Received 11 March 2024, accepted 20 March 2024, date of publication 25 March 2024, date of current version 1 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3381489

RESEARCH ARTICLE

Development of an Indoor Delivery Mobile Robot for a Multi-Floor Environment

TAEJIN KIM[1], **(Student Member, IEEE), GYUREE KANG**[2], **(Student Member, IEEE), DAEGYU LEE**[2], **(Student Member, IEEE), AND D. HYUNCHUL SHIM**[2], **(Member, IEEE)**

[1]Robotics Program, Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon 34141, Republic of Korea
[2]Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon 34141, Republic of Korea

Corresponding author: Taejin Kim (kimtj5521@kaist.ac.kr)

This work was supported by the Technology Innovation Program (Development of drone-robot cooperative multimodal delivery technology for cargo with a maximum weight of 40kg in urban areas) funded by the Ministry of Trade, Industry & Energy (MOTIE), South Korea, under Grant RS-2023-00256794.

ABSTRACT In recent years, the demand for delivery services has increased by applying robot technology in various fields such as food services, logistics, hospitals, and hotel business. However, it is still challenging to perform autonomous delivery in multi-floor buildings. Particularly for wheeled robots, the use of elevators is essential for indoor last-mile delivery service in buildings. To tackle the problem, We have developed an indoor delivery mobile robot and present its architecture designed for multi-floor environments. The architecture consists of five modules: map management for utilizing an integrated navigation map, localization, path planning, perception, and task planning. The integrated navigation map is generated by combining multi-floor point cloud maps and topological maps based on node graphs for effective localization and path planning. Additionally, the proposed 3D route planning allows inter-floor movement. A feasible path for boarding the elevator can be generated through the perception module, and delivery services to multiple destinations can be repeatedly performed through task planning. Our architecture's effectiveness is demonstrated through a month-long field test in an ordinary building during regular business hours. This study's contributions include a novel architecture for autonomous delivery without human intervention, an integrated map for efficient indoor navigation, and the proven robustness of the system in real-world scenarios.

INDEX TERMS Delivery service, mobile robot, multi-floor, route planning, elevator boarding, docking.

I. INTRODUCTION

In recent years, industrial demand for contactless services and automated systems has increased significantly due to the COVID-19 pandemic [1]. Concurrently, research is being conducted to apply and expand robot technology in various fields, such as food services, logistics, and hotel business, to replace a decreasing labor population and increase work efficiency [2], [3], [4], [5], [6]. In particular, mobile robots are considered an effective solution in the last mile delivery service field [7], [8], with companies like Amazon, Starship, and FedEx actively involved in research and development [9], [10]. The robots researched in this way perform tasks by moving indoor and outdoor environments for delivery services [11], [12]. Mobile robots significantly enhance efficiency in last-mile delivery by automating the delivery process, thus reducing the time required to transport packages to their final destination. Additionally, they offer substantial cost savings by minimizing the need for human labor, leading to a reduction in operational expenses associated with deliveries.

Such that we specifically focus on the indoor multi-floor environment. There are several considerations when providing delivery services using mobile robots in an indoor environment. First, indoor environments are often GPS-denied, meaning there are restrictions on receiving position information directly [13]. To address this challenge, localization systems that fuse data from sensors such as cameras, inertial measurement units (IMUs), and light detection and ranging (LiDAR) sensors are widely used.

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad AlShabi.

Examples include visual inertial odometry and LiDAR inertial odometry [14], [15]. Moreover, indoor spaces are typically narrower than outdoor spaces, and there are various obstacles, including people, which can make it difficult for robots to navigate. Therefore, obstacle avoidance methods are needed in narrow spaces. Efficient route planning is also essential to maximize the efficiency of last-mile services. In particular, indoor environments are often multi-floor, so it is necessary to develop algorithms that can find efficient routes even when the starting point and destination are on different floors [16]. Because of this, localization should be performed on separate floors. It is important to note that, unlike quadrupedal robots, wheel-based mobile robots ought to use elevators. Therefore, the robots should be able to call, board, and exit elevators automatically. In addition, since it is possible to board an elevator together with people, the robots must be able to board and exit the elevator safely [17]. From a mobile robot perspective, the challenging technical difficulties of last-mile delivery service can be summarized as follows: the need to board and exit the elevator autonomously, the requirement to continue autonomous navigation in a different environment after changing floors using the elevator, and the complexity of the parcel delivery process which involves multiple tasks that a single robot must be capable of performing.

In this study, as shown in Fig. 1, we have developed an indoor delivery mobile robot and present its architecture designed for multi-floor environments to meet these requirements. The proposed architecture consists of five modules: map management, localization, path planning, perception, and task planning. We perform drivable area extraction and elevator detection in the perception module and use the task planning module to enable the mobile robot to board and exit the elevator autonomously. In addition, the concept of an integrated navigation map is proposed so that autonomous driving can be maintained even after the floor is changed, thereby stabilizing the robot's position and path-following state. Furthermore, through task planning, we efficiently transition between the necessary tasks for parcel delivery, such as driving, elevator boarding, delivery, and docking.

The integration of these modules allows mobile robots to perform last-mile delivery services in indoor environments. The main contributions of this research are as follows:

- We propose an architecture for autonomous delivery from parcel receipt to delivery without human intervention. All processes are performed within an ordinary building not specifically designed for robot operation.
- We present an integrated navigation map for robot localization and efficient 3D route planning in indoor multi-floor environments, which includes node graph information containing the meaning of the drivable area.
- We demonstrated the robustness of the proposed architecture through a month-long field test conducted during business hours when people are moving around.

The rest of this paper is organized as follows. Section II introduces previous related works. Section III provides an overview of the proposed system. Section IV explains integrated navigation map generation, path planning, perception, and task planning. Section V presents the experimental results. Finally, in Section VI, we conclude this research.

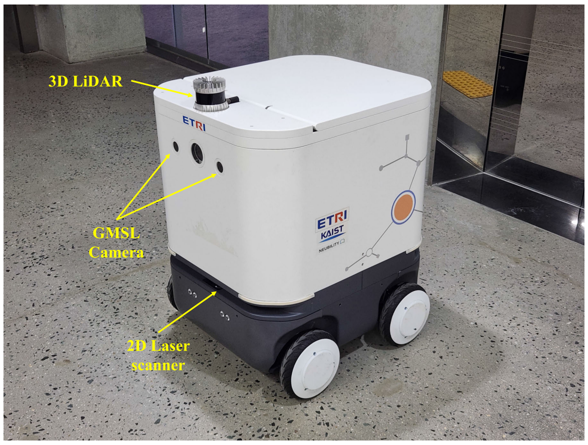


FIGURE 1. The appearance of the mobile robot used in this study.

II. RELATED WORKS

Mapping of indoor environments for delivery service robots is important for localization. Therefore, many studies have been conducted on how to construct a multi-floor map. One of the widely used methods is to use a barometer, which can calculate the altitude value using air pressure [18], [19]. This method generates a multi-floor map by stacking 2D maps vertically using the height values obtained at the time each floor's 2D map was created. However, generating a multi-floor map by aligning and stacking 3D maps can yield a map with much more geometrical information.

Various methods for localization in indoor environments have been researched. One approach involves using landmarks such as StarGazers attached to the ceiling of the building [20]. This method is effective when the robot only operates within a limited area. Since delivery robots may have destinations throughout a building, attaching landmarks to every floor's ceiling is inefficient. Using particle filters to estimate a robot's position has also been studied [21], [22]. However, increasing the number of particles to improve accuracy always tends to higher computational costs. Research has also been conducted on using WiFi signals for localization [23], [24]. However, WiFi signals are susceptible to interference and signal strength variability. To address this, methods using sensor fusion, such as cameras, LiDAR, and WiFi footprints, for localization have been conducted [25], [26].

Research on path planning for delivery service robots in buildings has been conducted from various perspectives. Mantha et al. [27] proposed a method for optimally visiting multiple delivery destinations from the perspective of the traveling salesman problem. Kim and Jung [16] proposed a method to optimize the delivery time using a genetic algorithm. Palacín et al. [28] proposed a method for

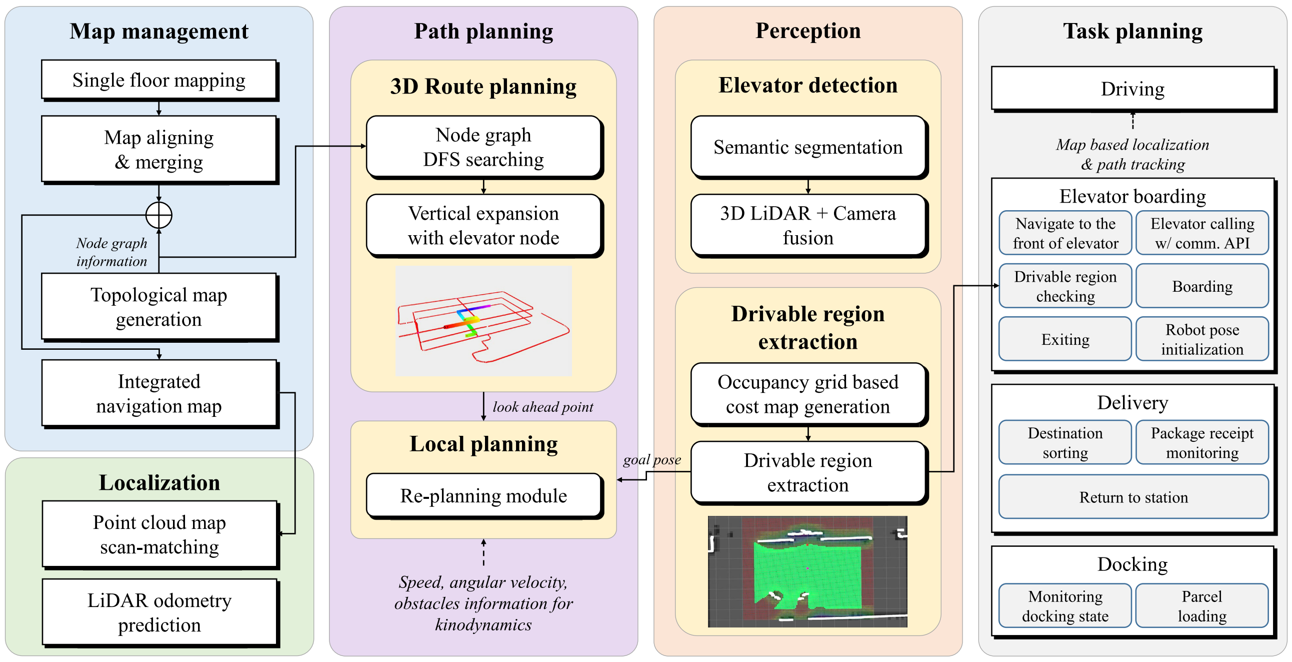

Map management
Single floor mapping
Map aligning & merging
Node graph information
Topological map generation
Integrated navigation map
Localization
Point cloud map scan-matching
LiDAR odometry prediction
Path planning
3D Route planning
Node graph DFS searching
Vertical expansion with elevator node
look ahead point
Local planning
Re-planning module
goal pose
Speed, angular velocity, obstacles information for kinodynamics
Perception
Elevator detection
Semantic segmentation
3D LiDAR + Camera fusion
Drivable region extraction
Occupancy grid based cost map generation
Drivable region extraction
Task planning
Driving
Map based localization & path tracking
Elevator boarding
Navigate to the front of elevator
Elevator calling w/ comm. API
Drivable region checking
Boarding
Exiting
Robot pose initialization
Delivery
Destination sorting
Package receipt monitoring
Return to station
Docking
Monitoring docking state
Parcel loading

FIGURE 2. Overview of the developed system architecture.

generating paths by using a predefined navigation tree in a 2D map, estimating the length of the path using Dijkstra's algorithm. However, these methods addressed route planning problems on a single floor and the paths on other floors are calculated separately.

Various methods for inter-floor movement have also been researched. Approaches where robots recognize people in elevators, follow them on board, and use barometers to detect floor changes to reach the desired floor have been studied [22], [29]. Research has also been conducted on recognizing buttons and numbers through vision [17], [30]. However, these methods rely on the precondition that the robot's view is not obstructed by passengers.

Existing studies have focused on single tasks such as generating multi-floor maps, path planning, or elevator boarding. However, for indoor delivery service, these tasks need to be connected so that a robot can perform them all, enabling effective last-mile delivery services.

III. SYSTEM OVERVIEW

Fig. 2 illustrates the overview of our developed system architecture. The system is divided into five parts including map management, localization, path planning, perception, and task planning.

The map management module initially creates point cloud maps for each floor of the building, then aligns the maps of different floors based on shared areas. After creating the map of each floor, nodes that can act as anchors are designated and added to the single-floor map. These nodes are used as references for the drivable region on that floor. They are composed of points corresponding to the end of the corridor, intersections where multiple directions are possible, points on elevators, and delivery destinations. The nodes also have attributes according to their roles. For example, intersection nodes include additional data like the number of branches. The map that includes these nodes is defined as an integrated navigation map.

The localization is performed by fusing the LiDAR odometry method and registration. The registration between current LiDAR data and the point cloud map is conducted simultaneously with the estimation of LiDAR odometry in parallel. the LiDAR odometry is utilized as the predictive transformation for the next step of the registration.

For path planning, the robot needs to move through multiple floors. Therefore, if the destination is on a different floor than the current floor, a 3D route is generated by connecting the elevator nodes on the current and destination floor. When the integrated navigation map of the current floor is loaded, the nodes are interpolated to connect them to a graph. This graph is then utilized as the route.

To detect elevators for inter-floor movement, both a camera and LiDAR sensor are used. First, we calibrate the camera and LiDAR sensor, then use the YOLACT [31] method to pixel-wise segment the elevator in the camera image. After that, we cluster the point cloud corresponding to the pixels segmented as the elevator and use the center point as the elevator position. When boarding the elevator, not only the position of the elevator is used, but also a drivable region extracted from the LiDAR data is utilized to decide whether to board the elevator or not. The drivable region obtained in this process is also applied to local path planning for effective obstacle avoidance.

Task planning is performed by dividing the entire delivery service into several tasks and distinguishing the robot's

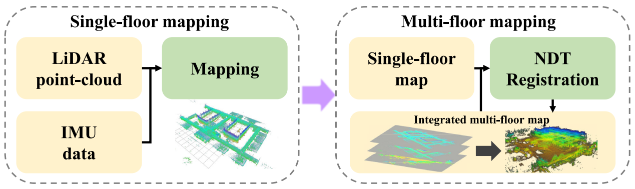


FIGURE 3. Pipeline of multi-floor map generation. A multi-floor map is generated by matching and aligning the same areas among each floor's 3D point cloud maps and then stacking them up.

behavior. Each task consists of a driving task that moves along the route, an elevator boarding/exiting task, a delivery task that is performed upon reaching the delivery destination, and a docking task for receiving parcels or returning to the station after delivery.

An Ouster OS1 128 channel 3D LiDAR sensor, two A2 RPLiDAR sensors, four Seconix SF3325-100 (RCCB) GMSL cameras, and MicroStrain 3DM-GX5-25 IMU sensors were used with an Intel NUC (Intel i7 processor, 64GB RAM) and an Nvidia Jetson Xavier AGX as the computing devices. All systems were implemented in C++ and Python. Also, it was executed using the Robot Operating System (ROS) [32] in Ubuntu 18.04 Linux.

IV. METHODOLOGY

A. INTEGRATED NAVIGATION MAP GENERATION

1) MULTI-FLOOR MAP ALIGNMENT

As shown in Fig. 3, the multi-floor map is generated in two stages: generating single-floor maps and then generating the multi-floor map through stacking these maps. The single-floor high-definition map is generated using the lightweight and ground-optimized LOAM (LeGO-LOAM) [33] method from the LiDAR Odometry And Mapping (LOAM) [34] series, which is commonly used in mobile robotics for map generation.

To stack the single-floor maps, a registration method based on the Normal Distribution Transform (NDT) is utilized [35], [36]. There are overlapping areas between the k-th floor and the (k-1)-th or (k+1)-th floor, and in these overlapping areas, the point cloud maps of the two floors have the same points. These include areas such as stairs between floors and areas that penetrate the building vertically. The key to the stacking method is to determine the rotation matrix $\mathbf{R}_k$ and the translation vector $\mathbf{t}_k$ using registration in these areas to stack the map of the k-th floor $\mathbf{M}_k$ on top of the accumulated map $\mathbf{U}_{1:k-1}$. Here, $\mathbf{U}_{1:k-1}$ is a multi-floor map accumulated from the 1st floor up to the (k-1)-th floor. The point clouds corresponding to the same area in both $\mathbf{M}_k$ and $\mathbf{U}_{1:k-1}$ are extracted from each map. These are denoted as $\mathbf{P}_{\mathbf{M}_k} = \{p_{\mathbf{M}_k,1}, p_{\mathbf{M}_k,2}, \cdots, p_{\mathbf{M}_k,n}\}$ and $\mathbf{P}_{\mathbf{U}_{1:k-1}} = \{p_{\mathbf{U}_{1:k-1},1}, p_{\mathbf{U}_{1:k-1},2}, \cdots, p_{\mathbf{U}_{1:k-1},n}\}$, respectively. Then, the transformation of $\mathbf{M}_k$ to $\mathbf{U}_{1:k-1}$ can be redefined as the transformation between $\mathbf{P}_{\mathbf{M}_k}$ and $\mathbf{P}_{\mathbf{U}_{1:k-1}}$. The transformation error $\mathbf{d}_k$ can be defined as follows:

$$d_k = \sum_{\forall j}((\mathbf{R}_k p_{\mathbf{M}_k,j} + \mathbf{t}_k) - p_{\mathbf{U}_{1:k-1},j}). \qquad (1)$$

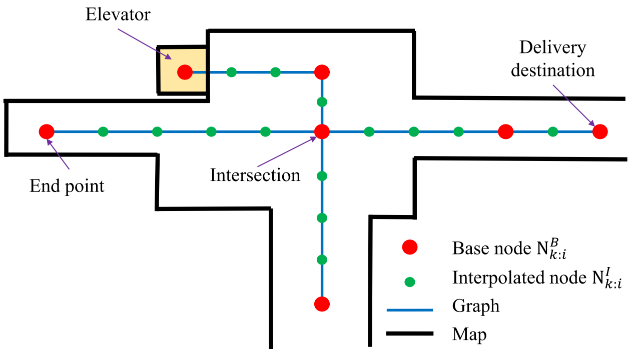


FIGURE 4. Conceptual illustration of the base and interpolated nodes and graph, which are elements of a topological map.

To minimize d_k, registration is iteratively performed until $d_k < \delta$ (where δ is a threshold) to find $\mathbf{R}_k$ and $\mathbf{t}_k$. In our work, the threshold of registration convergence score for transformation is chosen to be 0.2. Afterwards, the $\mathbf{t}_{new}$ is defined by adding the inter-floor gap z_{bias} to $\mathbf{t}_k$. $\mathbf{M}_k$ transformed by $\mathbf{R}_k$ and $\mathbf{t}_{new}$ into $\mathbf{M}'_k$ is then merged with $\mathbf{U}_{1:k-1}$ to generate $\mathbf{U}_{1:k}$ which includes up to the k-th floor. More detailed procedures can be found in our previous work [37], [38].

2) TOPOLOGICAL MAP GENERATION

We propose a topological graph map of the k-th floor $\mathbf{G}_k$ for the multi-floor route planning. While building $\mathbf{M}_k$, we register the base nodes $\mathbf{N}^B_k = \{n^B_{k,1}, n^B_{k,2}, \cdots, n^B_{k,n}\}$ which can serve as references for graph generation. $\mathbf{N}^B_k$ consists of four types:

- End-point corresponding to the end of the corridor
- Intersection where multiple branches are possible
- Point on an elevator
- Delivery destination

After building a $\mathbf{N}^B_k$, we interpolate $\mathbf{N}^B_k$ to generate $\mathbf{N}^I_k = \{n^I_{k,1}, n^I_{k,2}, \cdots, n^I_{k,n}\}$ for generating a dense points trajectory. Thus, the graph map for each floor $\mathbf{G}_k$, is defined as $\{\mathbf{N}^B_k, \mathbf{N}^I_k\} \subset \mathbf{G}_k$ and the cumulative graph map up to the k-th floor is defined as $\mathbf{G}_{1:k}$ as depicted in Fig. 4. Base nodes are registered on the point cloud map, and when the point cloud map of the corresponding floor is loaded, the nodes are interpolated to generate a graph. The graph maps are used for route planning, and routes across different floors are connected through elevator nodes.

We use OpenStreetMap [39] (OSM) and the Java OpenStreetMap Editor for convenient management of base nodes. Selecting base nodes and managing them through OSM offers several advantages. It facilitates the management and registration of delivery destinations when performing delivery service. Also, it is easy to edit the route that the robot will take. In addition, it is easy to add attributes to the nodes described later.

These nodes are only generated in areas where the robot can move. Therefore, when utilizing the graph connected

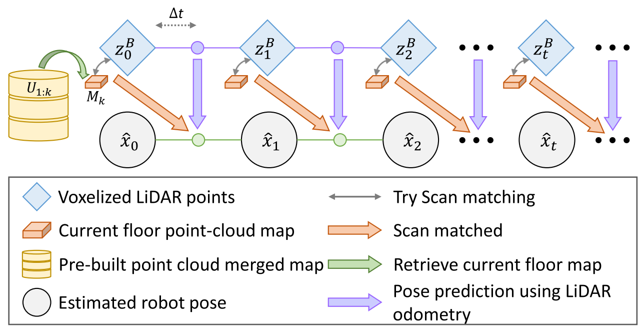


FIGURE 5. **The structure of localization.**

by these nodes for route planning, it naturally prevents the generation of routes in areas where the robot cannot drive.

Nodes have attributes that include the floor on which they were generated. However, some nodes have additional attributes depending on their role. Intersection nodes have information about the number of branches, and elevator nodes have an attribute identifying them as elevator nodes. Ordinary nodes have two direction attributes for the graph to be connected on both sides, whereas end-point nodes have only one direction attribute.

We define the map that combines the merged point cloud map and the topological graph map as an integrated navigation map.

3) LOCALIZATION

The robot's position is determined by scan-matching-based localization within the map M_k of each floor. However, there may be cases where this method fails. Therefore, LiDAR odometry has been combined to improve the robustness of localization.

Let $\mathbf{x}_t^M = [x_t, y_t, \theta_t]$ denote the state of the robot, with x_t and y_t representing the robot's position and θ_t is its heading in the 3D point-cloud map at time t. M is defined as the map coordinates. We also define the voxelized LiDAR points at time t as $\mathbf{z}_t^B$. B is defined as the robot body coordinates.

The robot's position $\mathbf{x}_t^M$ can be estimated by minimizing the distance between $\mathbf{z}_t^M$, which is converted from the body coordinate $\mathbf{z}_t^B$ using the transformation matrix $\mathbf{T}_t$, and the current floor's point cloud map M_k. This can be seen as a registration problem. The reason for using only the current floor map M_k instead of the merged map $U_{1:k}$ is to prevent registration with similar environments on other floors. For point-cloud registration, we employ the generalized iterative closest point (GICP) method [40]. This approach represents the surface from which a point is derived as a Gaussian distribution: $\mathbf{z}_t^M \sim \mathcal{N}(\hat{z}_i, C_i^z)$, $M_k \sim \mathcal{N}(\hat{m}_i, C_i^m)$. Then, the transformation error d_i can be defined as follow:

$$d_i = \hat{m}_i - \mathbf{T}_t \hat{z}_i. \tag{2}$$

The distribution of d_i can be described as

$$\begin{aligned} d_i &\sim \mathcal{N}(\hat{m}_i - \mathbf{T}_t \hat{z}_i,\ C_i^m - \mathbf{T}_t^T C_i^z \mathbf{T}_t) \\ &= \mathcal{N}(0,\ C_i^m - \mathbf{T}_t^T C_i^z \mathbf{T}_t). \end{aligned} \tag{3}$$

The GICP finds the transformation $\mathbf{T}_t$ that maximizes the log likelihood of (3) as follows:

$$\begin{aligned} \mathbf{T}_t &= \arg\max_{\mathbf{T}_t}(log(p(d_i))) \\ &= \arg\min_{\mathbf{T}_t}(\sum_i d_i^T (C_i^m - \mathbf{T}_t^T C_i^z \mathbf{T}_t)^{-1} d_i). \end{aligned} \tag{4}$$

For real-time operation, the FAST-GICP (voxelised GICP) method, one of the variants of the GICP method, is adopted [41].

As the map becomes larger, the registration algorithm becomes more computationally complex, as the number of points to be compared increases. This can lead to occasional incorrect registration. In addition, failure can also occur when the robot's movement becomes sudden. Therefore, we implement a sliding window to reduce the range of the target point cloud. Also, we calculate the LiDAR odometry based on the NDT method, which is robust to changes in scan data within a short interval to handle the robot's dynamic movement.

The sliding window involves utilizing only the points within a radius r from the robot's current position $\mathbf{x}_t^M$ in the point cloud map M_k. Then, points that are too far from the robot do not need to be compared, and the calculation complexity is also decreased, which can reduce the probability of alignment failure.

The registration involves iteratively finding T_t from the robot's starting point, so the robot's position in the previous step is estimated as T_{t-1}. We calculate LiDAR odometry for a short time Δt and use a predictive transformation $T_{t+\Delta t}$ instead of T_{t-1} for the next step of registration. This approach has the effect of preventing the next step of registration from diverging and re-calculating the robot's position when (2) does not converge and has a large error. The robust localization algorithm is illustrated in Fig. 5.

B. PATH PLANNING

1) 3D ROUTE PLANNING

This study introduces a 3D route planning algorithm that leverages elevator nodes for expandable traversal through multi-floor environments. The proposed method integrates a depth-first search (DFS) algorithm for efficient navigation through the interconnected floors of a building.

The proposed algorithm addresses the challenges of multi-floor navigation by integrating elevator nodes, enabling vertical traversal. The elevator nodes are strategically integrated into the constructed nodes N_k, ensuring that spatial information for each floor is interconnected through elevators. The use of elevators optimizes the overall route planning process by reducing the complexity associated with repetitive floor-by-floor planning. The path $P_{a\to b}^{global*}$ from floor a to floor b is defined as follows:

$$P_{a\to b}^{global*} = \arg\min_{\forall i,j \in G_{1:k}}(f_{a,i} + f_{b,j} + g_{elevator}) \tag{5}$$

where $f_{k,i}$ is the cumulative distance function on floor k, and $g_{elevator}$ is the function for floor transition enabling vertical node expansion in the pathfinding process.

The function $g_{elevator}$ for expanding elevator nodes can be refined to reflect the specific characteristics of vertical movement. Let $V_{elevator}$ denote the set of elevator nodes. The function $g_{elevator}$ incorporates the cost associated with moving between elevator nodes on different floors by introducing a weighting factor $w_{elevator}$.

$$g_{elevator}(n_i, n_j) = w_{elevator} \cdot \text{cost}(n_i, n_j) \tag{6}$$

where n_i and n_j represent nodes connected by an elevator, and $\text{cost}(n_i, n_j)$ denotes the cost associated with transitioning between these elevator nodes.

The DFS algorithm is employed in our approach to efficiently explore and navigate the interconnected graph nodes. DFS facilitates a systematic traversal of the graph structure, exploring as far as possible along each branch before backtracking. This characteristic of DFS contributes to the effectiveness of the proposed 3D route planning method by ensuring that the algorithm systematically and thoroughly explores available paths. The DFS algorithm is implicitly integrated into the entire route planning process, contributing to the systematic exploration of the interconnected multi-floor graphs in (5).

To explicitly introduce the DFS algorithm into the path planning equation, we can represent the recursive nature of DFS. Let $N_{visited}$ denote the set of visited nodes during the DFS traversal. The recursive DFS equation can be expressed as follows:

$$\text{DFS}(n_i) = n_i \cup \left(\bigcup_{n_j \in \text{neighbors}(n_i) \setminus N_{visited}} \text{DFS}(n_j) \right). \tag{7}$$

This equation defines the DFS traversal starting from node n_i, exploring its neighbors recursively while avoiding previously visited nodes. The set union operation ($\cup$) ensures the accumulation of visited nodes.

The overall cost function $f_{k,i}$ for a given floor k in the path planning equation can be decomposed into individual components, such as distance traveled within the floor ($f_{k,i}^{N_k}$), elevator node expansion cost ($g_{elevator}$), and any other relevant factors:

$$f_{k,i} = f_{k,i}^{N_k} + g_{elevator}. \tag{8}$$

This decomposition provides transparency into the contributing factors influencing the cost function.

2) LOCAL PLANNING

For autonomous navigation in indoor environments, the ability to avoid obstacles that may exist in its path is essential. Therefore, real-time local path planning is utilized to enable obstacle avoidance. Our research team previously proposed a system capable of operating in complex environments with various obstacles [3], but we discovered limitations in indoor environments, especially when riding an elevator. In the previous research, a local planner based on motion primitives was used, but in situations like elevators or narrow areas, there were cases where all primitives were infeasible for travel, making it impossible to choose samples for avoidance maneuvers. To overcome these limitations, a time elastic band (TEB) based local path planning algorithm was utilized [42]. This algorithm enabled the generation of local paths considering speed, angular velocity, and obstacles. Additionally, although the robot was a differential drive type and thus had no kinematic constraints, we implemented path generation considering kinodynamics applicable to car-like models for use in tight spaces like elevators. Kinodynamic motion planning generates a path when various dynamic obstacles are included in the input, however, when multiple dynamic obstacles are considered, the path sometimes becomes longer than necessary. To solve this problem, as described in Algorithm 1, we modified the algorithm to perform re-planning in such situations, allowing the robot to generate easily navigable paths again.

Algorithm 1 Online TEB Local Planner

```
1:  procedure tebalgorithm($traj, p_c, p_g, O, robot\_model$) ▷ Invoked each interval
2:      Initialize or update trajectory
3:      for Iteration 1 to $I_{teb}$ do
4:          Adjust length $n$ of the trajectory
5:          Update obstacles constraints from set $O$
6:          $traj^* \leftarrow$ CallOptimizer($traj$)    ▷ solve Nonlinear least-square function
7:          if $length\_of\_traj^* > threshold$ then
8:              Replanning
9:          Check feasibility
10:     return First (sub-) optimal control inputs ($v, w$)
```

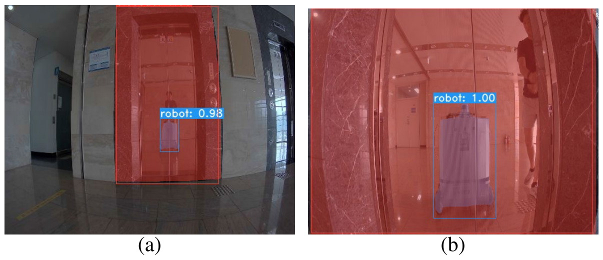


FIGURE 6. Semantic segmentation result for elevator and robot detection. (a) segmentation results of the elevator and robot when the robot is far from the elevator. (b) results of the robot segmenting itself from the image reflected on the door when the robot is close to the elevator.

C. PERCEPTION

1) ELEVATOR DETECTION

The relative position of the elevator is determined through camera-LiDAR sensor fusion. For accurate elevator door recognition, the YOLACT image semantic segmentation algorithm is used, known for its fast real-time recognition speed.

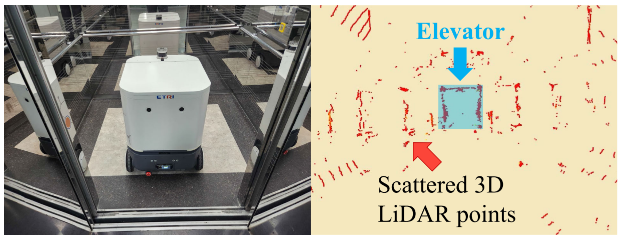


FIGURE 7. Challenging scenario inside the elevator. Inside an elevator made of mirrors and scattered point cloud data.

The elevator door in the building where the research was conducted was made of highly reflective metal. When the robot approached the elevator door, the camera image recognized reflections of the surroundings instead of the elevator door. Therefore, the robot was trained to detect the reflected robot itself to use the pixels corresponding to the robot to determine the goal point. As shown in Fig. 6, not only the elevator doors but also the robot itself is detected. However, if the robot encounters a mirror while navigating, it can mistake the mirror for an elevator door because it can see itself. Therefore, it only determines to be an elevator only when the robot recognizes itself near the elevator node on the global map.

We use a semantic segmentation algorithm instead of an object detection algorithm such as YOLO [43] which produces bounding box results. Because the elevator does not always appear as a rectangle in images depending on the viewing angle, and the robot also has an irregular shape. When matching the LiDAR point cloud to the image detection result, if the bounding box is used for matching, point clouds corresponding to the wall pixels are also extracted. This leads to the center of the point cloud cluster being calculated further from the expected point. To prevent this, the semantic segmentation algorithm is used that can extract pixels in the shape of the detected object.

$$x_{cam} = \tan\left(\frac{\theta}{2}\right)\frac{2u-w}{w}z_{cam} \tag{9}$$

$$y_{cam} = \tan\left(\frac{\alpha}{2}\right)\frac{2v-h}{h}z_{cam}. \tag{10}$$

The pixels corresponding to the elevator are matched with the LiDAR point clouds through equations (9) and (10). Afterward, they are used to set a 3D region of interest (ROI) for extracting only the points corresponding to the elevator, following a coordinate transformation using the camera-LiDAR transformation matrix. In this context, $x_{cam}, y_{cam}, z_{cam}$ represent the position in the camera coordinate system, while θ, α represent the camera's horizontal and vertical field of view, respectively. u, v denote the pixel's position on the camera's image plane, and w, h are the width and height of the camera's image, respectively.

The center point of the LiDAR points corresponding to the elevator is set as the elevator position $\mathbf{P}^{elev}_{lidar}$. Similarly, the center point of the LiDAR points corresponding to the robot reflected on the elevator door is also set as the position

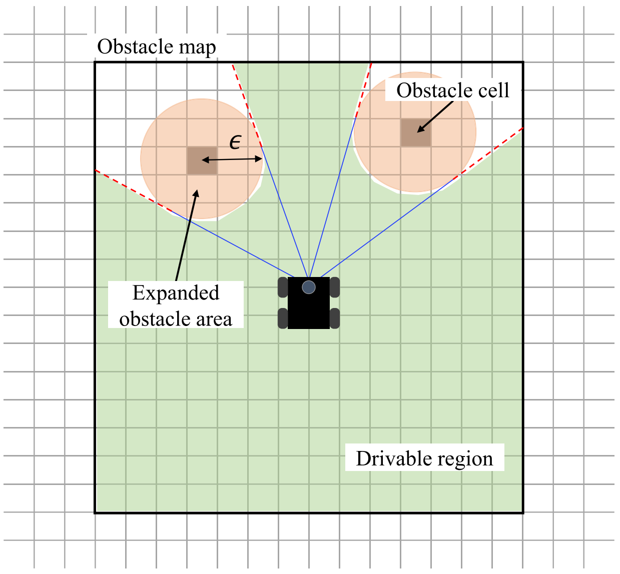


FIGURE 8. Drivable region on obstacle map. The orange area indicates an expanded obstacle area with a radius ϵ. The area around the robot, excluding the expanded obstacle area and its behind, is designated as the drivable region (marked in green).

of the reflected robot $\mathbf{P}^{reflected\_robot}_{lidar}$. When the robot boards the elevator, the closest point to $\mathbf{P}^{elev}_{lidar}$ or $\mathbf{P}^{reflected\_robot}_{lidar}$ in the drivable region is set as the goal point.

2) DRIVABLE REGION EXTRACTION

As shown in Fig. 7, when the interior of the elevator is made of highly reflective material, the data from the 3D LiDAR sensor becomes scattered, causing localization inaccuracies when employing scan-matching within the elevator. Additionally, the appearance of LiDAR points beyond mirrors complicates the generation of an obstacle map for extracting drivable regions. Moreover, since people can ride in the elevator together, using artificial markers inside the elevator for localization is also challenging, as the markers can be obscured by people. Therefore, in this study, we use a method that employs a 2D laser scanner sensor, which has a lower scattering effect compared to 3D LiDAR, mounted on the lower part of the robot to detect drivable regions.

An occupancy grid map of size $5m$ by $5m$ centered around the robot is designated as the region of interest. The size of one cell in the grid map is set to $0.05m^2$. Detected points from the 2D laser scanner sensor are considered obstacles and registered in cells to generate an obstacle map. The registered cells are expanded using the following equation to calculate the cost of cells within a predefined range ϵ around them:

$$M^{cost}_{x,y} = \sum_{k=1}^{n}\left\{w * \sqrt{(\epsilon - |x - x_k|)^2 + (\epsilon - |y - y_k|)^2}\right\}. \tag{11}$$

where n is the number of obstacle cells in the obstacle map, x_k and y_k are the coordinates of the obstacle cell, x and y

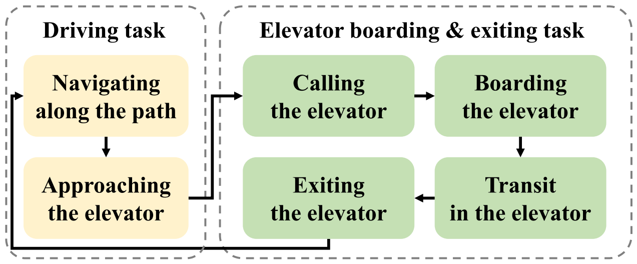


FIGURE 9. Pipeline of elevator boarding and exiting task.

denote the coordinates of all cells within the ϵ range, and w is a constant coefficient. The expanded obstacle area depicted in Fig. 8 is the range of ϵ and the total sum of the expanded obstacle areas for all k points becomes the cost map $M^{cost}_{x,y}$ as expressed in (11).

The drivable region $M^{drv}_{x,y}$ is defined as the set of points on the line connecting the robot's center point to the obstacle map boundary points up to the point where it meets the expanded obstacle area. The drivable region is used to set the goal point in the elevator boarding task and the local path planning.

D. TASK PLANNING

In this study, the process of last-mile delivery service is divided into four task states. Each task consists of the following:

- Driving task: Moving with the parcel.
- Elevator boarding & exiting task: For inter-floor movement.
- Delivery task: Delivering the parcel to the recipient at the destination.
- Docking task: Returning to the docking station to wait or reload parcels after completing the delivery.

The robot provides delivery service by repeatedly performing these tasks. The following describes each task in detail.

1) DRIVING TASK

The driving task is the most basic state, performing the task of driving along a given path while conducting map-based localization. The driving task is the default state, except when performing the elevator boarding task, the delivery task at the destination, and the docking task at the docking station, as described below.

2) ELEVATOR BOARDING AND EXITING TASK

The driving task applies a localization algorithm using the integrated navigation map. However, as mentioned in the perception section, the elevator boarding task is performed independently based on the drivable region due to the inaccuracy of localization inside the elevator. Therefore, after getting off the elevator, the process of reloading the integrated navigation map and initializing localization on the map is performed.

TABLE 1. Docking process states.

State	Description
1	Approach from a long distance (detect outside marker)
2	Approach from a close distance (detect inside marker)
3	Stop at the docking point
4	Rotate 180 degrees for parcel loading
5	Parcel loading
6	Departure after loading
7	Change mode from docking to driving

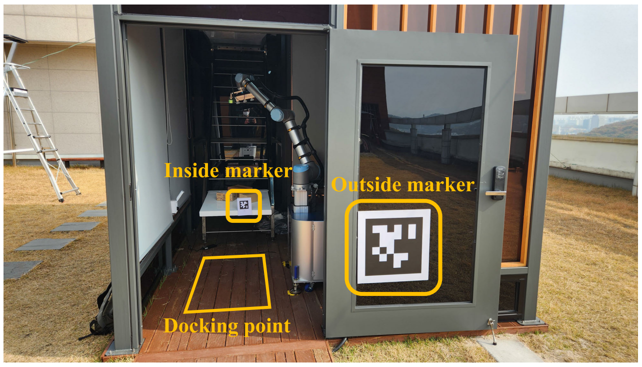


FIGURE 10. Outside marker and inside marker attached to the rooftop docking station.

In this study, the elevator task is performed using Internet of Things (IoT) communication based on a Request/Response mechanism utilizing the elevator API. Therefore, the robot can call the elevator without directly pressing the elevator buttons and the destination floor button can be pressed automatically. Fig. 9 depicts the pipeline for the elevator task. When the mobile robot receives a parcel at the docking station, the floor number it needs to visit is determined based on the recipient's address (room number within the building). When inter-floor movement is required, the robot first navigates toward the elevator node based on the global map and then switches the task to the elevator boarding/exiting task. A request to call the elevator is sent, and once the elevator arrives, boarding is carried out after checking the drivable region. It is important to judge whether the robot has successfully entered the elevator during the boarding process. If the maximum distance r_{max} from the robot's center to the boundary of the drivable region is less than the user-defined threshold γ, it is determined that the robot is aboard the elevator and the door is closed. After boarding, the robot exits when the floor information provided by the API matches the destination floor. Upon reaching the destination floor, the robot pose is initialized in front of the elevator to enable navigation based on the integrated navigation map for resuming delivery service.

3) DELIVERY TASK

The delivery task is performed by communicating with the control center via TCP/IP when departing from the docking station and upon arrival at the destination. This task consists of the following behavior: sorting the destination, sending a

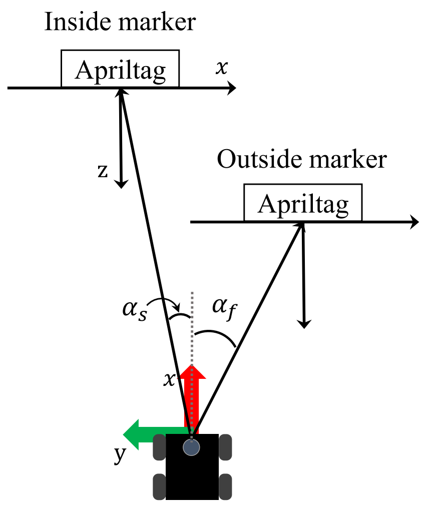


FIGURE 11. Calculate relative distance and angle from Apriltag marker for docking.

text message to the recipient, waiting, delivering the parcel to the recipient or passing the destination if the recipient is absent, and returning to the docking station after delivery completion.

When departing from the docking station, the robot receives destination information from the control center. Since the robot starts from the rooftop, it initially sorts the destination information from the higher to the lower floors. The remaining behaviors are performed sequentially in conjunction with the control center upon arrival at each delivery destination.

When the robot arrives at the delivery destination, a text message is sent to the parcel recipient from the control center, and the robot switches to a standby state. If the recipient checks the message and confirms that they will receive the parcel, the control center sends a command to the robot to open the parcel box door. If the recipient does not respond within a certain time, the recipient is considered absent and the robot moves on to the next destination. After all deliveries are complete, the robot switches to return mode and moves to the docking station on the rooftop.

4) DOCKING TASK

For parcel delivery, a hub is needed where the robot can receive parcels. Such a hub is set up on the rooftop of a building and used as the robot's docking station. Fig. 10 shows the appearance of the docking station. There is a gate on the roof of the station, so when the drone drops parcels through it, a manipulator inside the station picks them up and puts them into the mobile robot. The mobile robot performs docking at this station according to the states as shown in Table. 1.

Since there is no ceiling to block the sunlight on the rooftop, the luminance changes over time. However, the manipulator inside the station needs to pick parcels based on vision, requiring minimal changes in luminance. Therefore,

Algorithm 2 Docking Process

Input: Outside marker pose from front camera p_o^f, inside marker pose from front camera p_i^f, the time when the outside marker was detected using the rear camera t_o^r, the time when the inside marker was detected using the rear camera t_i^r, threshold approach distance to outside marker α, threshold approach distance to inside marker β, leaving time threshold γ

Output: Goal pose of robot p_g

```
1:  procedure docking($p_i$, $p_o$)
2:      Detect outside marker from a long distance
3:      if $p_o^f < \alpha$ then
4:          Change the state to 1
5:          Approach the station entrance
6:          if $p_i^f < \beta$ then
7:              Change the state to 3
8:          else
9:              Change the state to 2
10:             Approach the docking point
11:     if Time when the state is 3 > 3 sec then
12:         Change the state to 4
13:         Detect inside marker using the rear camera
14:         Open the cover for parcel loading
15:     if $t_i^r > \gamma$ then
16:         if Parcel loading is done then
17:             Change the state to 6
18:         else
19:             Change the state to 5
20:     if State number is 6 and $t_o^r > \gamma$ then
21:         Change the state to 7
```

Each state has a goal pose p_g appropriate for the situation.

blackout curtains are installed in the station, and as a result, when the robot arrives on the rooftop, the inside of the station is dark and the docking point can not be clearly seen. Therefore, there is a need to guide the robot to a position where it can see inside the station. For this purpose, two artificial markers in the form of Apriltag [44] are used for docking.

As shown in Fig. 11, the robot approaches the station's entrance using a large marker attached to the outside. Then, the robot drives slowly to accurately locate the docking point while looking at the inside marker. Initially, the robot adjusts its angle α_f using the pose between the robot and the outside marker to face the entrance of the docking station. Once it reaches the threshold distance, the robot detects the inside marker and then moves to the docking position, adjusting its angle α_s accordingly. When the inside marker is detected by the front camera and the inside marker pose from the front camera is calculated, while the robot continues to approach the entrance of the station and the outside marker becomes invisible in the field of view of the front camera, the distance

TABLE 2. The results of a month-long field test in terms of localization's mean error, number of elevator boarding, sequence of delivery floors, number of parcels delivered, daily delivery distance, and docking outcomes.

Test date	Mean error (m)	# of elevator boarding	Sequence of delivery floors	# of parcels delivered	Daily delivery distance (m)	Docking outcomes
Nov 01	0.1566	5	8-7-4-8 8-7-8	3	389.31	success
Nov 02	0.1497	9	8-2-8 8-2-8 8-6-4-8 8-5-8	5	553.57	success
Nov 03	0.1879	5	8-6-3-8 8-5-8	3	402.35	success
Nov 06	0.1344	4	8-4-8 8-4-8	3	273.48	success
Nov 07	0.3214	7	8-7-6-5-8 8-6-4-8	5	502.27	success
Nov 09	0.1933	6	8-7-3-8 8-3-2-8	4	457.26	success
Nov 10	0.1867	5	8-4-3-8 8-4-8	3	409.69	success
Nov 13	0.3363	4	8-7-3-2-8	3	296.32	success
Nov 14	0.1016	6	8-6-5-4-8 8-2-8	4	418.64	success
Nov 15	0.1175	6	8-7-6-5-3-2-8	5	378.45	success
Nov 20	0.0553	3	8-7-4-8	3	223.04	success
Nov 21	0.0749	4	8-7-5-3-8	3	316.1	success
Nov 22	0.1475	7	8-2-8 8-7-6-8 8-7-8	5	392.39	success
Nov 23	0.0567	7	8-7-5-4-3-8 8-2-8	5	404.96	success
Nov 24	0.0495	2	8-2-8	1	150.37	success

from the inside marker is then determined as the threshold distance value.

As described in Algorithm 2, we define a process whereby the robot changes its docking state using the relative pose of the marker and the robot assigns a corresponding goal pose p_g for each state, allowing the robot to stop at the docking point, receive the parcel, and then continue with the delivery.

V. EXPERIMENTS

In this study, to validate the proposed system architecture, a parcel delivery service demonstration experiment was conducted at the Electronics and Telecommunications Research Institute (ETRI) in South Korea. For about a month, a parcel delivery service was provided to people in a building of the research institute. The service was conducted for two hours in the afternoon when most people were present at work. Considering the battery consumption of the robot, the tests were divided into one-hour segments and always started from the 8th floor where the docking station was located. When the drone delivered parcels to the station, the manipulator loaded them into the mobile robot. The sequence of the delivery floor was arranged in descending order according to the recipient's address. After the delivery, the robot returned to the station on the 8th floor. While performing the parcel delivery service, the performance of localization was also measured, and the number of elevator boarding, the number of parcels delivered, and the daily delivery distance were recorded.

As shown in Table. 2, during the entire period, the mean error of localization was below 0.4 m and the average was 0.1513 m. The robot boarded the elevator a total of 80 times, delivered 55 parcels, and traveled a total of 5568.2 m during the test period. Moreover, all docking tasks were successful during the delivery service. The provision of the parcel delivery service enabled the validation of all components of the proposed system architecture. Localization within the building, route planning in a multi-floor environment using a node-graph-based topological map, elevator detection and drivable region extraction for elevator boarding and local path planning, as well as last-mile delivery and docking through task planning were all performed without human intervention. Experimental verification of each element is discussed in detail below.

The generation of the integrated navigation map was verified in two buildings, and the results are shown in Fig. 12. As shown in Fig. 12(a), multi-floor maps were generated using the proposed map alignment method from a single-floor point cloud map. Building A had a total of three floors and building B had a total of eight floors. As can be seen in the top view images, the common areas of each floor were overlapped and aligned at the same location. Similarly, as depicted in Fig. 12(b), the topological map was also successfully generated on all floors, where robots could navigate. The nodes were represented by red dots, and it was observable that these nodes corresponded to their respective floors on the point cloud map.

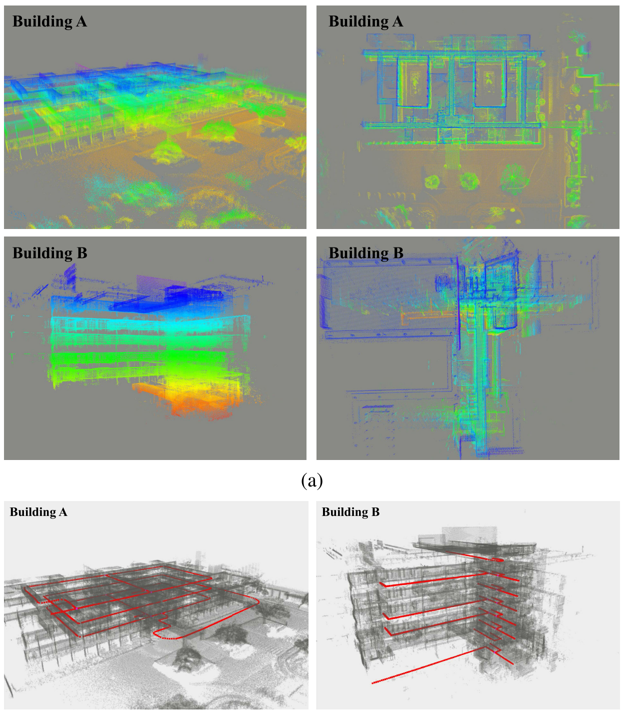


FIGURE 12. Result of integrated navigation map. (a) The map was generated using the proposed multi-floor map alignment method. (b) The proposed integrated navigation map is illustrated.

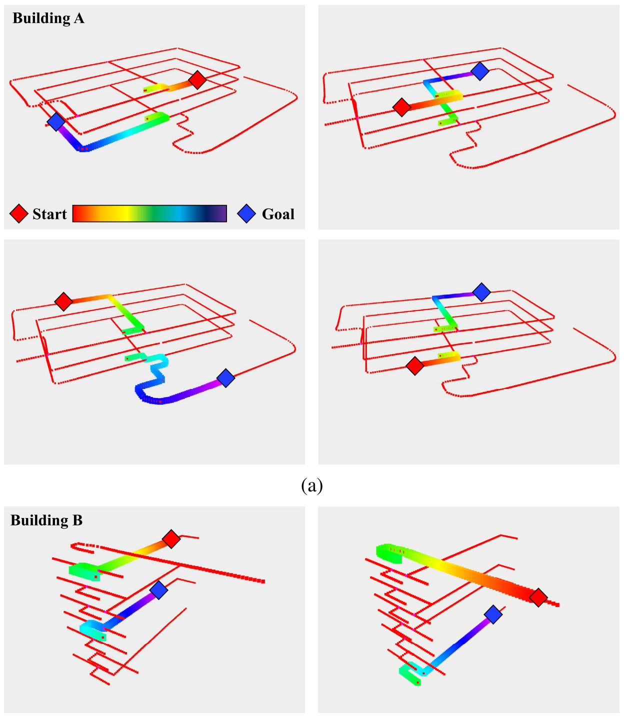


FIGURE 13. Result of multi-floor route planning algorithm. Examples of routes are illustrated by rainbow colors in two buildings: (a) building A. (b) building B.

Next, we verified whether route planning within the building, using the node information from the generated topological map, could calculate routes from the current robot's position to the destination. The results are presented in Fig. 13. The red dots represent the elements that make up the topological map, with Fig. 13(a) corresponding to a 3-floor building A, and (b) to an 8-floor building B. Since the purpose was to verify that routes were correctly generated in a multi-floor environment, the destinations were designated to floors different from the start floor. Due to the connectivity of the elevator nodes located on each floor, the route could be extended to different floors, and the generated route enabled the execution of the elevator boarding task. The generated route was indicated in rainbow colors, with the route extending from the start point to the elevator node on the same floor, then recommencing from the elevator node on the floor where the goal point was located and continuing to the goal point.

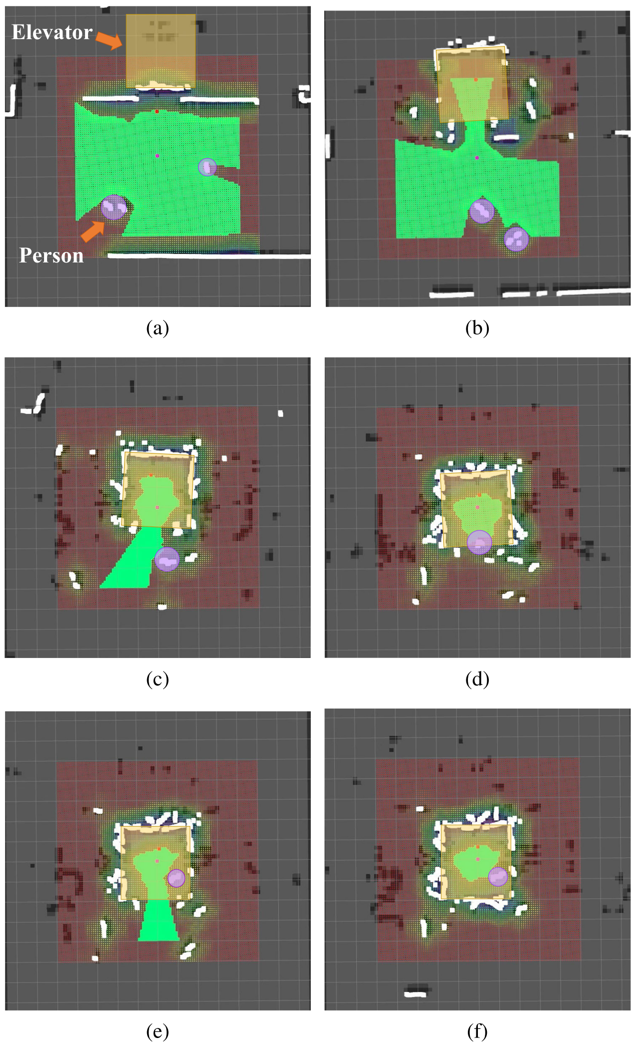


FIGURE 14. Result of drivable region extraction with cost map. (a) The robot is waiting for the elevator (door is closed) (b) The robot is boarding the elevator (door is opened). (c) The robot is got on the elevator (door is opened). (d) A passenger is boarding the elevator (The robot is in the elevator). The robot and a passenger are in the elevator together. ((e) door is opened) ((f) door is closed).

Validation was conducted on the extraction of the drivable region, which is an important part of the elevator boarding

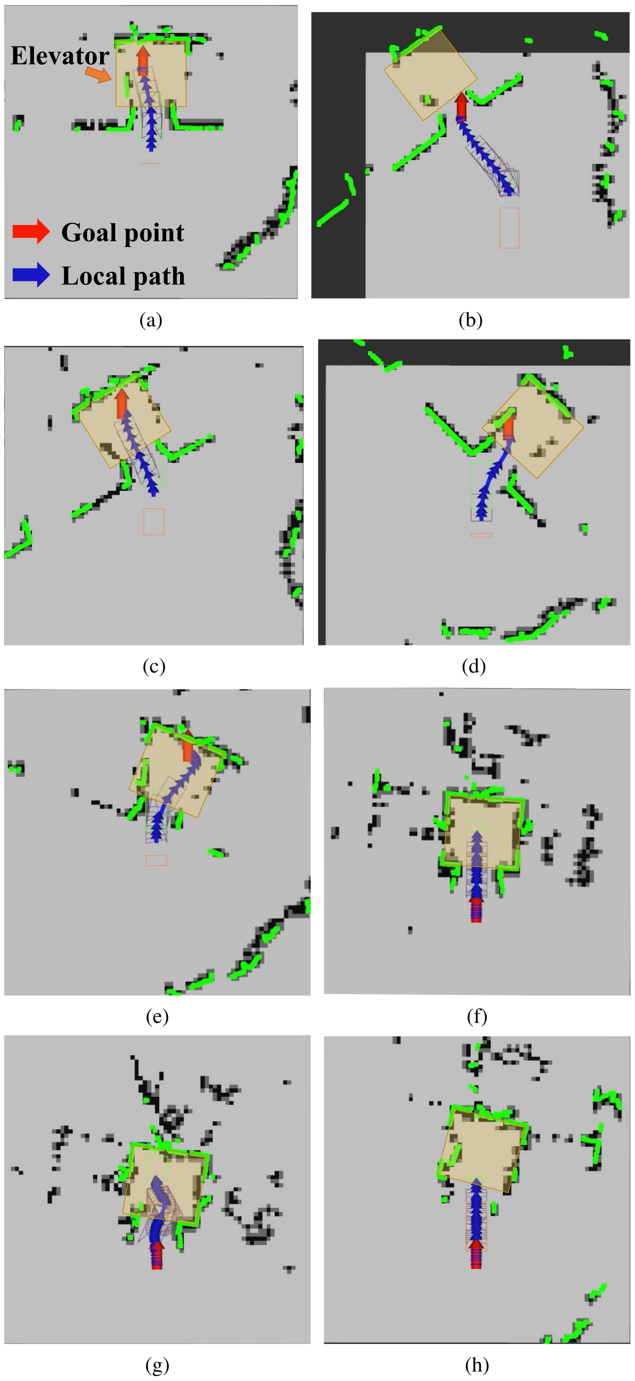


FIGURE 15. Various test scenarios for local path planning. (a)-(e) Cases where the local path is generated in forward directions while the robot is boarding the elevator. (f)-(h) Cases where the local path is generated in the reverse direction while the robot is exiting from the elevator.

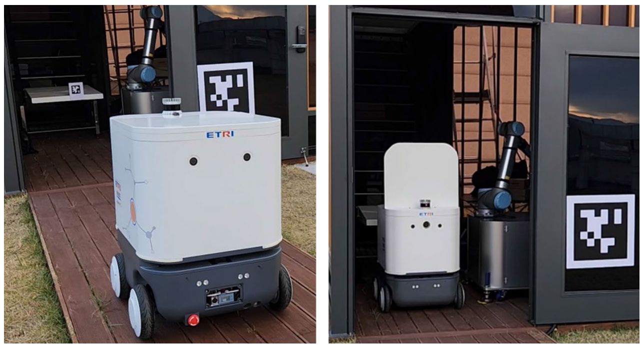


FIGURE 16. A snapshot of entering the docking station and waiting for the parcel after stopping at the docking point.

task, and the results are illustrated in Fig. 14. In the cost map surrounding the robot red indicates a low cost and colors closer to blue signify a higher cost. The drivable region represented in green is generated using the cost value. In Fig. 14(a), which shows the robot in a waiting state before the elevator doors open, it can be seen that the drivable region is not generated behind a person recognized as an obstacle. Figs. 14(b) and 14(c) depict the situation after the elevator doors open and the robot boards the elevator. Even though the points of the internal boundary of the elevator were not all detected, the cost map effectively ensured that the drivable region was safely generated without penetrating the elevator wall. Figs. 14(d) - 14(f) show a person boarding the elevator with the robot. The absence of a drivable region near the person indicates that the robot can safely board the elevator with people. As shown in Fig. 14(f), the drivable region was generated only inside the elevator after the elevator door closed. Through this, it was demonstrated that the robot's elevator boarding judgment was working.

To validate the robustness of local path planning in narrow environments, the algorithm was evaluated in an elevator boarding scenario. The results of the generated local path planning are indicated by blue arrows in Fig. 15. After moving to the front of the elevator door and recognizing its opening, the algorithm determined the drivable region and the destination inside the elevator and generated a local path. To verify that the local path is generated reliably in various situations, the robot was positioned to face the elevator from different directions, and the case of exiting the elevator was also examined. It was observed that the local path was stably generated until the robot boarded the elevator from the front, as well as from the left and right. Additionally, it was observed that the robot generated a local path towards the rear when it exited. Similarly, when a person exited the elevator, the robot generated a local path to exit without collisions.

To validate the docking task, an experiment was conducted where the robot automatically repeated the process of entering the docking station and receiving parcels. Docking was always performed throughout the field tests. The behaviors of automatically receiving the parcels, departing, and returning after completing the delivery were all performed at the station. As shown in Fig. 16, it was observed that the docking task started from the moment the robot recognized the outside marker, and it was performed stably until the robot stopped exactly at the docking point and waited to receive the parcel from the manipulator. After receiving the parcels, it was also observed that the robot smoothly transitioned from the docking task to the driving task when the rear camera recognized the outside marker as it left the station.

VI. CONCLUSION

In this study, we developed an indoor delivery mobile robot and an architecture for a multi-floor building environment. By integrating four key modules - integrated navigation map generation, path planning, perception, and task planning - we have developed a robust system capable of navigating complex indoor multi-floor environments efficiently. Our architecture utilized an integrated navigation map that combines multi-floor map generation and node-graph-based topological map for accurate localization and effective route planning across different floors. The successful implementation of the elevator boarding method and task planning strategies further enhanced the practicality of this system in real-world scenarios. From loading the parcel at the docking station to delivering it to the recipient, the architecture demonstrated through extensive field testing provides an advancement in delivery service, operating seamlessly in ordinary buildings.

REFERENCES

[1] A. Grau, M. Indri, L. Lo Bello, and T. Sauter, "Robots in industry: The past, present, and future of a growing collaboration with humans," *IEEE Ind. Electron. Mag.*, vol. 15, no. 1, pp. 50–61, Mar. 2021.

[2] C. S. Chen, C. J. Lin, and C. C. Lai, "Non-contact service robot development in fast-food restaurants," *IEEE Access*, vol. 10, pp. 31466–31479, 2022.

[3] D. Lee, G. Kang, B. Kim, and D. H. Shim, "Assistive delivery robot application for real-world postal services," *IEEE Access*, vol. 9, pp. 141981–141998, 2021.

[4] S. Mukherjee, M. M. Baral, C. Venkataiah, S. K. Pal, and R. Nagariya, "Service robots are an option for contactless services due to the COVID-19 pandemic in the hotels," *Decision*, vol. 48, no. 4, pp. 445–460, Dec. 2021.

[5] M. Tsunoda and C. Premachandra, "Remote control of a wheeled robot by visible light for support in infectious disease hospitals," *IEEE Access*, vol. 9, pp. 124165–124175, 2021.

[6] I. T. Lakmal, K. L. A. Nuwan Perera, H. A. Harindu Y. Sarathchandra, and C. Premachandra, "SLAM-based autonomous indoor navigation system for electric wheelchairs," in *Proc. Int. Conf. Image Process. Robot. (ICIP)*, Mar. 2020, pp. 1–6.

[7] V. Engesser, E. Rombaut, L. Vanhaverbeke, and P. Lebeau, "Autonomous delivery solutions for last-mile logistics operations: A literature review and research agenda," *Sustainability*, vol. 15, no. 3, p. 2774, Feb. 2023.

[8] I. Diddeniya, I. Wanniarachchi, H. Gunasinghe, C. Premachandra, and H. Kawanaka, "Human–robot communication system for an isolated environment," *IEEE Access*, vol. 10, pp. 63258–63269, 2022.

[9] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Ind. Robot: Int. J.*, vol. 43, no. 6, pp. 583–587, Oct. 2016.

[10] M. Dlugosz, P. Wegrzyn, and M. Roman, "Autonomous delivery robot AQUILO," in *Advances in Intelligent Systems and Computing*. Springer, 2020, pp. 1213–1224.

[11] J. Choi, M. Kim, J. Kim, and W. Lee, "Designing an interactive indoor delivery robot and its implications," in *Proc. Int. Conf. Robot Intell. Tech. Appl.* Springer, 2021, pp. 201–209.

[12] J. Lee, G. Park, I. Cho, K. Kang, D. Pyo, S. Cho, M. Cho, and W. Chung, "ODS-bot: Mobile robot navigation for outdoor delivery services," *IEEE Access*, vol. 10, pp. 107250–107258, 2022.

[13] M. A. K. Niloy, A. Shama, R. K. Chakrabortty, M. J. Ryan, F. R. Badal, Z. Tasneem, M. H. Ahamed, S. I. Moyeen, S. K. Das, M. F. Ali, M. R. Islam, and D. K. Saha, "Critical design and control issues of indoor autonomous mobile robots: A review," *IEEE Access*, vol. 9, pp. 35338–35370, 2021.

[14] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[15] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

[16] J. Kim and H. Jung, "Robot routing problem of last-mile delivery in indoor environments," *Appl. Sci.*, vol. 12, no. 18, p. 9111, Sep. 2022.

[17] E.-H. Kim, S.-H. Bae, and T.-Y. Kuc, "Mobile service robot multi-floor navigation using visual detection and recognition of elevator features(ICCAS 2020)," in *Proc. 20th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2020, pp. 982–985.

[18] Z. Fan, C. Li, Y. Wang, L. Zhao, L. Yao, G. Zhu, Z. Li, H. Xie, and Y. Xiao, "3D mapping of multi-floor buildings based on sensor fusion," in *Proc. Int. Conf. Ind. Informat. Comput. Technol., Intell. Technol., Ind. Inf. Integr. (ICIICII)*, Dec. 2017, pp. 10–15.

[19] A. G. Ozkil, Z. Fan, J. Xiao, S. Dawids, J. K. Kristensen, and K. H. Christensen, "Mapping of multi-floor buildings: A barometric approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Sep. 2011, pp. 847–852.

[20] A. A. Abdulla, H. Liu, N. Stoll, and K. Thurow, "A new robust method for mobile robot multifloor navigation in distributed life science laboratories," *J. Control Sci. Eng.*, vol. 2016, pp. 1–17, 2016.

[21] S.-Y. Huang, H.-Y. Huang, H. Y. Chong, J.-B. Jiang, J.-S. Leu, and S. Vítek, "A directional particle filter-based multi-floor indoor positioning system," *IEEE Access*, vol. 10, pp. 116317–116325, 2022.

[22] J. Luo, C. Zhang, and C. Wang, "Indoor multi-floor 3D target tracking based on the multi-sensor fusion," *IEEE Access*, vol. 8, pp. 36836–36846, 2020.

[23] Z. Hao, J. Dang, W. Cai, and Y. Duan, "A multi-floor location method based on multi-sensor and WiFi fingerprint fusion," *IEEE Access*, vol. 8, pp. 223765–223781, 2020.

[24] R. Santos, R. Leonardo, M. Barandas, D. Moreira, T. Rocha, P. Alves, J. P. Oliveira, and H. Gamboa, "Crowdsourcing-based fingerprinting for indoor location in multi-storey buildings," *IEEE Access*, vol. 9, pp. 31143–31160, 2021.

[25] G. Zhou, S. Xu, S. Zhang, Y. Wang, and C. Xiang, "Multi-floor indoor localization based on multi-modal sensors," *Sensors*, vol. 22, no. 11, p. 4162, May 2022.

[26] Y.-C. Lee and J. Kim, "Multi-floor localization method for mobile robots using elevator," in *Proc. Int. Conf. Control, Automat. Syst.*, 2016, pp. 869–872.

[27] B. R. K. Mantha, M. K. Jung, B. G. de Soto, C. C. Menassa, and V. R. Kamat, "Generalized task allocation and route planning for robots with multiple depots in indoor building environments," *Autom. Construct.*, vol. 119, Nov. 2020, Art. no. 103359.

[28] J. Palacín, E. Rubies, R. Bitriá, and E. Clotet, "Path planning of a mobile delivery robot operating in a multi-story building based on a predefined navigation tree," *Sensors*, vol. 23, no. 21, p. 8795, Oct. 2023.

[29] J. Zhao, Y. Chen, and Y. Lou, "A human-aware robotic system for mobile robot navigating in multi-floor building with elevator," in *Proc. WRC Symp. Advan. Robot. Automat.*, 2019, pp. 178–183.

[30] S. Jiang, W. Yao, M.-S. Wong, M. Hang, Z. Hong, E.-J. Kim, S.-H. Joo, and T.-Y. Kuc, "Automatic elevator button localization using a combined detecting and tracking framework for multi-story navigation," *IEEE Access*, vol. 8, pp. 1118–1134, 2020.

[31] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Sep. 2019, pp. 9157–9166.

[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA*, vol. 3, Kobe, Japan, 2009, p. 5.

[33] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 4758–4765.

[34] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot., Sci. Syst.*, Berkeley, CA, USA, Jul. 2014, pp. 1–9.

[35] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2003, pp. 2743–2748.

[36] C. Ulas and H. Temeltas, "3D multi-layered normal distribution transform for fast and long range scan matching," *J. Intell. Robotic Syst.*, vol. 71, no. 1, pp. 85–108, Jul. 2013.

[37] G. Kang, D. Lee, and H. Shim, "3D multi-floor precision mapping and localization for indoor autonomous robots," *J. Korea Robot. Soc.*, vol. 17, no. 1, pp. 25–31, Mar. 2022.

[38] D. Lee, G. Kang, T. Kim, D. H. Shim, H. Jung, and E. Kim, "Route planning and elevator boarding algorithms for last mile delivery service in multi-floor environments," *J. Korea Robot. Soc.*, vol. 18, no. 1, pp. 10–17, Feb. 2023.

[39] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

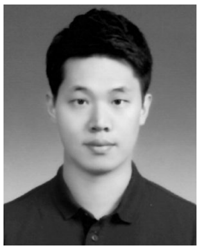
[40] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot. Sci., Syst.*, vol. 2, no. 4, p. 435, Jun. 2009.

[41] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 11054–11059.

[42] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5681–5686.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[44] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ IROS*, Oct. 2016, pp. 4193–4198.



TAEJIN KIM (Student Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in robotics program from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011 and 2014, respectively, where he is currently pursuing the Ph.D. degree in robotics program.

He was with Korea Research Institute of Ships and Ocean Engineering (KRISO), from 2014 to 2017. His research interests include autonomous systems, robotics, SLAM, and localization based on unmanned ground vehicles.

GYUREE KANG (Student Member, IEEE) received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon-si, Gyeonggi-do, South Korea, in 2020, and the M.S. degree in robotics program from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2022, where she is currently pursuing the Ph.D. degree in electrical engineering.

Her research interests include autonomous systems, robotics, and motion planning based on unmanned ground vehicles.

DAEGYU LEE (Student Member, IEEE) received the B.S. degree in automotive engineering from Kookmin University, Seoul, South Korea, in 2018, and the M.S. and Ph.D. degrees from the Division of Future Vehicle and Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020 and 2024, respectively.

He is currently a Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon. His research interests include autonomous systems and localization, focusing on robust and learning-based approaches.

D. HYUNCHUL SHIM (Member, IEEE) received the B.S. and M.S. degrees in mechanical design and production engineering from Seoul National University, Seoul, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2000.

He was with Hyundai Motor Company and Maxtor Corporation, from 1993 to 1994 and from 2001 to 2005, respectively. In 2007, he joined the Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, where he is currently a tenured Professor with the Department of Electrical Engineering, and an Adjunct Professor with the Graduate School of AI. He is also the Director of the Korea Civil RPAS Research Center. His research interests include control systems, autonomous vehicles, and robotics.