

LiDAR-Enhanced Real-Time Tree Position Mapping for Forestry Robots

M.A Munjer*

Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-City, Fukuoka, 820-8502, Japan

Tan Chi Jie

Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-City, Fukuoka, 820-8502, Japan

Eiji Hayashi

Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-City, Fukuoka, 820-8502, Japan

Email: munjer.abul-md690@mail.kyutech.jp, tan.jie-chi339@mail.kyutech.jp, haya@ics.kyutech.ac.jp

Abstract

This article evaluates the effectiveness of an autonomous robot in creating a tree pose map using both simulated and experimental environments, validating its potential for forestry applications. This work also demonstrates the implementation of FastSLAM on a four-wheeled differential-drive robot, integrating real-time tree detection and tracking through LiDAR-based point cloud data. An algorithm is proposed to generate a map showing both the robot's path and detected tree positions during movement. Performance Metrics Analysis (PMA) revealed a high True Acceptance (TA) rate, confirming accurate tree position estimation. Experimental results validated the algorithm's reliability, showcasing strong distance accuracy with minimal discrepancies between actual and estimated positions. These findings highlight the system's potential for advancing forestry management through precise robotic navigation and mapping.

Keywords: LiDAR , DBSCAN, FastSLAM, Eccentricity

1. Introduction

Recent studies indicate, due to lack of labor force [1], mobile robots have the potential to substitute human labor in various sectors as forestry tasks. Among various Simultaneous Localization and Mapping (SLAM) methods, the particle filter-based FastSLAM approach has been demonstrated to accurately estimate a robot's trajectory while concurrently constructing a map of its surroundings, as described in study [2]. This method is particularly advantageous in scenarios involving dynamic or partially observable environments, where traditional SLAM techniques may struggle with computational efficiency or data association challenges. In the context of forestry robotics, study [3] on Online SLAM for Forestry Robots assumed the central positions of trees to correspond to the centers of clusters derived from LiDAR point-cloud data. However, due to the inherent limitations of LiDAR sensors, it was observed that trees are only partially visible within the limited scope of a single LiDAR scan, often leading to incomplete data capture for individual tree trunks. Additionally, study [4] noted that when robots operate near trees, only a smaller portion of the tree trunks is detected by the LiDAR sensor. This discrepancy causes significant deviations between the calculated cluster centers and the actual tree trunk centers, impacting the accuracy of SLAM and overall mapping fidelity. To address these challenges, it has been identified that a more precise and reliable method for estimating tree positions is essential for the successful application of FastSLAM in autonomous forestry robots. This research suggests improving tree position estimation by detecting

tree trunks up to a specific height after separating other non-tree objects from the 3D point-cloud data generated by the LiDAR sensor. Such segmentation not only enhances the accuracy of the detected tree locations but also mitigates the effects of occlusions and noise in complex forestry environments. Furthermore, this study introduces a robust and real-time methodology for tree detection and tracking, leveraging point-cloud data from LiDAR sensors. The proposed approach integrates the capabilities of the Robot Operating System (ROS2) [5] for system management, Open3D for efficient point-cloud processing [6], and advanced filtering and clustering techniques to ensure the accurate identification of trees in diverse scenarios. By focusing on real-time processing, the method is designed for static forest environments, where tree positions remain relatively stable, allowing accurate mapping and tree identification based on point cloud data. This research not only contributes to improved tree mapping but also lays the groundwork for enhancing the navigation and task-planning capabilities of autonomous field robots in forestry applications.

2. System overview

The proposed forestry mapping system integrates multiple sensors and computational components to enable autonomous navigation and real-time mapping. Fig. 1 represents the system overview of this research where it relies on a differential-drive robot equipped with a Velodyne LiDAR sensor, an Intel RealSense Depth Camera, and an XSense IMU sensor. The Velodyne LiDAR captures high-density 3D point cloud data, crucial for detecting tree clusters and estimating their

positions. Complementing this, the Intel RealSense camera provides RGB and depth data to enhance environmental perception, while the IMU sensor supplies orientation, angular velocity, and acceleration data to support accurate localization and robot stability. All sensor data is processed by a ROS2-based computational unit, which serves as the control hub for real-time operations.

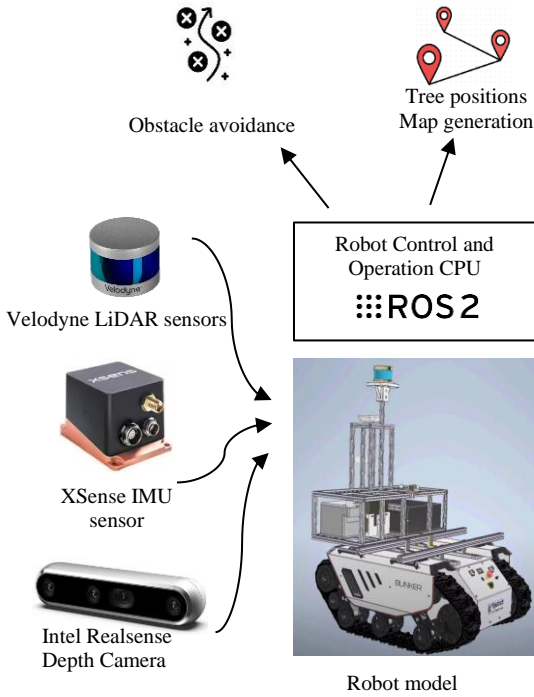


Fig. 1. Robot architecture and system overview for forestry mapping.

The system leverages LiDAR and camera inputs for tree detection and tracking while utilizing IMU data to stabilize navigation in uneven terrains. Key functionalities include obstacle avoidance, which ensures safe traversal through the forest, and tree position mapping, where the detected tree locations are transformed into a global frame for map generation. This architecture is designed for scalability and robustness, allowing the robot to operate in challenging forestry environments. The generated maps, displaying both the robot's path and tree locations, provide valuable insights into the environment, contributing to efficient forestry management and autonomous operations.

3. Methodology

3.1. Process Flow for Real-Time Tree Mapping

The system follows a structured pipeline for tree detection and mapping which have been shown in fig. 2. It begins with LiDAR and camera data collection, followed by preprocessing steps like noise filtering and down-sampling. Clustering is performed using DBSCAN to identify initial tree groups. These clusters are refined by calculating eccentricity through PCA. Tree positions

are then mapped to a global frame, with Kalman Filter smoothing applied for accuracy.

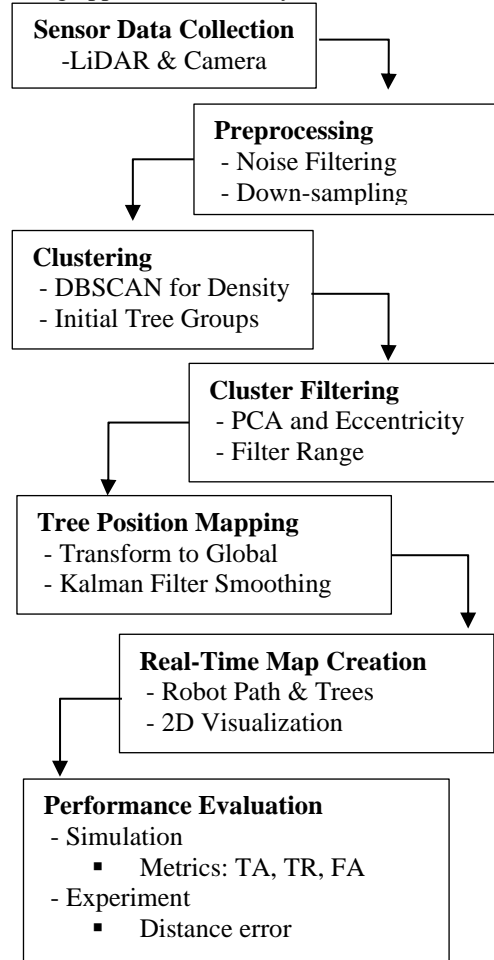


Fig. 2. Process flow diagram of the tree pose mapping.

The final output is a real-time 2D map combining the robot's path and tree locations. The system's performance is evaluated using metrics like TA, TR, FA, and FR under the Performance Metrics Analysis (PMA) framework. The details of the methodology can be described in the next sections.

3.2. Point Cloud Acquisition and Preprocessing

The initial stage of the methodology involves acquiring a 3D point cloud using a Velodyne LiDAR sensor mounted on mobile robot. This sensor generates a high-density array of points in 3D space by emitting laser pulses and capturing their reflections off surrounding objects [7], resulting in a set P of individual points. This point cloud data can be expressed as in Eq. (1)

$$P = \{p_i \mid p_i = (x_i, y_i, z_i) \in \mathbb{R}^3, i=1\dots,N\} \quad (1)$$

where each point p_i is represented by coordinates (x_i, y_i, z_i) , with N denoting the total number of points. However, raw point clouds typically contain noise due to sensor inaccuracies or interference from environmental elements like foliage or small debris. To address this, a

series of preprocessing steps have been applied to remove irrelevant data and reduce computational complexity. The first step in preprocessing is applying a pass-through filter along each axis, which removes points outside a specified spatial range. For instance, to filter along the x-axis, points are retained only $p_i \in P$ if $0 < x_i < \beta_x$, $-\beta_y < y_i < \beta_y$, $-\beta_z < z_i < \beta_z$ where β_x , β_y and β_z are the distance threshold along the x, y and z-axis respectively in the LiDAR's frame. By selectively filtering points, the algorithm reduces data size and removes noise outside our region of interest. Following this, a voxel grid filter is applied to down-sample the point cloud further. The voxel grid divides the 3D space into small cubes (voxels) of size δ , averaging the points within each voxel into a single representative point. This creates a reduced set of down-sampled points, P' , which can be mathematically represented as in Eq. (2)

$$P' = \left\{ \frac{1}{|V|} \sum_{p_i \in V} p_i \mid V \subset P, |V| > 0 \right\} \quad (2)$$

where V is a voxel containing multiple points, and $|V|$ is the number of points within that voxel. This step not only reduces data volume but also preserves key structural features, allowing for efficient processing in later stages.

3.3. Tree Detection and Clustering

With a cleaned and down-sampled point cloud, the next stage involves identifying clusters within the data that likely represent tree trunks. To achieve this, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm has been applied [8], which is effective for clustering data with arbitrary shapes and noise. DBSCAN identifies clusters based on two main parameters: ϵ , the maximum distance between two points to consider them as neighbors, and minPts , the minimum number of points required to form a cluster. The clustering can be defined mathematically as $\text{DBSCAN}(P', \epsilon, \text{minPts}) \rightarrow C = \{C_1, C_2, \dots, C_k\}$ where C_k represents a cluster of points satisfying DBSCAN's density-based criteria. These clusters are candidates for tree trunks, but additional filtering is required to confirm their shape. To illustrate the application of the mathematical principles of voxelization and DBSCAN clustering, Fig. 3 demonstrates the sequential processing of a noisy point cloud captured by a LiDAR sensor. As shown in Fig. 3(a), the raw point cloud data contains noise and extraneous points due to sensor inaccuracies and environmental interference. To reduce the computational complexity, the point cloud is voxelized, as shown in Fig. 3(b). The voxelization process divides the point cloud into uniform 3D grid cells, retaining a representative point for each occupied cell. This step significantly reduces the number of points while preserving the overall structure of the data. After voxelization, DBSCAN clustering is applied to identify dense regions and separate noise points.

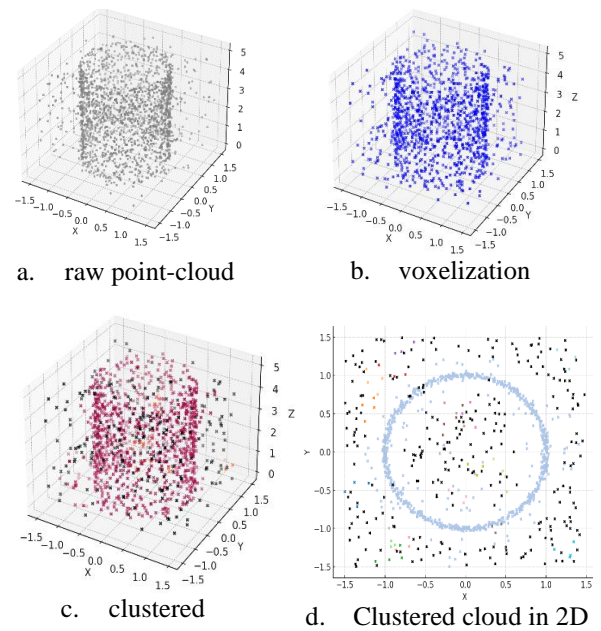


Fig. 3. Point-cloud processing for the cauterization.

Fig. 3(c) shows the output of DBSCAN, where each cluster is marked with a distinct color, representing isolated groups such as tree trunks. Noise points, which fail to meet the density criteria, are excluded. To simplify interpretation, the clustered point cloud is projected onto a 2D plane, as depicted in Fig. 3(d). This 2D visualization provides a clear representation of the spatial distribution of clusters, such as tree trunks, and aids in analyzing their positions relative to the environment. Once the clusters are identified, their shapes are analyzed to distinguish tree trunks from other objects. Eccentricity, a measure of how much a shape deviates from being circular, is calculated for each cluster. To assess the shape of each cluster and distinguish trees from other objects, eccentricity is being calculated using Principal Component Analysis (PCA) [9]. For each cluster C_k , the covariance matrix Σ has been computed, which quantifies how points in the cluster vary along each axis.

The covariance matrix is represented by Eq. (3)

$$\Sigma = \frac{1}{|C_k|} \sum_{p_i \in C_k} (p_i - \bar{p})(p_i - \bar{p})^T \quad (3)$$

where \bar{p} represents the centroid of the cluster. An eigenvalue decomposition of Σ yields two principal eigenvalues, λ_1 and λ_2 , with $\lambda_1 > \lambda_2$. Eccentricity e can be calculated as in equation Eq. (4).

$$e = \frac{\lambda_1}{\lambda_2} \quad (4)$$

This ratio indicates how elongated or cylindrical a cluster is; higher values suggest a more tree-like, hyperbolic-cylindrical shape, while lower values imply a more rounded or irregular shape. Clusters with an eccentricity

above a specific threshold are classified as trees, while others are discarded. In this research, it was observed that eccentricity values between 10 and 20 effectively classify tree-like hyperbolic clusters. This threshold aligns with the inherent geometry of partial tree stems, distinguishing them from noise or non-tree objects, which either exhibit lower eccentricity (more circular shapes) or significantly higher eccentricity (extremely elongated shapes). By exploiting the hyperbolic geometry of tree clusters, the proposed method ensures tree like shape detection.

3.4. SLAM and Trajectory Estimation

To localize the robot while mapping tree positions, FastSLAM has been implemented, an algorithm that combines particle filtering with landmark-based mapping[10]. FastSLAM maintains multiple particles, each representing a hypothesis of the robot's state and map. Each particle, denoted $S_t^{(k)}$, consists of a state hypothesis (the robot's estimated position and orientation) and a map hypothesis (detected tree positions), formulated as Eq. (5)

$$S_t^{(k)} = \{state_t^{(k)}, map_t^{(k)}\} \quad (5)$$

where k indexes the particle. The robot's state x_t at time t is updated based on control inputs (e.g., wheel velocities) and a process noise term ω_t , using the motion model can be express as in Eq. (6).

$$x_t = f(x_{t-1}, u_t) + \omega_t \quad (6)$$

Here, $f(x,u)$ is a function describing how control inputs u_t affect the robot's position, and ω_t accounts for uncertainties in movement. Upon detecting a tree, each particle's map hypothesis is updated based on the observed tree position, with an observation model given by Eq. (7)

$$z_t = h(x_t, t_i) + v_t \quad (7)$$

where $h(x_t, t_i)$ is a function mapping the robot's state and the tree's local position to the predicted measurement, and v_t is measurement noise. Each particle is weighted based on how well its predicted map aligns with observed data, favoring particles that are consistent with detected trees.

3.5. Transformation of Tree Coordinates to Global Frame

Since detected tree positions are initially represented in the LiDAR's local frame, we need to transform them to the robot's global frame for consistency. Each detected tree position is part of a set $T_{velodyne}$ that can be represented as in Eq. (8).

$$T_{velodyne} = \{t_i \mid t_i = (x_i, y_i, z_i) \in \mathbb{R}^3, i=1\dots M\} \quad (8)$$

To represent these coordinates in the global odometry frame, we apply a series of transformations. The global transformation matrix T_{global_odom} is computed as following Eq. (9).

$$T_{global_odom} = T_{velodyne} \cdot T_{base_link} \cdot T_{odom} \quad (9)$$

where $T_{velodyne}$ represents the LiDAR data in its own frame, T_{base_link} maps the LiDAR frame to the robot's base, and T_{odom} maps the base frame to the global odometry frame. Each tree coordinate $t_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ is transformed to $t'_i = (x'_i, y'_i, z'_i)$ in the global frame, ensuring alignment with the robot's path and other map elements. By applying these transformations, each tree's position is now fixed on the global map. However, these global coordinates may still contain slight inaccuracies due to sensor noise or minor fluctuations in the robot's position.

3.6. Kalman Filter for Tree Position Estimation

Once the tree coordinates are transformed to the global frame, the Kalman Filter refines these positions by filtering out noise and providing a more stable estimate over time [11]. This step is essential because even with the global transformation, the tree positions can vary slightly due to measurement noise or sensor drift. The Kalman Filter treats each tree's global position as a state that is updated continuously, using a combination of prediction and measurement updates. In the Kalman Filter, the state transition matrix predicts each tree's future position based on the previous estimate, in each dt time step. This matrix relates the robot's previous state to its current state. Using the Kalman Gain the predicted state is adjusted according to measurement z , providing a more stable and accurate tree position estimate. The Kalman Gain K is a dynamic matrix that determines how much weight the filter should place on the new measurement relative to the prediction where K is calculated as in Eq. (10).

$$K = PH^T(HPH^T + R)^{-1} \quad (10)$$

The measurement matrix H maps the predicted state to the actual measurement space, enabling the filter to compare the predicted position with the observed position and covariance matrix P in a Kalman Filter represents the uncertainty in the state estimate and is typically initialized based on the expected initial uncertainty. Since our measurements only provide the position (not the velocity), H extracts the positional elements from the state whereas measurement noise covariance matrix R represents the uncertainty in the measurement process.

3.7. Real-Time Map Generation

With the tree positions refined in the global frame, the final step is to integrate these positions into a real-time map that also includes the robot's trajectory. The robot's path R_t , which is a sequence of positions over time based on odometry data, and the refined global tree positions T_{global} are now combined to create a cohesive, accurate map both of them can be represented as in Eq. (11) and Eq. (12).

$$R_t = \{r_j | r_j = (x_j, y_j) \in R^2, j = 1, \dots, T\} \quad (11)$$

$$T_{global} = \{t'_i | t'_i = (x'_i, y'_i) \in R^2, i = 1, \dots, M\} \quad (12)$$

The combination of these elements results in a dynamic, real-time 2D map. This map continuously updates as the robot moves, reflecting both the path the robot has traveled, and the positions of trees detected in the environment.

4. Simulation and Experimental Results

The proposed algorithm was tested in a simulated environment using Gazebo with real-world scenarios. Key performance metrics, such as True Accepts (TA), True Rejects (TR), False Accepts (FA), and False Rejects (FR), were calculated to evaluate the system's effectiveness according to the analysis in study [12]. In this case TA and TR signifies number of trees correctly identified and amount of non-trees object correctly identified respectively. On the other hand, FA and FR signify the amount of non-tree objects mistakenly identified as trees and trees that were missed are estimated.

Table 1. Tuned Parameters and Threshold Values.

Parameter	Value(s)
number of particles for fastSLAM (unit)	100
eccentricity_threshold (unitless)	20
voxel_size (meter)	0.01
eps (DBSCAN) (meter)	0.25
min_points (DBSCAN) (number of points)	20
proximity_radius (meter)	1.0
cylindrical_stddev_thresh (meter)	0.1

Table 1 outlines various tuned parameters used in the algorithm, including a voxel size of 0.01m for down-sampling the point cloud and an eps value of 0.25m for DBSCAN clustering to define the neighborhood distance where voxel size of 0.01 means each voxel in the grid has a dimension of 0.01 units, reducing the point cloud density. Additionally, the pass-through filters have specific ranges for x (0 to 10), y (-10 to 10), and z (-0.5 to 0) axes, ensuring relevant points are retained and an eccentricity threshold of 20 are also specified to optimize the detection and tracking process.

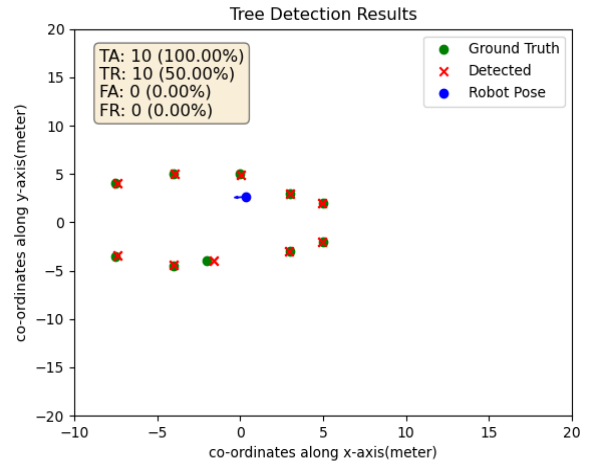


Fig. 4. Simulation result of tree detection accuracy.

The X and Y axes are represented in Fig. 4, the global coordinate system used to plot the positions of the robot and the trees, indicating the area covered by the robot's exploration. The results in Fig. 2 indicate that the system has a high rate of correctly identifying trees (TA: 100%) and non-tree objects (TR: 50%). However, the False Accepts rate (FA: 0%) is relatively low, suggesting that the system didn't rarely mistake non-tree objects for trees. The False Rejects rate (FR: 0%) indicates that no actual trees were missed, indicating good sensitivity. The percentage rate of TR could be due to the presence of objects in the environment that have similar shapes or features to trees, such as human or other vertical structures. On the other hand, the rate of TR as such might be attributed to occlusion, where trees are blocked by other trees or objects, or due to the limitations of the clustering and filtering algorithms used.

Fig. 5 represents the positions and trajectories of the robot navigating through a forest-like environment in Kyutech, Iizuka campus playground area. The red triangle marks the current position and orientation of the robot detected, dynamically tracked as it navigates the environment. The grey line traces the path that the robot has traveled, providing a visual representation of the robot's movement over time. Different colored circles represent confirmed objects in the environment, identified as trees. Table 2

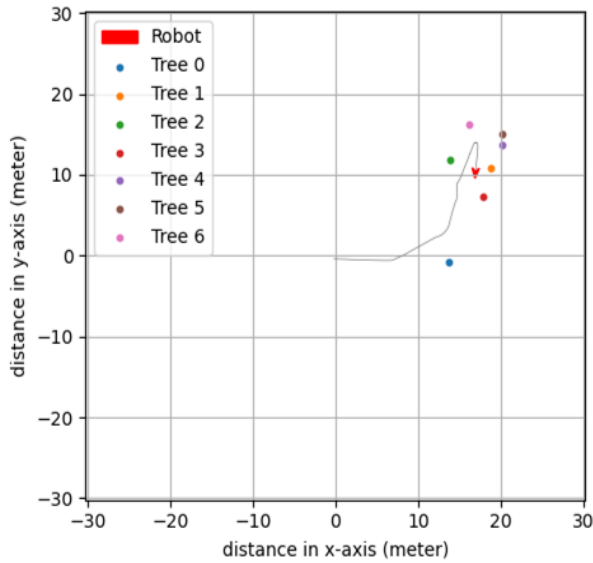


Fig. 5. Real time estimated map through experiment.

represents the coordinates and respective eccentricity of detected trees by the robot. Each tree is defined by its X and Y coordinates, indicating its location in the 2D mapping space, and an eccentricity value that characterizes the shape of the detected cluster associated with each tree.

Table 2. Tree position and their eccentricity values.

Tree Number	X	Y	Eccentricity
0	13.599	-0.782	11.539
1	18.74	10.778	14.58
2	13.91	11.912	10.003
3	17.897	7.7.342	16.457
4	20.201	13.664	11.066
5	20.177	14.983	10.765
6	16.02	16.214	12.993

The x and y coordinates reveal a varied spatial distribution, with positions ranging from approximately (13.599, -0.804) for "Tree 0" to (20.177, 14.963) for "Tree 5," suggesting a spread across a specific area. The eccentricity values, derived from Principal Component Analysis (PCA), quantify the elongation of each tree cluster, where higher values represent more elongated, cylindrical shapes indicative of tree trunks. For instance, "Tree 3" has an eccentricity of 16.457, reflecting a highly elongated shape, while "Tree 2" has a lower eccentricity of 10.003, indicating a less pronounced elongation. This data is critical for distinguishing tree-like structures from other objects in the environment, supporting accurate mapping and analysis in forestry applications by identifying clusters with shapes typical of tree trunks. From Fig. 6, it can be observed a visual representation of the spatial distribution and distances between selected trees within the experimental environment. Each tree is labeled from Tree 0 to Tree 6, with unique colors for easy

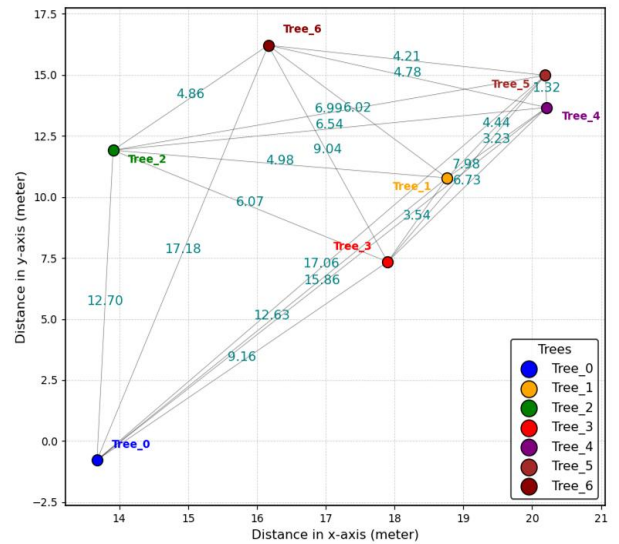


Fig. 6. Distance plot of detected tree pairs.

identification. The graph illustrates both the x-axis and y-axis distances, with each inter-tree distance labeled, providing insight into the relative proximity of each tree. The distances between pairs are annotated, showing varying values based on spatial positioning. The lines connecting the trees indicate the paths used for calculating the distances, which are essential for mapping and navigation purposes. This visualization highlights the relative proximity of trees in the environment and provides a quantitative way to validate the accuracy of tree position detection. By comparing these distances, we can assess whether the detected tree positions align with the expected real-world configuration, which is crucial for verifying the algorithm's performance in tree detection and mapping. In Fig. 7, the matrix plot illustrates the percentage errors in distance measurements between various tree pairs. The average error of 7.13% highlights the effectiveness of the tree detection and mapping algorithm where most tree pairs exhibit minimal

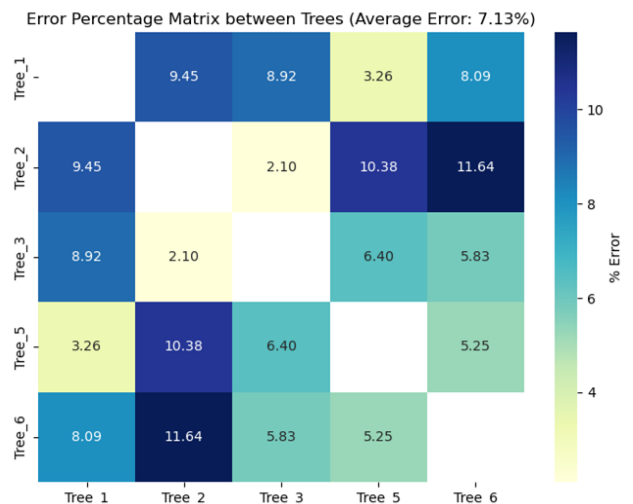


Fig. 7. Matrix plot of errors between measured and estimated distance.

error, indicating high accuracy and consistent detection. revealing higher errors in pairs such as "Tree 2 to Tree 6" (11.64%) and "Tree 2 to Tree 5" (10.38%). Several factors likely contribute to these discrepancies in a real-time forestry mapping context. First, sensor noise and interference from environmental elements like dense foliage and branches can scatter or absorb LiDAR beams, leading to inaccuracies, particularly over greater distances or when foliage obstructs the line of sight.

Additionally, tree occlusion and overlapping canopies pose challenges; trees may partially block each other from the sensor's view, which can distort perceived distances and increase errors for certain pairs. Real-time constraints add to this challenge, as rapid processing is required to maintain continuous updates, often involving approximations that prioritize speed over precision. Furthermore, terrain variability and sensor angles can introduce alignment issues; as the robot moves over uneven ground, slight tilts can misalign point cloud data, affecting distance estimates. In a forestry environment, these levels of error are generally acceptable given the inherent complexity and natural variability. For this navigation task, low rate of error suggest that the robot can effectively perceive its environment and navigate safely around obstacles without significant impact on overall performance. Overall, for real-time forestry mapping, the observed errors may be tolerable for general navigation. If higher precision is needed, though, further refinement in the algorithms or sensor calibration would be advisable to reduce the impact of environmental factors and improve accuracy.

5. Conclusion

The system shows strong performance in correct detections; however, challenges remain, such as distinguishing tree trunks from other cylindrical objects like poles or humans, which can lead to false positives. Environmental factors, including low-hanging branches or overlapping canopies, also introduce noise and hinder accurate detection. Despite these challenges, the low average error for experimental output demonstrates the algorithm's robustness and reliability in estimating tree positions. Future work could focus on integrating camera-based tree detection to complement LiDAR data, enabling more precise tree identification and classification. This multimodal approach could address current limitations, such as false positives from cylindrical objects, by leveraging advanced image-based classification techniques. Additionally, improvements in filtering techniques, clustering algorithms, and adaptive thresholding could enhance performance in complex forest environments. The generated maps remain a valuable tool for visualizing the robot's navigation and tree pose estimation capabilities, contributing significantly to the development of autonomous robots for forestry applications.

References

1. C. J. Tan, S. Ogawa, T. Hayashi, T. Janthori, A. Tominaga, and E. Hayashi, "3D Semantic Mapping based on RGB-D Camera and LiDAR Sensor in Beach Environment," *2024 1st International Conference on Robotics, Engineering, Science, and Technology, RESTCON 2024*, pp. 21–26, 2024.
2. Sebastian Thrun, Michael Montemerlo, Daphne Koller, Ben Wegbreit, Juan Nieto, and Eduardo Nebot, "FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data," *Journal of Machine Learning Research*, pp. 1–48, 2004.
3. Sylvain Geiser, Sakmongkon Chumkamon, Ayumu Tominaga, Takumi Tomokawa, Eiji Hayashi, Online SLAM for Forestry Robot, *Journal of Robotics, Networking and Artificial Life*, vol-9-2, pp-177-182, 2022-2023.
4. Geiser Sylvain, Chumkamon Sakmongkon, Tominaga Ayumu, Tomokawa Takumi, Jie Tan Chi, and Hayashi Eiji, "Practical Implementation of FastSLAM for Forestry Robot, Proceedings of International Conference on Artificial Life and Robotics," Feb. 2023, pp. 318–322.
5. S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Sci Robot*, vol. 7, no. 66, May 2022.
6. Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847*, pp. 1–6, Jan. 2018.
7. Tan Chi Jie, "The BCRobo dataset for Robotic Vision and Autonomous Path Planning in Outdoor Beach Environment," *ICAROB2023*, pp. 327–332, Feb. 2022.
8. D. Deng, "DBSCAN Clustering Algorithm Based on Density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEAA)*, Hefei, China, 2020, pp. 949–953.
9. A. Maćkiewicz and W. Ratajczak, "Principal components analysis (PCA)," *Comput Geosci*, vol. 19, no. 3, pp. 303–342, Mar. 1993.
10. M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001.
11. R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960.
12. Alireza Baratloo, Mostafa Hosseini, Ahmed Negida, and Gehad El Ashal, "Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity," *Emerg (Tehran)*, vol. 3(2), p. 48, 2015.

Authors Introduction

M.A Munjer



He received his Masters of Science in Engineering from the Department of Electrical and Electronic Engineering, RUET, Bangladesh in 2019. He is currently a Doctoral student at Kyushu Institute of Technology and conducts research at Hayashi Laboratory.

Tan Chi Jie



He received his Masters of Science in Engineering from the Department of Computer Science and Systems Engineering, Kyushu Institute of Technology, Japan in 2023. He is currently a Doctoral student at Kyushu Institute of Technology and conducts research at Hayashi Laboratory.

Prof. Eiji Hayashi



Prof. Eiji Hayashi is a professor in the Department of Intelligent and Control Systems at Kyushu Institute of Technology. He received the Ph.D. (Dr. Eng.) degree from Waseda University in 1996. His research interests include Intelligent mechanics, Mechanical systems and Perceptual information processing. He is a member of The Institute of Electrical and Electronics Engineers (IEEE) and The Japan Society of Mechanical Engineers (JSME).