Autonomous Exploration in a Cluttered Environment for a Mobile Robot with 2D-Map Segmentation and Object Detection

Hyungseok Kim, Hyeongjin Kim, Seonil Lee and Hyeonbeom Lee, *Member, IEEE,*

***Abstract*— Frontier-based exploration is widely adopted for exploring an unknown region. The conventional frontier-based exploration for a mobile robot may collide with three-dimensional (3D) obstacles or can suffer from a slower exploration time because the robot may move to another place before completely exploring the current area. To solve this problem, in this paper, we propose a new exploration algorithm by considering a path traveled by a mobile robot and segmenting a two-dimensional (2D) map. The segmented 2D map is generated in real-time by using the position of the robot and the location of the detected frontiers. To apply our algorithm to the actual experiment, we develop an object detection-based exploration algorithm that can remarkably reduce the probability of collision with 3D obstacles. To verify the effectiveness of our proposed algorithm, we perform simulations (Gazebo) and experiments (in the real world) to compare the conventional approach and our algorithm in a cluttered environment. The simulation and experiment results show that our algorithm can satisfactorily shorten the exploration path and time.**

I. INTRODUCTION

Autonomous exploration is one of the representative methods for exploring unknown areas. This method can be used in various scenarios such as disaster [1], rescue [2,3], planetary exploration [4], environmental monitoring [5] and infrastructure inspection [6,7]. Autonomous exploration explores unknown areas faster and more efficiently by selecting the most meaningful point as the next goal point. However, a conventional exploration algorithm for a 2D mobile robot [7]–[12] is developed in simulations or experiments without any structural obstacles such as chair, and table, so a careful approach must apply these algorithms to actual home or office environment.

To achieve a collision-free exploration in the cluttered environments unlike the conventional explorations [7]–[12], we focus on two aspects. 1) Developing a next target selection algorithm to improve the efficiency of the exploration algorithm, 2) handling the three-dimensional (3D) size of obstacles using an object-based costmap to improve the safety of a mobile robot. These algorithms are the focus of this paper.

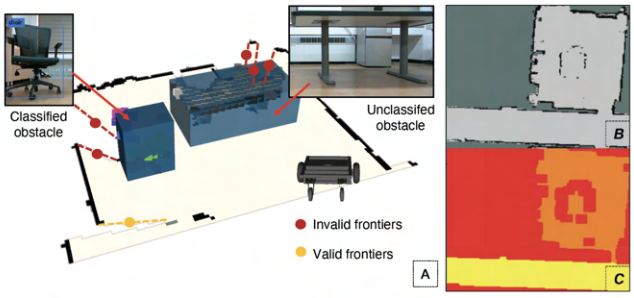


Fig. 1. Considering the size of the obstacles and the robot based on the object detection algorithm, a mobile robot automatically determines invalid frontiers for a safe exploration. (A) object detection image. (B) occupancy grid map, and (C) segmentation map.

A. Contribution

The contributions of this paper can be summarized as follows. First, we propose an integrated framework for autonomous exploration by exploiting the real-time (2D) map segmentation and geometric information between frontiers (i.e., the boundary between unknown and known areas) and mobile robot (Fig.1). Unlike the conventional exploration approach in [8]–[10], 2D map segmentation minimizes the chances of the robot moving to other areas (e.g., room, corridor, etc.) before completely exploring a given region. Second, we find feasible exploration areas by detecting obstacles and calculating the 3D size of the obstacles only using RGB-D camera. Since autonomous exploration with the 3D Light Detection and Ranging (LiDAR) sensor [12] requires an expensive sensor and a high computational payload, we have developed a real-time object detection and exploration algorithm using an RGB-D camera and a 2D laser scanner. Unlike the conventional explorations using RGB-D camera, tested only in a simulation [7] or an experiment without the structured obstacles [11], we perform the experiment in the cluttered environments. For this work, we generate an object-based costmap by developing the object detection algorithm for both the classified and unclassified objects to safely conduct autonomous exploration. Although the table was not recognized due to the location of the camera mounted on the mobile robot (see Fig.1), our algorithm can recognize obstacles well and find a safe exploration point. Finally, to verify the performance of our proposed algorithm, we conduct experiments in simulated and real-world environments

Manuscript received: December 29, 2021; Revised: March 23, 2022; Accepted: April 14, 2022. This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This research was supported in part by the Unmanned Vehicle Advanced Research Center (UVARC) by the Ministry of Science and ICT, Republic of Korea, under Grant NRF-2020M3C1C1A01086411 and the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2021R1A6A1A03043144). *(Hyungseok Kim and Hyeongjin Kim contributed equally to this work. Corresponding author: Hyeonbeom Lee)*

Hyungseok Kim, HyeongJin Kim, Seonil Lee and Hyeonbeom Lee are with the School of Electronic and Electrical Engineering, Kyungpook National University, 80, Daehak-ro, Buk-gu, Daegu, Republic of Korea. (email: hbeomlee@knu.ac.kr)

[1]. Experimental results [2] show the feasibility of our proposed algorithm in enabling exploring autonomous exploration in unknown regions.

B. Related Works

Yamauchi et al. [13] developed earlier works on frontier-based autonomous exploration, which is the boundary between unknown and known areas. Their method is fast and simple to implement. However, since their method does not use obstacle information, it may increase the exploration time by targeting unreachable points or moving in a zigzag pattern.

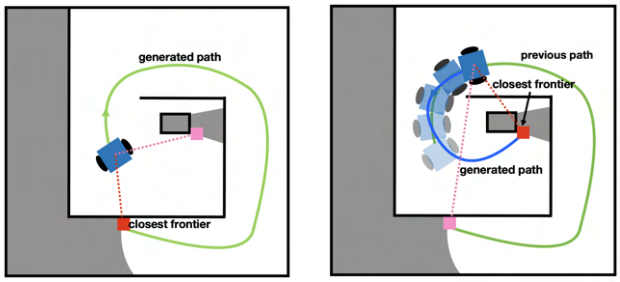
To overcome this issue, many researchers have proposed algorithms to select the next destination using information gain [8]–[10]. Pimentel et al. [8] and Strom et al. [9] proposed an information-gain-estimation algorithm by predicting the relevant areas. In [8], they used a heuristic wall segmentation method. In [9], they calculated the information gain of an unexplored area by comparing a map in a database and an acquired map of a robot. Umari et al. [10] computed the cost of detected frontiers using rapid random trees. In comparison with earlier works [13], these algorithms could efficiently explore unknown areas using information gain. However, in complex environments, robots employing these algorithms tend to move to other areas before completely exploring a specific region, e.g., a room or corridor.

Deep learning (DL) algorithms have been used to solve the autonomous exploration problem by predicting unknown regions and optimizing sensing action. To predict unknown regions, Saroya et al. [14] presented a world prediction method based on a convolutional neural network by extracting topological features, such as loops and dead-ends, to adaptively select navigation goals. Shrestha et al. [15] proposed a DL-based algorithm to autocomplete the geometric information of an unseen map. To optimize the sensing action, Bai et al. [16] proposed a method to predict the most informative sensing action using a deep neural network. Chen et al. [17] predicted a robot's optimal sensing action in belief space using graph neural networks, along with deep reinforcement learning, enabling decision-making over graphs containing exploration information. These DL-based algorithms are more efficient in complex environments than the previously mentioned algorithms [8]–[10]; however, since they focus on predicting the map using DL, they may become unstable in environments with structural obstacles and take a long time to learn the maps.

In autonomous exploration with structural obstacles, obstacle avoidance is one of the critical technologies that guarantees vehicle safety. Since the 2D-LiDAR-based obstacle avoidance [18]–[20] cannot determine the exact 3D size of an obstacle, a robot-obstacle collision may occur in a cluttered environment. To solve this problem, a 3D map obtained from 3D LiDAR [12] or RGB-D camera [7] was exploited for autonomous exploration. Although these methods [7,12] can generate a safer path using obstacle information from the 3D map, 3D LiDAR is more expensive than 2D LiDAR and depth measurement from the RGB-D camera is noisy to build an accurate 2D map. Additionally, if the aforementioned obstacle avoidance algorithm is applied to 3D space, the amount of data to be processed increases exponentially. Wang et al. [11] proposed the exploration algorithm with a 3D map using an RGB-D sensor and a 2D laser scanner. They found a traversable map using a 2D projected map and an RGB-D camera; however, the accuracy of the projected map may be degraded due to repeated movement of the robot, making it difficult to find the frontier.

[1]All code and data used in the experiments are available at https://github.com/KimHyung/autonomous_exploration

[2]Video: https://youtu.be/2d7KO31L7ac



(a) (b)

Fig. 2. Conventional frontier exploration explores by selecting the closest frontier. (a) Red Frontiers for the next target (b) A mobile robot changes the next target again since the closest frontier is changed.

II. SYSTEM OVERVIEW

Fig.2 describes typical problems of the conventional frontier-based exploration algorithms [3]. These algorithms use the linear distance between the frontier and mobile robot when selecting the next target point. Furthermore, they have the advantage of being fast and simple in calculating, but they increase the exploration time because of a rotation motion of the robot or re-exploring the remaining part of the explored area as illustrated in Fig.2(a). While the robot explores new areas such as rooms and corridors, it often moves to other areas before completely exploring the specific region.

Fig.3 shows the integrated framework of our proposed autonomous exploration method. First, we detect all the frontier points ($frontiers$) based on the explored map, and then we classify them into clusters ($frontier\_groups$) using the 8-neighbor search algorithm. Simultaneously, it computes the center point and size representing $frontier\_groups$. Then, to obtain a 2D segmented map, we use an incremental dual-space decomposition (DuDe) [21] that uses a contour based part segmentation [22] and compute the information gain for each $frontier\_groups$. Place segmentation can be achieved by 3D scene graphs [23,24], but in this paper we use the dual-space decomposition algorithm based on a 2D map image to reduce the computational payload. Finally, we select the minimum cost frontier as the next target point using the cost function.

For the 3D obstacle avoidance during exploration of an unknown area, the robot recognizes 3D objects using YOLO

[3]http://wiki.ros.org/frontier_exploration

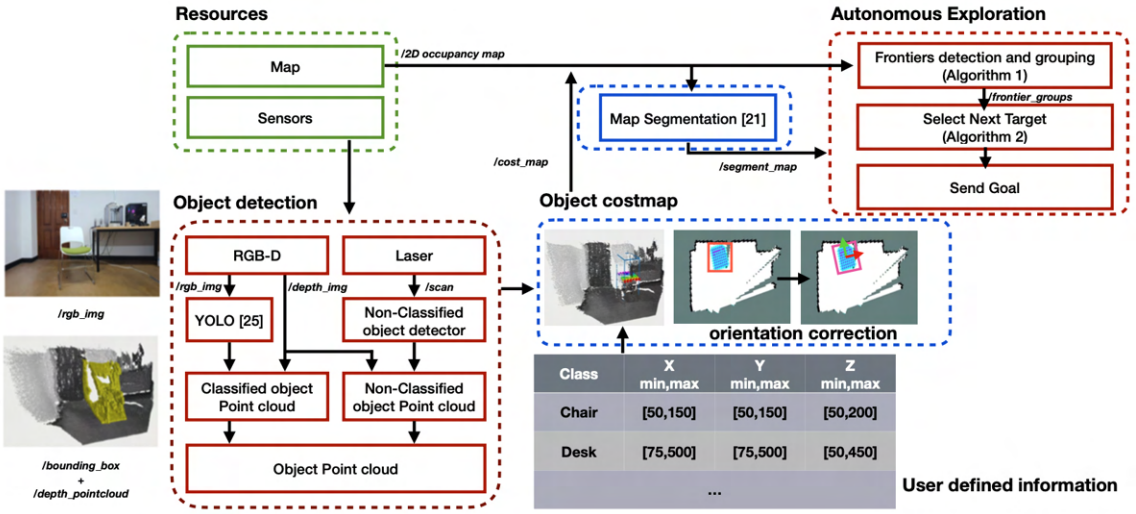


Fig. 3. Autonomous exploration and object-based costmap system diagram

Fig. 4. Detect and Cluster frontiers, frontiers(red), clustered frontiers(yellow), and center point of clustered frontiers(blue)

(you look only once) [25] for the classified object or 2D-LiDAR-based object detection for the unclassified object. The location and approximate dimension of the detected object are initially estimated based on the 2D bounding box in the color image given by YOLO. The point cloud of the corresponding area is extracted based on the extracted information about the location and size. Thereafter, the point cloud in the bounding box is preprocessed through the SOR (statistical outlier removal) using PCL (point cloud library) and Euclidean distance cluster filtering, and a three-dimensional bounding box is created using min and max values in the x, y, and z direction of the point cloud. For the bounding box, we modify and confirm the size of the detected object by exploiting roughly-known geometric size of the object. Finally, we create a virtual point cloud based on the size of the box and apply the point cloud to the global costmap of navigation.

III. METHODS

A. Detect and Cluster Frontier

Frontiers detection and grouping is described in Algorithm 1. The function **detectFrontiers** (Alg. 1, line 3) detects the frontiers from the 2D occupancy grid map M_o which was built by the 2D SLAM (simultaneous localization and mapping) algorithm. In M_o, an occupied cell and a free cell have a value of one and zero respectively, and an unknown cell that has not yet been detected by a sensor has a value of 0.5. Using these values, the **detectFrontiers** function checks whether each cell in M_o is unknown, and it finds the frontiers by searching all eight directions of the cell. Simultaneously, this function builds a frontier map q_f which is the same size as M_o. $q_f.visit$ is used to check if the frontier was visited or not.

After finding frontiers, we must group them for efficient computation. To do so, we find neighbor frontiers using BFS (Breadth-First Search). The **neighbor-cell** function (Alg. 1, line 8) finds all neighbor cells and checks whether the neighbor cell is frontier and unvisited cell. Then, the BFS algorithm finds all connected frontiers associated with the frontier $q_f(k)$ and saves the associated cells as q_n and the indices of the q_n as Nid. If the neighbor cell is a frontier, the algorithm changes the variable from unvisited to visited (Alg. 1, line 10-11). Then, the frontier is re-defined as $groups$ (line 9). Each group structure has three variables as follows: id, $cSize$ and $cPoint$. The variable id is the $group$'s unique id; the $group$ are accessed through this id. The variable $cSize$ is the size number of frontiers in each $group$ (**clusterSize** function, Alg. 1 line 12) and this variable is used to calculate the size gain. The variable $cPoint$ is the representative point of each $group$ (**centerPoint** function, Alg. 1 line 13). When a group is clustered, the algorithm increases the gId (Alg. 1, line 14). Finally, if all frontiers are visited, the algorithm stops. The result of the experiment using the $detected\_and\_cluster$ module is shown in Fig.4.

B. Select Next Target

After finding the frontier groups, we select the next goal point among these frontier groups. Algorithm 2 is designed to calculate the cost of each $group$ and selects $cPoint$ with the minimum cost as the next target point. The cost function consists of $sizeGain\{i\}$, $distGain\{i\}$ and $regionGain\{i\}$

Algorithm 1: Frontiers detection and grouping

Input: $M_o \leftarrow$ Grid map
Output: groups{id}.{frontiers, cSize, cPoint}
q_f = **detectFrontiers**(M_o)
$q_f.visit = \emptyset$
id=0
for *k=0,1,...,q_f.size()* **do**
  if $q_f(k).visit == unvisited\ frontier$ **then**
    // Performs 8 − directions neighbor search
    [q_n, Nid] = **neighbor-cell**($q_f(k)$)
    groups{id}.frontiers $\leftarrow (q_f(k), q_n)$
    $q_f(k).visit = visited\ frontier$
    $q_f(Nid).visit = visited\ frontier$
    // Calulate the group size
    group{id}.cSize = **clusterSize**($q_f(k)$, q_n)
    // Calulate the center point for each group
    group{id}.cPoint = **centerPoint**($q_f(k)$, q_n)
    id++
  end
end

Algorithm 2: Select Next Target

Input: pose $\leftarrow$ position of the robot
  $M_s \leftarrow$ segmented map (ref. [21])
  groups $\leftarrow$ result of Algorithm 1
Output: goal
$minCost$=∞
for $k = 0, 1, \ldots,$ $groups.size()$ **do**
  $dist\{k\} \leftarrow$ **path**(pose,groups{k}.cPoint)
end
for $k = 0, 1, \ldots,$ $groups.size()$ **do**
  $distGain\{k\} \leftarrow$ eq.(1), $sizeGain\{k\} \leftarrow$ eq.(2),
  // If frontier and robot are in the same area
  if *groups{k}.cPoint* $\in$ *M_s(pose)* **then**
    $regionGain = 0$
  else
    $regionGain = \gamma$
  end
  $cost\{k\} \leftarrow$ eq.(3)
  if $minCost > cost\{k\}$ **then**
    $minCost = cost$
    $goal = group\{k\}.cPoint$
  end
end

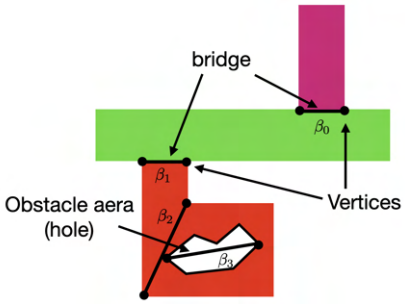


Fig. 5. 2D map segmentation using DUDE [21,22]. Bridge β_3 is the antipodal vertices of the hole i.e., two farthest apart vertices of the hole. Bridge β_2 is the child of β_1. The polygonal P segmented by β_0 and β_1.

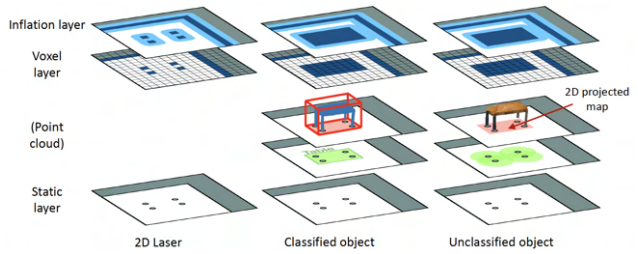


Fig. 6. The costmap updates for the classified and unclassified objects

for the i-th frontier group as

$$distGain\{i\} = \frac{dist\{i\}}{\sum_{j=1}^{N} dist\{j\}} \quad (1)$$

$$sizeGain\{i\} = \frac{groups\{i\}.size}{q_f.size()}, \quad (2)$$

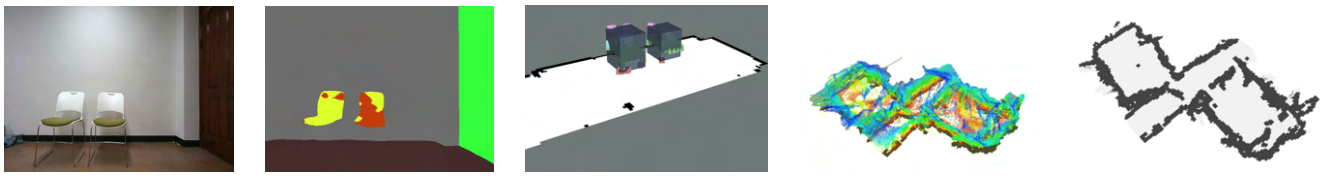
where $dist\{i\}$ is a path length from the current robot location to the $cPoint$ of i-th $group$ (**path** function, Alg. 2 line 7). If $group\{i\}.cPoint$ is in the obstacle area, we set $dist\{i\}$ as infinity. $q_f.size()$ is the total number of frontiers. To make the unit between $sizeGain$ and $distGain$ similar, $dist\{i\}$ is divided by the total summation of $dist\{i\}$. In this case, a larger value of $sizeGain$ means that there are more unknown regions around the location. To minimize the cost, $1/sizeGain$ is considered for the cost. So, the cost of the i-th $group$ (i.e., $cost\{i\}$) can be calculated as

$$cost\{i\} = \alpha \times distGain\{i\} + \frac{\beta}{sizeGain\{i\}} + \gamma \quad (3)$$

where α and β are the user defined gain. γ is the user defined value but it can be changed based on the location of the robot(Alg. 2 line 11).

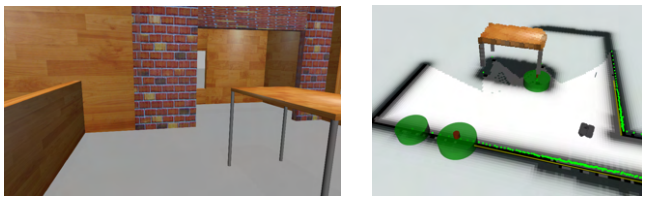
The $regionGain$ (Alg. 2 line 11-15) is the cost that determines whether the current robot and $cPoint$ of each $group$ are in the same area. Mentioned earlier, the robot often moves to other areas before completely exploring the specific region. To solve this problem, incremental dual-space decomposition (DuDe) segmentation [21] based on [22], was used to determine whether the robot and the frontier are in the same space.

The segmentation method transforms a 2D occupancy grid into polygons and segments the space by creating the nearly convex decomposition of the shapes by segmenting the regions of the shape (see Fig.5). For DuDe, the input is a polygon P, which possibly has holes in it. The polygon and the holes are represented as a sequence of 2D points whose ends are assumed to be connected. The binary image of the 2D occupancy grid is transformed into a polygon representation with holes, including an obstacle



(a) (b) (c) (d) (e)

Fig. 7. The comparison of classic costmap and object based costmap, (a) Input image, (b) Semantic segmentation [26], (c) Our 2D map, (d) Vision-based 3D mapping (Fiesta mapping [27] with T-265 sensor), (e) Projected 2D map of (d).



(a) (b)

Fig. 8. The costmap of the undetected object, (a) non-detected object, and (b) costmap by (a).

area, using *OpenCV* function $findContours$. Each hole is bridged based on connectivity to each other, and then the regions are decomposed by the holes and the bridge β_i. In order to reduce the computational payload of segmentation algorithm, our proposed method compares the differences of the incremental map. More details can be found in [21]. As illustrated in Fig.1, the map is classified into different color spaces.

To exploit the color information of the map, the segmented map and the 2D occupied grid map are synchronized. Using the color information, the $cPoint$ of each $group$ and the location of the robot are clustered (Alg. 2 line 12). If the two locations are in the same space, $regionGain$ has the value of zero, otherwise, it is γ.

C. Object-based Costmap

For safe navigation, a mobile robot should plan the desired trajectory and avoid obstacles using a costmap. The costmap comprises a static map, inflation, and voxel layers to express the free space for navigation. Inspired by this, we have created an object-based costmap by estimating the location and 3D size of obstacles in the cluttered environments (See Fig.6). Here, the inflation layer consists of a process of propagating the cost values out from the occupied cells, later used for path planning considering the robot size.

Floor segmentation [28] or semantic segmentation [26,29] was also used for the safe indoor or outdoor navigation for a mobile robot. However, vision-based methods [26,28] may fail to detect the thin legs of a chair made of special materials such as aluminum as shown in Fig.7 (b). In contrast, by the proposed method, the point cloud of the desks was generated precisely in the costmap as shown in Fig.7 (c). 2D projected map generated by the 3D mapping algorithm [11,27] can solve the object detection problem. However, if an accumulated odometry error occurs due to repeated movement, the accuracy of the 2D map may be degraded, which may make it difficult to find the frontier as shown in Fig.7 (d,e).

To overcome the aforementioned problem, we use a 2D laser-scanner-based slam algorithm and an object detection algorithm YOLO. YOLO and pre-trained v2-tiny weight were used as the object detection method, and a stereo camera was used to calculate the localization and 3D size of a detected object. YOLO detects a 3D object in the cluttered environment and finds a 2D bounding box in the color image. Then, to reduce the noisy measurement from the depth image, SOR filter and Euclidean clustering were added as pre-processing methods. Then, a 3D bounding box is generated by calculating the min and max values of the point cloud in the x, y, and z directions. However, the filtered point cloud is noisy while the mobile robot is moving. Therefore, it produces an inaccurate 3D bounding box. To resolve these errors, we used the class of the object information obtained with YOLO to predefine the maximum and minimum values of the three-dimensional size of each object (see the table in Fig.3). Initially, we assume that the objects are aligned on the 2D map, and the initial size of the 3D bounding box is determined. After point clouds are projected on the 2D map, the box size is resized by estimating the object orientation on a 2D map. To estimate the orientation of the objects, we use principal component analysis (PCA) in *OpenCV* (see Fig.3). Using this box information of the classified object, the voxel layer of the costmap in a 2D grid map is filled with obstacle regions, as shown in Fig.6, and we set the obstacle grid cell to the maximum value (i.e., one in the occupancy grid cell).

There is the unclassified object as shown in Fig.8 (a). The object is unclassified due to some reasons such as the location of the camera, and quality of the image. To handle this object, we first investigate the scanned laser point, because a place that appears in the form of a single dot such as the legs of a table or a chair on the 2D map is a dangerous area. For example, in the environment shown in Fig.6, the 2D laser sensor recognizes only the desk leg as an obstacle. To handle this issue, we find the object candidate points by clustering the 2D laser points. The green circle in Fig.8 (b) means the obstacle candidate. Next, we investigate the 3D point cloud near these possible obstacles, which is then projected on the voxel layer of the costmap in the 2D grid map, as shown in Fig.6. By generating the projection map for the corresponding obstacle region, the degradation of 2D map accuracy caused by the odometry error in [11] can be resolved. Note that the algorithm for the unclassified object takes a longer time (around 100 ms in Jetson NX

TABLE I
AVERAGE COMPUTATION TIME OF OUR PROPOSED METHODS IN JETSON NX MOUDLE

scene	detect	cluster & select	Yolo detection	costmap* update	total
house	15.5	3.7	35.3	100	154.5 [ms]
maze	15.9	3.98	35.7	105	160.58 [ms]

*This module includes the algorithm for handling an unclassified object.

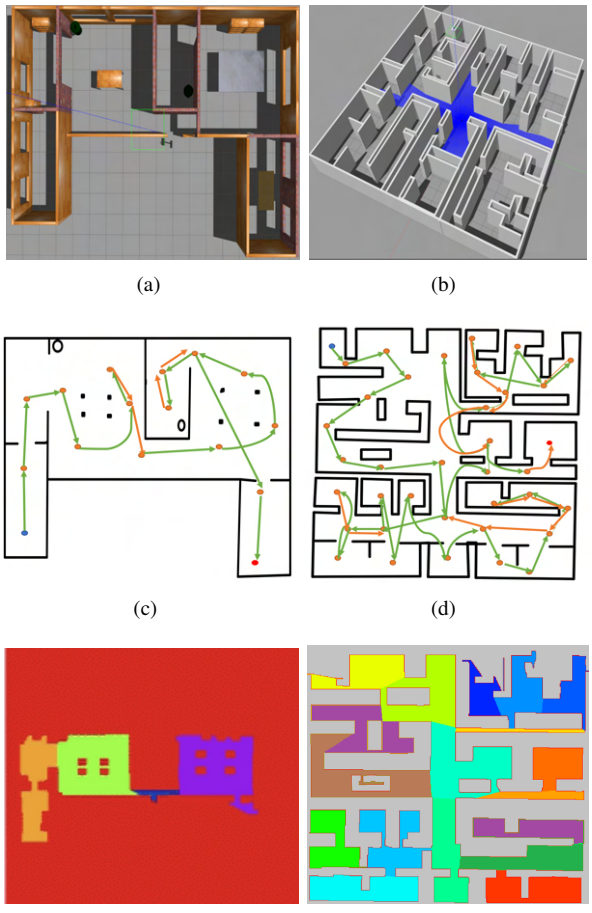


Fig. 9. Autonomous exploration in gazebo environments (a) House environment1(20m x 20m), (b) Maze environment2(40m x 40m), (c) Travelled path of (a), (d) Travelled path of (b), (e) segmented map of (a), (f) segmented map of (b).

module) than YOLO-based method (around 30 ms in Jetson NX module) because they should investigate not the specific area but the whole laser-scanned obstacle-candidate points.

IV. EXPERIMENTS

To verify the performance of our proposed algorithm, its exploration time was compared with those of other algorithms [10,13,30,31] in both virtual and real environments.

A. Autonomous Exploration in the Simulation

Gazebo, a virtual environment simulator, is a 3D robot simulator that supports plugins of various sensor modules.

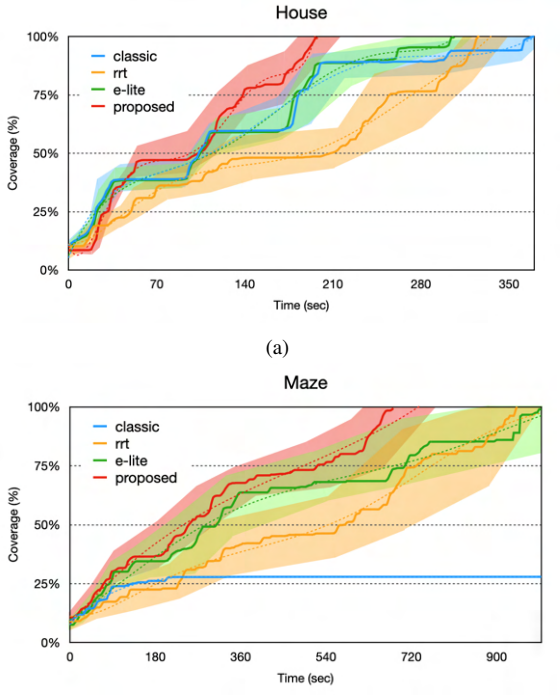
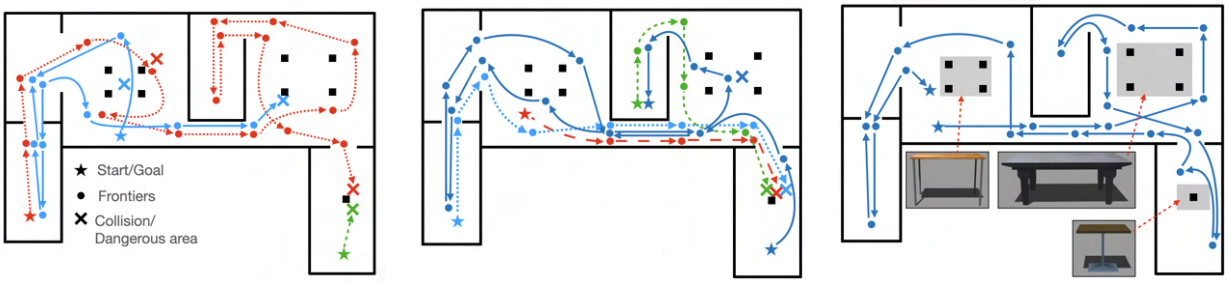


Fig. 10. Percentage map coverage against time for two floor plans. The solid line means the actual coverage and the dotted line means the interpolated coverage. (a) House (b) Maze. Results are averaged over 20 simulations.

Turtlebot3_waffle was used as a mobile robot, and an LDS-01 scanner was mounted for ray-casting measurements. The field of view (FOV) and range of the LDS-01 were limited to 360° and 5 m, respectively. The sensor update rate was 10 Hz in simulation time. The environment map is a 2D occupancy grid with a resolution of 0.05 m per pixel. Map updates and localization were performed using the GMapping ROS package. Trajectories were generated by the Navigation ROS package (based on the A* algorithm) and the move_base ROS package was used to follow the trajectories. The proposed algorithms run on a laptop computer with an Intel Core i7-9750H CPU and a single-board computer with Jetson Xavier NX.

To compare the exploration time, 20 m x 20 m house and 40 m x 40 m maze virtual maps were explored. Fig. 9 illustrated that the proposed method performed autonomous exploration well compared with the conventional methods [10,13,30,31]. Fig. 10 shows the percentage of total map coverage during the explorations. In fig. 10, the solid line of each algorithm means the actual coverage, and the dotted line means the interpolated coverage using polynomial functions. We compare our proposed method with the classic frontier-based exploration [13], rrt exploration [10], and e-lite [30] methods. We simulated each method 20 times on the gazebo house and maze world (Fig.9 (a), (b)) and averaged the exploration time. In the map with a maze environment, the graph of Fig. 10 (b) shows that the classic frontier-based exploration could not explore the map, whereas the



(a) (b) (c)

Fig. 11. Case analysis for the collision , (a) Exploration by [13] and [10] , (b) Our exploration without the object detection, (c) Our exploration with the object detection.

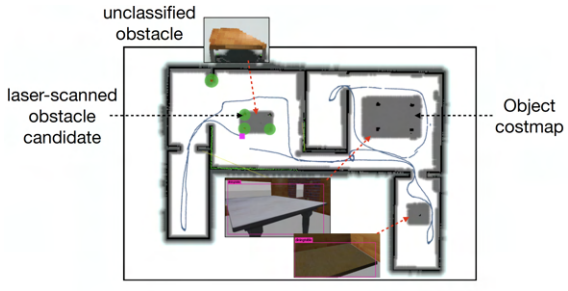
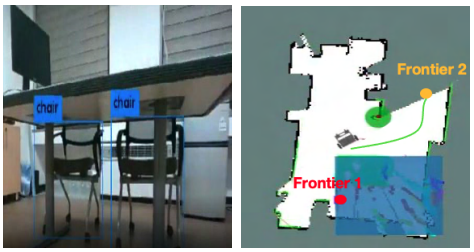


Fig. 12. Frontier exploration result with the object-based costmap



(a) (b)

Fig. 13. Autonomous exploration with object-based costmap, (a) Input image (b) Frontier not chosen as next destination due to object-based costmap(red), goal point(yellow), path(green).

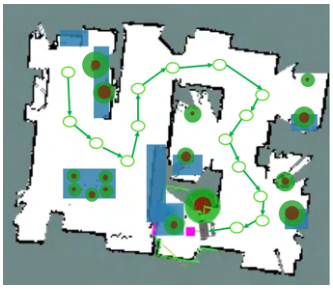


Fig. 14. Exploration result in an office environment.

other methods could. In addition, the proposed methods significantly reduce the exploration time compared to the explore-lite and rrt-exploration methods, when the average coverage is greater than 40%.

The computation time of our proposed method is listed in Table.I. According to the table, our proposed algorithm has a good run time, even on a single-board computer, such as the Jetson Xavier NX. Additionally, through experiments, it was confirmed that the autonomous exploration is executed in real time even while object detection is being executed(fps:15).

Since the simulation in Fig. 9 does not contain the cluttered objects, we perform the simulation again for the realistic house gazebo environment. The simulation results are described in Fig. 11. The existing method [10,13] tended to enter a dangerous area such as under a desk or collide with an obstacle as shown in Fig.11 (a). Our proposed algorithm also showed the same result when the object-based costmap was not used as shown in Fig.11 (b). However, when the proposed object-based costmap was used, the exploration was completed without collision as shown in Fig.11 (c) and Fig.12.

B. Autonomous Exploration Experiment

To verify the performance of the object-based costmap, we conducted an experiment in a cluttered environment where obstacles such as desks and chairs exist as shown in Fig. 13(a). For the simplicity of the experiment, we assume that the objects are aligned on the 2D map, which means the orientation of the object is 0 or 90 degrees. 'Frontier 1' is closest to the robot as shown in Fig.13 (b), so the frontier should be selected as the next destination. However, due to the 3D bounding box created using the proposed method, the corresponding area becomes an obstacle area. $distGain$ of the corresponding frontier becomes infinity and is excluded from destination selection, so 'Frontier 2' is selected as the next destination as shown in Fig.13(b). The autonomous exploration experiment is shown in Fig.14. The green circle means the obstacle candidate, and blue box means the size of the detect object using YOLO. Green line means the traveled path. The experimental result confirms the feasibility of the proposed algorithm for exploring a cluttered environment.

V. CONCLUSION AND FUTURE WORK

We proposed a new exploration approach for a 2D mobile robot using the segmented map. To identify and solve the problems of the classic frontier-based exploration, we improved the performance by calculating the distance from the current robot location to the i-th frontier group $group$ and the region gain. Our algorithm selected a next target point to

fully explore a complex area (such as rooms, hallway, etc.) by classifying generated maps in real-time. In addition, we identified navigation problems caused by misidentifying the size of objects when navigating with only 2D laser sensor. To solve this problem, we increased driving safety by creating a navigation cost map that takes into account the size of objects that are explored during autonomous exploration through the process of YOLO, SOR filtering, and Euclidean Clustering.

The segmented map is crucially required for the autonomous exploration proposed in this paper. However, there is a problem that space is wrongly classified when there are many dynamic obstacles in the area. In future works, to eliminate this issue, additional research is needed to resolve the correlation of classified spaces.

REFERENCES

[1] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, *et al.*, "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.

[2] S. Wirth and J. Pellenz, "Exploration transform: A stable exploring algorithm for robots in rescue environments," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*. IEEE, 2007, pp. 1–5.

[3] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Robot Soccer World Cup*. Springer, 2013, pp. 624–631.

[4] K. Otsu, A.-A. Agha-Mohammadi, and M. Paton, "Where to look? predictive perception with applications to planetary exploration," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 635–642, 2018.

[5] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 658–685, 2019.

[6] D. Klimentjew, M. Arli, and J. Zhang, "3d scene reconstruction based on a moving 2d laser range finder for service-robots," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2009, pp. 1129–1134.

[7] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems*, vol. 11, 2015, pp. 3–12.

[8] J. M. Pimentel, M. S. Alvim, M. F. Campos, and D. G. Macharet, "Information-driven rapidly-exploring random tree for efficient environment exploration," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 2, pp. 313–331, 2018.

[9] D. P. Ström, I. Bogoslavskyi, and C. Stachniss, "Robust exploration and homing for autonomous robots," *Robotics and Autonomous Systems*, vol. 90, pp. 125–135, 2017.

[10] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1396–1402.

[11] C. Wang, L. Meng, S. She, I. M. Mitchell, T. Li, F. Tung, W. Wan, M. Q.-H. Meng, and C. W. de Silva, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 109–116.

[12] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, 2019.

[13] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 1997, pp. 146–151.

[14] M. Saroya, G. Best, and G. A. Hollinger, "Online exploration of tunnel networks leveraging topological cnn-based world predictions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 6038–6045.

[15] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1197–1204.

[16] S. Bai, J. Wang, F. Chen, and B. Englot, "Information-theoretic exploration with bayesian optimization," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1816–1822.

[17] F. Chen, J. D. Martin, Y. Huang, J. Wang, and B. Englot, "Autonomous exploration under uncertainty via deep reinforcement learning on graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 6140–6147.

[18] A. Oustaloup, B. Orsoni, P. Melchior, and H. Linares, "Path planning by fractional differentiation," *Robotica*, vol. 21, no. 1, pp. 59–69, 2003.

[19] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[20] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2-4, pp. 190–216, 2010.

[21] L. Fermin-Leon, J. Neira, and J. A. Castellanos, "Incremental contour-based topological segmentation for robot exploration," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 2554–2561.

[22] G. Liu, Z. Xi, and J.-M. Lien, "Dual-space decomposition of 2d complex shapes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 4154–4161.

[23] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," in *Robotics: Science and Systems*, 2020.

[24] U.-H. Kim, J.-M. Park, T.-J. Song, and J.-H. Kim, "3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE transactions on cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2019.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[26] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal on Computer Vision*, vol. 127, p. 302–321, 2019.

[27] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4423–4430.

[28] A. Aladrén, G. López-Nicolás, L. Puig, and J. J. Guerrero, "Navigation assistance for the visually impaired using rgb-d sensor with range expansion," *IEEE Systems Journal*, vol. 10, no. 3, pp. 922–932, 2016.

[29] H. Wang, Y. Sun, and M. Liu, "Self-supervised drivable area and road anomaly segmentation using rgb-d data for robotic wheelchairs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4386–4393, 2019.

[30] Y. Alborzi, B. S. Jalal, and E. Najafi, "Ros-based slam and navigation for a gazebo-simulated autonomous quadrotor," in *2020 21st International Conference on Research and Education in Mechatronics (REM)*, 2020, pp. 1–5.

[31] D. Deng, R. Duan, J. Liu, K. Sheng, and K. Shimada, "Robotic exploration of unknown 2d environment using a frontier-based automatic-differentiable information gain measure," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2020, pp. 1497–1503.