraded in the case of dynamic environment, because it assumes the surroundings are static. To overcome, we suggests the 3D LiDAR based Monte Carlo Localization(MCL) [3] for the non-static environment. By applying 3D particle filter, our method stands robust even in the case that environment varies from the pre-built map.

II Methodology

Our method is composed of two parts: front-end and back-end. Front-end tracks the robot's pose using the particle filter [3] to provide Monte Carlo Localization. Particle filter requires high computational cost because it uses the swarm of particles to estimate the robot pose's probability distribution. In addition, 3D LiDAR releases dense point cloud data, which leads to more burden for computation. For real time operation, proposed method uses 3D grid submap as a sensor measurement model. Moreover, the submap is generated in the back-end module which operates parallel to the front-end, so that the submap generation does not affect the pose tracking. Fig. 1 shows the overall structure of the proposed method. The following sections explain the details about the prediction step by the motion model, and the particle resampling step by the sensor model of the particle filter.

A. Prediction

Robot and particle's position in the map is represented with 6 Degrees of Freedom (DoF) vector, as described in (1),

$$p(t) = (x, y, z, \phi_x, \phi_y, \phi_z)^T \tag{1}$$

where (x, y, z) is robot's position in x, y, z coordinate, (ϕ_x, ϕ_y, ϕ_z) is orientation of the robot in the map coordinate, in Rodrigues rotation. The equation of motion is based on the IMU preintegration [4] model. Since we are using the wheel odometry for motion, the acceleration and bias for IMU is ignored. The simplified equation of motion is described in (2) to (8).

$$R(t) = Exp(\begin{bmatrix} 0 & -\phi_z(t) & \phi_y(t) \\ \phi_z(t) & 0 & -\phi_x(t) \\ -\phi_y(t) & \phi_x(t) & 0 \end{bmatrix}) \tag{2}$$

$$\boldsymbol{\omega} = \hat{\boldsymbol{\omega}} + N(\mathbf{0}, \Sigma_1) \tag{3}$$

$$\mathbf{v} = \hat{\mathbf{v}} + N(\mathbf{0}, \Sigma_2) \tag{4}$$

$$R(t+\Delta t) = R(t)Exp(\begin{bmatrix} 0 & -\boldsymbol{\omega}_z & \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z & 0 & -\boldsymbol{\omega}_x \\ -\boldsymbol{\omega}_y & \boldsymbol{\omega}_x & 0 \end{bmatrix} \Delta t) \tag{5}$$

$$\begin{aligned} \boldsymbol{\tau}(t+\Delta t) &= \boldsymbol{\tau}(t) + \boldsymbol{v}\Delta t \\ \boldsymbol{\tau}(t) &= (x(t), y(t), z(t))^T \end{aligned} \tag{6}$$

$$\begin{bmatrix} 0 & -\phi_z(t+\Delta t) & \phi_y(t+\Delta t) \\ \phi_z(t+\Delta t) & 0 & -\phi_x(t+\Delta t) \\ -\phi_y(t+\Delta t) & \phi_x(t+\Delta t) & 0 \end{bmatrix} = Log(R(t+\Delta t)) \tag{7}$$

$$p(t+\Delta t) = (\boldsymbol{\tau}(t+\Delta t)^T, \phi_x(t+\Delta t), \phi_y(t+\Delta t), \phi_z(t+\Delta t))^T \tag{8}$$

where $\hat{\boldsymbol{\omega}}$ and $\hat{\mathbf{v}}$ are measured angular velocity and linear velocity by wheel encoder, $\boldsymbol{\omega}$ and $\mathbf{v}$ are the angular velocity and linear velocity with random Gaussian noise. Each particle's pose is updated with the motion model with random noise to represent the motion uncertainty.

B. Resampling

Since our system is based on 3D LiDAR and MCL, the computation cost is expensive. Therefore, an efficient measurement model for 3D LiDAR is necessary for real time pose tracking. In our method, a 3D grid submap (submap from below) is applied for fast calculation. Submap is composed of the origin, and the 3 dimensional grid array. Each grid in the array represents whether the space is occupied by pre-built map data or not. The origin is a position of the first grid in an array, in the map coordinate. The submap's resolution, x, y, z directional size is defined as a constant. In our method, 100m, 100m, and 40m for x, y, z directional size are used, and the resolution is 0.1m. Smaller resolution can improve the localization accuracy, but more memory is required. The 3D grid submap is able to project the LiDAR point data into the grid in a constant time complexity, the overall time complexity is O(N), where N is a number of points in LiDAR point cloud.

The submap is generated as the traveled distance from the last submap generated pose exceeds a certain threshold. The submap is generated by following Algorithm 1. Generated submap is used for particle weight calculation. For each particle, we transform the 3D LiDAR data using the particle's pose and project to the submap. Then, the particle's weight is calculated as the sum of grids' weight that transformed 3D LiDAR data points projected on. Algorithm 2. shows the sequence for the weight calculation of particles. After the weight calculation, resample the particle according to each particles' weight distribution. Particles are duplicated proportion to it's weight distribution. Least variance sampler [5] is applied for our method. In short summarize, it decides which particle and it's duplication times by using the particles' weight cumulative sum and distribution. The number of particles remains same before and after the resampling.

Algorithm 1 Submap Generation using Pre-built Point Cloud Map

Require: M and P: The pre-built point cloud map and robot's position in SE(3) matrix. $\mathbf{S} = (s_x, s_y, s_z)$: submap size in x, y, z direction. r : submap resolution
Ensure: m: submap output.

$\boldsymbol{\tau} = get\_translation(P)$
$m.origin = \boldsymbol{\tau} - 0.5 * (s_x, s_y, s_z)^T$
$m.array = zeros(s_x/r, s_y/r, s_z/r)$
for each point **pt** in M **do**
  $\mathbf{pt} = transform\_point\_to\_map\_frame(pt, P)$
  $(n_x, n_y, n_x)^T = round((\mathbf{pt} - origin)/r)$
  if $0 \leq n_x < s_x \&\& 0 \leq n_y < s_y \&\& 0 \leq n_z < s_z$ **then**
    $m.array[n_x][n_y][n_z] = m.array[n_x][n_y][n_z] + 1$
  end if
end for
return m

Algorithm 2 Particle Weight Calculation

Require: $[P]$: Set of the particles. T_{LR} : 3D LiDAR's relative pose to the robot(SE(3) matrix). m: submap
Ensure: m: particles' weight.

for each particle p_i in $[P]$ **do**
  $\mathbf{pt} = transform\_point\_to\_map\_frame(pt, P)$
  $(n_x, n_y, n_x)^T = round((\mathbf{pt} - origin)/r)$
  if $0 \leq n_x < s_x \&\& 0 \leq n_y < s_y \&\& 0 \leq n_z < s_z$ **then**
    $m.array[n_x][n_y][n_z] = m.array[n_x][n_y][n_z] + 1$
  end if
end for
return m

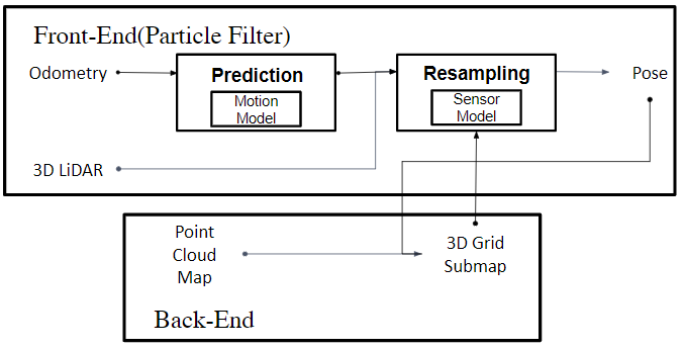

Fig. 1. Overall structure of the proposed method. Front-End is a particle filter that tracks the robot pose in real time, while back-end generates 3D grid submap. Tracked pose is used to make submap, and submap is used for sensor measurement model to track the pose.

III Result

A. Environmental Setting

We tested our method in the Gazebo simulator environment [6], as shown in Figure 2. There are large two-story shelves placed on either side of the warehouse, with smaller shelves and stacks of boxes scattered in the middle. Several pedestrian are moving around, creating an environment similar to a real logistics warehouse. The point cloud map was first generated using LIO-SAM [7], as shown in Figure 4. Then, some structures in the test field were removed so that the experiment environment would not perfectly match the map, as shown in Figure 3

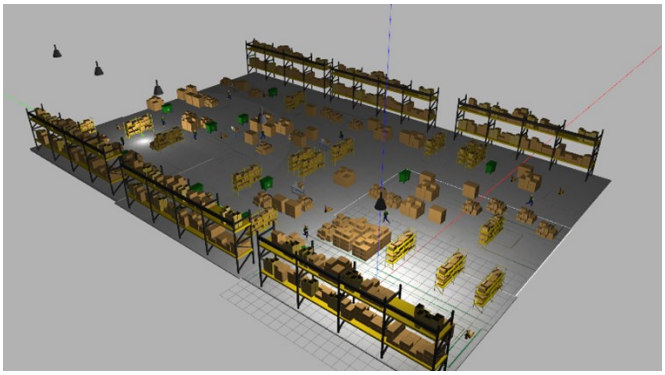

Fig. 2. Dynamic Logistics Warehouse Environment in Gazebo simulator

Fig. 3. Modified Dynamic Logistics Warehouse Environment in Gazebo simulator

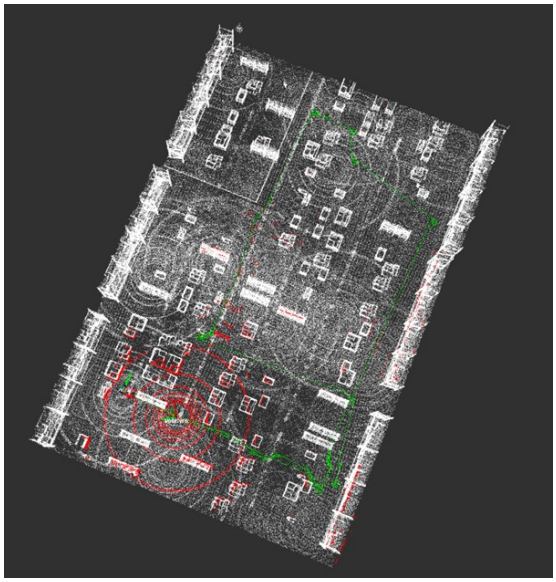

Fig. 4. Point Cloud Map of a Dynamic Logistics Warehouse Environment

B. Robot Setting

We used the Jackal robot for our simulations [8], as shown in Figure 8 The sensors employed for the robot's localization included a Velodyne VLP-16 LiDAR, a wheel encoder, and an IMU sensor.

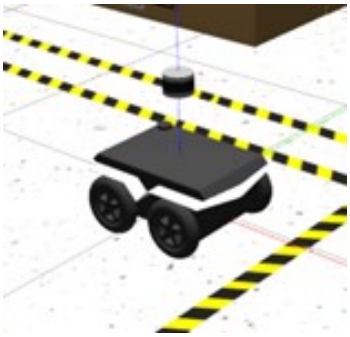

Fig. 5. Dynamic logistics warehouse environment in Gazebo simulator

C. Data Analysis

In our environmental setting, GICP algorithm failed to localize accurately when several structures in the warehouse environment disappeared, whereas the proposed Particle Filter algorithm successfully achieved robust localization despite the environmental changes. This is illustrated in Figure 5, which shows the x and y coordinates of the robot's path in the logistics warehouse from a bird's eye view. For a 188-second simulation of the same robot movement, the localization results of the GICP and Particle Filter algorithms were derived. The blue dotted line represents the localization result using the Particle Filter algorithm, the green dotted line represents the localization result using GICP algorithm, and the red dotted line is the ground truth of the actual robot's path in the Gazebo environment. The performance comparison between GICP and Particle Filter was conducted by plotting the positional error graph, showing the difference between the ground truth and the positions estimated by each algorithm over time. As shown in Figure 6, the positional error of the Particle Filter algorithm over time stabilized at a low value, whereas the positional error of the GICP algorithm diverged. Numerical comparison was made using the Root Mean Square (RMS) error, as shown in Table 1. Consequently, it was confirmed that the Particle Filter algorithm demonstrated superior performance.

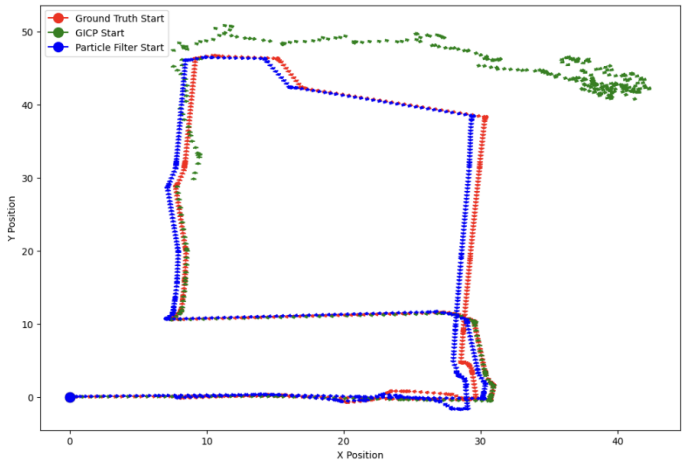

Fig. 6. Bird's Eye View of robot 2D Trajectory

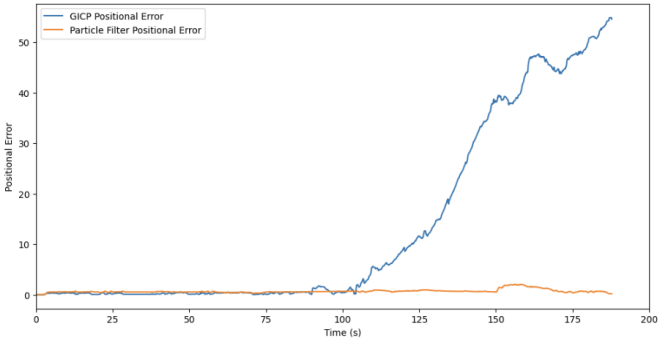

Fig. 7. Positional Error Over Time

	GICP	Particle Filter
RMS error	22.69m	0.79m

TABLE I

IV Conclusion

Our proposed method provided robust and accurate localization in the case of highly deformed environment, while the conventional scan matching showed significant drift and tracking failure. Locations such as warehouses, factories can be deformed since the stocks and massive gadgets are frequently moving. In those cases, our method can provide robust localization even if the pre-built map and the current environment are not exactly correspondence to each other. It might reduce the labor for operator to rebuilt the map in the case of stock replacement, environmental deformation. For further study, collaborating the proposed method and the semantic mapping might be conducted. Since our current method relies equally on both deformed and non-deformed map data, semantic mapping might provide information about whether the specific data on the map is more likely to be static(e.g. wall) or not(e.g. chairs, person).

References

[1] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 4758-4765

[2] K. Koide, M. Yokozuka, S. Oishi and A. Banno, "Voxelized GICP for Fast and Accurate 3D Point Cloud Registration," 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021, pp. 11054-11059

[3] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo localization for mobile robots," Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 1999, pp. 1322-1328 vol.2

[4] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," in IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1-21

[5] R. Nicole, Josh Bongard; Probabilistic Robotics. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. (2005, MIT Press.) 647 pages. Artif Life 2008; pp. 110

[6] B. Ibrahim, Dynamic Logistics Warehouse. Available at: https://github.com/belal-ibrahim/dynamic$_{l}ogistics_{w}arehouse$.

[7] H. Xiao, Y. Han, J. Zhao, J. Cui, L. Xiong and Z. Yu, "LIO-Vehicle: A Tightly-Coupled Vehicle Dynamics Extension of LiDAR Inertial Odometry," in IEEE Robotics and Automation Letters, vol. 7, no. 1, pp. 446-453

[8] T. Shan, Jackal Velodyne. Available at: https://github.com/TixiaoShan/jackal$_{v}elodyne$.