2023 62nd Annual Conference of the Society of
Instrument and Control Engineers (SICE)
September 6-9, 2023. Tsu City, Japan

# Cooperative System of Ground Robots based on dynamic exploration of Drone Occupancy Grid Maps

Hiroaki Kobori† and Kosuke Sekiyama

Department of Mechatronics Engineering, Graduate School of Science and Engineering, Meijo University
(E-mails: sekiyama@meijo-u.ac.jp)

**Abstract:** This study proposes a real-time action planning system for autonomous mobile robots, allowing them to request assistance from other robots when necessary. The system focuses on search and uses a ground robot and a drone to search unknown areas. The drone creates an Occupancy Gird Map and transfers it to the ground robot for optimal path planning. The Behavior Trees used for autonomous robot search reactively select actions to uncertain events by monitoring the state, which is grown using a reachability graph generated in advance by offline planning. The system allows robots to automatically generate tasks and decide which robot should perform them, optimizing battery consumption and avoiding unnecessary movements.

**Keywords:** cooperative system, motion planning, Behavior Tree, Occupancy Grid Map

## 1. INTRODUCTION

In recent years, much research has been conducted on autonomous mobile robots, including the development of cooperative exploration systems using ground robots and drones. In cooperative systems, there are many meticulously designed systems in which robots work together at predetermined times specified by the user[1]. However, in such cooperative systems, the strengths of the robots may not be fully exploited due to changes in the environment. In addition, changes in the environment may cause robots to perform cooperative tasks even when cooperation is not necessary. If the robots being coordinated are drones or other devices with low battery capacity, battery consumption due to unnecessary movement should be avoided. Therefore, it is necessary for the robots to automatically generate tasks and decide when and which robot should perform which task. In addition, it is necessary to develop a system that can request assistance from other robots when a task is difficult to solve by itself in performing a task.

Therefore, this study proposes a system in which robots with different specifications can request tasks as needed using a real-time action planning system. If the task is simple, the task is performed by a single robot only. If the task is complex, the system will recognize that it is too demanding for a single robot to perform the task alone, and will develop a system that requests help from a robot that is either performing another task or is free from the task. In this study, we focused on the path planning problem in cooperative systems, which is important for developing such systems, and developed a system in which a ground robot searches for an optimal path with the assistance of a drone using Visual Support. The drone creates a map of unknown areas that the ground robot cannot obtain through occlusion and transfers it to the ground robot to search along the optimal path. We also confirmed that the drone automatically generates tasks and executes plans. The Behavior Trees used for autonomous robot path planning reactively selects actions to uncertain events, such as environmental changes, by continuously monitoring the state. The Behavior Tree is grown using a reachability graph generated in advance by offline planning.

† Hiroaki Kobori is the presenter of this paper.

## 2. RELATED WORK

### 2.1. Path planning probrem

To solve the path planning problem, it is necessary to develop a system that can autonomously solve the problem when the given environment changes or when the area is complex. As a solution for complex areas, a drone flew over the area in a previous study and the operator set waypoints based on the drone's camera information[2]. The system then moves the ground robot based on the set waypoints. However, this method does not support the case where the given environment changes. In other words, it is difficult to operate multiple robots in the same environment using waypoints.

### 2.2. Automatic action plan

For the robot to act autonomously, it must re-plan according to environmental information and the results of its actions. In addition, if it fails to act, it must recognize and correct the failure. A previous study of automatic action planning systems is the Hybrid Backward-Forward (HBF)[3]. It continuously searches from the initial state to the target state, selecting each state ' s possible actions. In the backward search, the generated Reachability Graph is used to efficiently perform a sampling of actions toward the target state. In the backward search, the generated Reachability Graph is used to efficiently sample actions toward the target state. However, depending on the complexity of the problem, processing may be time-consuming or it may be difficult to obtain all necessary environmental information. In addition, it is necessary to detect and correct for self-recognition of robot failures and inconsistencies in planning that may occur when recognition is incorrect.

978-4-9077-6480-7 PR0001/23 ¥400 © 2023 SICE

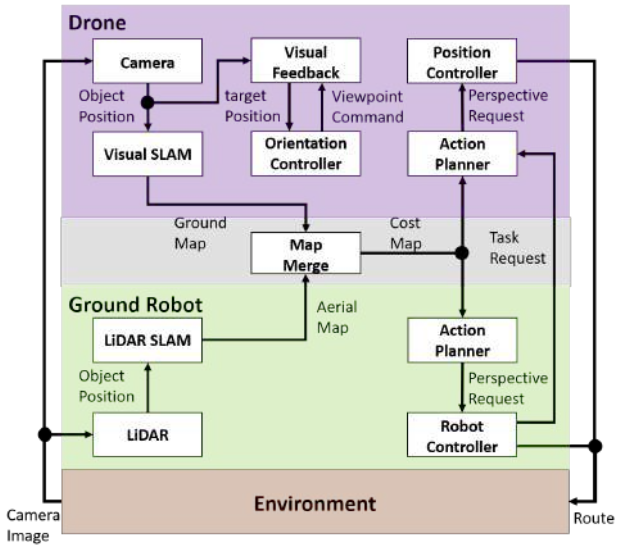


Fig. 1 System architecture of cooperative systems with multiple robots

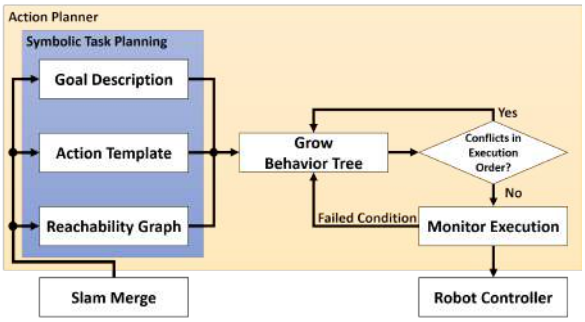


Fig. 2 System architecture of action plan

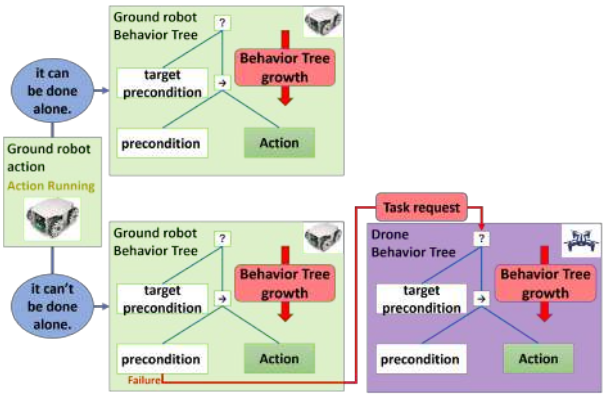


Fig. 3 Determining how to execute the next task

Table 1 Action Template

| | **Takeoff**$(q_0, q_1)$ | **Mapping**$(q_1, q_2)$ | **Move**$(q_3, q_4)$ | **Land**$(q_3, q_4)$ |
|---|---|---|---|---|
| **Precondition** | $q_0 = S_{t1}$ | $t = Mapping$<br>$q_1 = S_{t1}$ | $q_2 = S_{l0}$ | $t = None$<br>$q_3 \in P(q_4)$ |
| **Target precondition** | $q_1 = S_{t1}$ | $t = None$ | $q_3 \in P(q_4)$ | $q_4 = S_{l0}$ |

## 3. OPTIMAL ROUTE SEARCH SYSTEM

The system architecture of this system is shown in Fig.1. In this paper, Visual SLAM (Simultaneous Localization and Mapping) is used for the drone's self-position estimation. The AR markers are used for coordinate transformation to the map coordinate system. The ground robot uses LiDAR SLAM. Occupancy Grid Map created by each SLAM is combined by matching the origin of the map and the cell size. Then, based on the combined map, each robot executes its actions using the Behavior Tree.

The system architecture of the action plan is shown in Fig.2. where the Action Templates in Fig.2 represent the pre-conditions and goal constraints of the action. The Action Template is shown in Fig.1. Symbolic Task Planning generates a semi-sequential plan called Reachability Graph, which is generated by Online Planning, leaving a choice between Goal Description and Action Template. Next, the Behavior Tree is grown by generating action sequences using the Reachability Graph, and by growing the Behavior Tree, Online Action Planning is performed, and actions are commanded to the robot controller while monitoring the state. The robot controller is then commanded to perform actions while monitoring the state of the Behavior Tree. The generation of the action sequence determines scheduling, alternatives, and how the actions will be realized.

The Behavior Tree used in our system mainly utilizes a Reachability Graph, which is a reverse planning approach that assumes action failure when generating the Tree. For instance, in the case of a Behavior Tree for a single robot, if an action fails, the robot will recover within its capability. However, there are situations where a robot may not be able to recover on its own when an action fails, or multiple recovery options exist but the one that can be executed within the robot's capability may take more time. Therefore, by utilizing multiple Behavior Trees, if a node fails, the robot can access other robots' Behavior Trees and request a task. This task request leads to autonomous collaboration between robots, allowing multiple robots to complete a single task. If recovery can be performed individually, the robot will perform it without requesting a task from other robots at the failed node. Fig.3 illustrates the concept of task requests between robots. If there is a task that cannot be accomplished by a single robot when a node fails, the robot will request a task from other robots and collaborate with them.

Therefore, this system aims to develop a mechanism that allows ground robots to decide whether to collaborate with other robots for executing tasks. In this study, we focused on tasks that can be performed alone but are time-consuming. Specifically, we investigated which approach is more efficient in selecting the optimal path when there is one path to reach the goal in a given environment, but the ground robot is unable to observe the map information around the goal location. We conducted experiments to compare the performance of a single robot executing the task alone versus the same task executed with the collaboration of multiple robots.

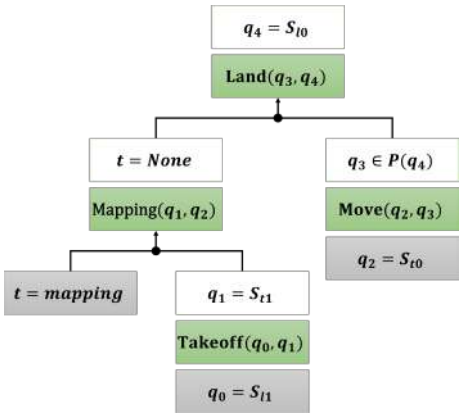


Fig. 4 Reachability Graph for Drones

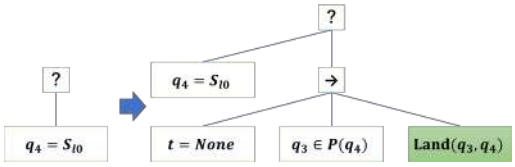


Fig. 5 Growth flow of Behavior Tree

**Algorithm 1** Reactive Execution with Behavior Tree

$T \leftarrow \emptyset$
**for** $c$ in $C_{goal}$ **do**
  $T \leftarrow SequenceNode(T, c)$
**while** $True$ **do**
  $T \leftarrow RefineAction(T)$
  **do**
    $r \leftarrow Tick(T)$
  **while** $r \neq Failure$
  $c_f \leftarrow GetFailedConditionNode(T)$
  $T \leftarrow ExpandTree(T, c_f)$

**Algorithm 2** Behavior Trees, Expand Tree

**function** $ExpandTree(T, c_f)$
  $A_T \leftarrow GetAllActTemplatesFot(c_f)$
  $T_{fall} \leftarrow c_f$
  **for** $a$ in $A_T$ **do**
    $T_{seq} \leftarrow \emptyset$
    **for** $c_a$ in $a.con$ **do**
      $T_{seq} \leftarrow sequenceNode(T_{seq}, c_a)$
    $T_{seq} \leftarrow sequenceNode(T_{seq}, c_a)$
    $T_{fall} \leftarrow FallbackNodeMemory(T_{fall}, T_{seq})$
  $T \leftarrow Substitute(T, c_f, T_{fall})$
  return $T, T_{fall}$

# 4. ACTION PLANNING SYSTEM

## 4.1. Reachability Graph

Reachability Graph(RG) shows the relationship between actions and conditions necessary to satisfy constraints. In this article, we will explain how to use the RG of drones. First, takeoff is performed from point $q_0$ to point $q_1$. $q_1$ is moved, the task is checked, and if the task is mapping, mapping is performed. $q_3$ is moved from the position $q_2$ where mapping was completed to $q_3$ where there are no obstacles on the ground. $q_3$ is then moved to land to satisfy the target constraints. In Fig.4, the white nodes are states, the green nodes are actions for transitioning states, and the gray nodes are pre-conditions for actions. To transition to the target state ($q_4 = S_{l0}$), the drone must be Landing (Land($q_3, q_4$)) with no task; to execute Land, the drone must have no task ($t = None$) and be at the landing position ($q_3 \in P(q_4)$). To move, the drone must be hovering in a task-free state. To transition to the state $t = None$, the drone should perform mapping, which requires that the task is mapping ($t = mapping$) and that the drone is hovering ($q_1 = S_{t1}$). Takeoff requires that the task is Landing ($q_0 = S_{t1}$) with the task in place. As described above, RG indicates the conditions and actions required to transition to a certain state. However, when multiple conditions need to be satisfied, the order in which the conditions are satisfied is not planned.

## 4.2. Behavior Tree

Behavior Tree(BT) is utilized to control the behavior of the robot, monitoring its state and selecting actions accordingly. The Sequence node executes the child nodes in order from left to right, failing if any child node fails and succeeding if all of them succeed. The Fallback node selects the child nodes in order from the left, and if a child node fails, the next child node is executed. If any child node succeeds, it succeeds. If any child node fails, the next child node is executed. Condition nodes always monitor the state and Action nodes perform actions. If a condition node is not satisfied during an action, the action is canceled. The action is canceled if any of the condition nodes are not satisfied during the action.

The PA-BT (Planning and Acting using Behavior Trees)[4] is used to automatically generate BT. PA-BT generates a tree by retrieving actions that satisfy given pre-conditions from an Action Template. Algorithm 1 is based on PA-BT. Algorithm 1 is the main loop of PA-BT. It searches for actions that satisfy the goal constraint $C_{goal}$ using the Action Template and generates a tree. If any of them satisfy the goal constraints, the action that satisfies the goal constraint is placed on the right side of the tree as shown in Fig.5. Then, by placing the necessary preconditions for the action on the right, a new tree is generated and the BT is grown. Algorithm 2 shows how to generate a subtree when a condition node fails. The "→" is a Sequence node and "?" is a Fallback node.

This time, when an action node of BT fails, it checks whether or not it can resolve itself, and if it cannot resolve itself, it requests other robots to cooperate with it. If there is an unknown area when generating a route, the ground robot does not know whether the generated route is passable. Therefore, by requesting a task to the BT generated by the drone, it is possible to confirm whether the unknown area is passable or not.


Fig. 6 Matching the coordinate systems between environment and drone. AR marker coordinate system (left), drone coordinate system (right)

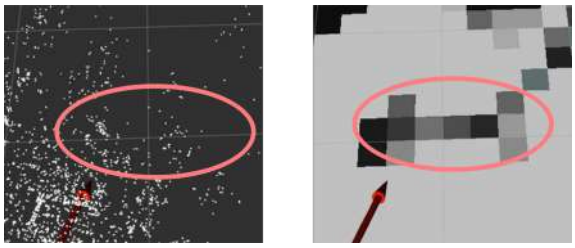

Fig. 7 : Environmental Mapping via Occupancy Grid Map Generation from Drone-Generated point cloud map. Point cloud map generated by the drone (left), Occupancy Grid Map using point cloud map (right).

## 5. CREATION OF 2D GRID MAP

### 5.1. ORB-SLAM

To estimate the drone's self-position in a non-GPS environment, we utilized ORB-SLAM(Oriented FAST and Rotated BRIEF - Simultaneous Localization and Mapping), a real-time, open-source, and CPU-efficient Visual SLAM technology that can be used with a monocular camera on a drone. ORB-SLAM extracts feature points from image data, creates a 3D map, and simultaneously estimates the drone's self-position.

However, the estimated position and orientation provided by ORB-SLAM are in a local coordinate system relative to the initialization point. Obtaining the current position directly from the floor is not possible. To address this, a conversion is performed to transform the drone's position and orientation from the local coordinate system of ORB-SLAM to a global coordinate system with the floor as the reference origin. In this conversion, AR markers are utilized as aids. Fig.6 illustrates the coordinate systems of the drone and AR marker. To achieve this conversion, we denote the coordinate system of the estimated AR marker as $\boldsymbol{p}_{ar}$, and the coordinate system in which the point cloud was captured using ORB-SLAM as $\boldsymbol{p}_{local}$. Using Eq.(1), we obtain $\boldsymbol{T}_{ar}^{local}$, which represents the simultaneous transformation matrix from the position and orientation in the local coordinate system.

$$\boldsymbol{T}_{ar}^{local} = \boldsymbol{p}_{ar}\boldsymbol{p}_{local}^{-1} \tag{1}$$

Applying the transformation matrix $\boldsymbol{T}^{local}ar$ from Eq.(1) to the position and orientation of the local coordinate system in Eq.(2) yields the corresponding position and orientation in the global coordinate system, denoted as $\boldsymbol{p}_{global}$.

$$\boldsymbol{p}_{global} = \boldsymbol{T}_{ar}^{local}\boldsymbol{p}_{local} \tag{2}$$

### 5.2. Occupancy Grid Map

We created an Occupancy Grid Map using a drone. The method used was to use the feature points from ORB-SLAM[5], which is used as the drone's Visual SLAM. The probability of passability $p_{(x,y)}(k)$ for each grid was set to 0.5 as the initial value since the area was unknown and had not been observed. Then, we modeled the distribution of the heights of the feature points within each grid using a Poisson distribution with the mean height $\lambda_{(x,y)}$. To represent a distribution using a probability density function that exclusively deals with positive integers, I opted to utilize the Poisson distribution. The Poisson distribution $p_{(x,y)}(k)$ is given by the following Eq.(3).

$$p_{(x,y)}(k) = \frac{\lambda_{(x,y)}^{k} e^{-\lambda_{(x,y)}}}{k!} (k = 0, 1, 2...) \tag{3}$$

In this case, $x$ refers to the height of feature points within each grid. by integrating the Poisson distribution used in this study at an arbitrary height $h$, we obtained the traversability probability. The arbitrary height $h$ refers to the minimum height that could be a fatal obstacle for each robot. The traversability probability $m_{(x,y)}$ is represented by the following Eq.(4).

$$m_{(x,y)} = \int_{0}^{h} p_{(x,y)}(k)dk \tag{4}$$

To conform to the specifications of the ground robot used in this study, the height parameter $h$ was set to 0.05m. The generated point cloud map and occupancy grid map using the drone are shown in Fig.7(left) and 7(right), respectively. In Fig.7(right), the scale of each grid was set to 0.2m to ensure that each grid contains about 3 to 4 feature points when assuming an environment of 10m $\times$ 10m. Also, the color of each grid indicates the occupancy rate, with darker colors indicating higher occupancy rates as the color goes from white to black.

In map evaluation, it is crucial to assess the reduction in uncertainty for each generated map. The entropy of a grid can be determined based on its probability of being traversable, represented by $m_{(x,y)}$, and its probability of being impassable, which is equal to $1 - m_{(x,y)}$. By considering these probabilities, the entropy can be calculated using the equation given in Eq.(5).

$$\begin{aligned} H_{(x,y)} = &- m_{(x,y)} \log_2 m_{(x,y)} \\ &- (1 - m_{(x,y)}) \log_2 \{(1 - m_{(x,y)})\} \end{aligned} \tag{5}$$

Consequently, the total entropy sum of $N$ grids can be expressed as $H_{sum}$ using Eq.(6) as follows.

$$H_{sum} = \sum_{x=1}^{N} \sum_{y=1}^{M} H_{(x,y)} \tag{6}$$

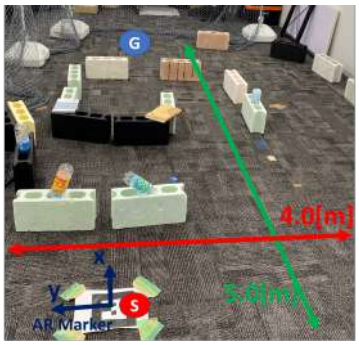


Fig. 8 Search and movement experiment environment

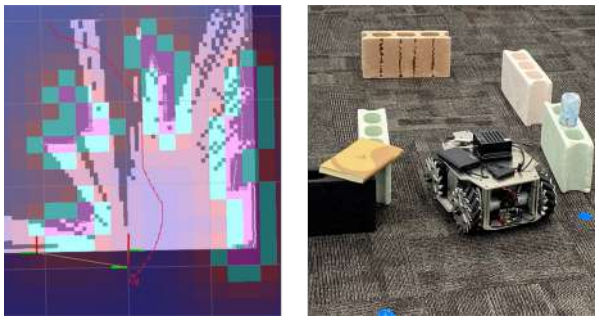

Fig. 9 : Ground robot solo path planning. Ground robot Occupancy Grid Map (left). Dead end reached while following the generated path (right).

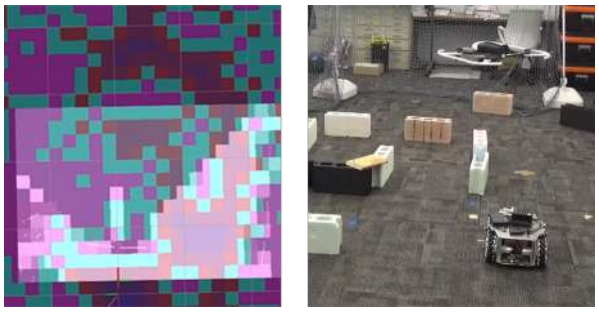

Fig. 10 : Multiple robot path planning. Merge Occupancy Grid Maps of drone and ground robot (left). Optimal path selection from the generated path (right).

Where, let us denote $N$ as the total number of grids in the $x$-axis direction, while $M$ represents the total number of grids in the $y$-axis direction. We investigated how the value of $H_{sum}$ changes with and without the drone's map to determine the uncertainty of the map. Specifically, we compared the value of $H_{sum}$ between the two cases to quantify the impact of the drone's map on the map uncertainty.

## 6. OPTIMAL PATH SEARCH

### 6.1. Experimental environment

The implemented real-time action planning was utilized to verify whether the ground robot could select the optimal path by dynamically exploring with the drone. To conduct the verification, a comparison was made between path finding with a single robot and path finding with multiple robots collaborating in the same experimental environment. The experimental environment is depicted in Fig.8, where the red circle represents the starting point and the blue circle represents the goal point. The origin of the map is set as the AR marker, and the goal point is positioned at x = 1.0m and y = 3.0m relative to the start point. The Dijkstra algorithm from the Navigation2 package was employed for path finding of the ground robot [6]. The representation of occupancy on the map includes pink areas indicating unknown regions and cyan areas representing regions taking into account the cost for the ground robot. Additionally, as the color transitions from blue to purple and red, the occupancy rate increases. The dynamic exploration of the drone involves the user specifying four waypoints in advance, and the search is performed to pass through these waypoints. The assumptions are that the user specifies the goal point of the ground robot, that there are no obstacles on the AR marker, and that no obstacle information is given in advance. The user should specify the path of the ground robot, there should be no obstacles on the AR marker, and no obstacle information should be given in advance.

### 6.2. Exploring single robot

In the case of a single robot, the map generated during path finding is shown in Fig.9(left). With only the obstacle information shown in Fig.9(left), the ground robot generates a path indicated by the dark pink color, which leads it to the dead end as shown in Fig.9(right). When making a U-turn for recovery, the robot often encroached into the light blue area and came to a stop. In other words, it is difficult for a stand-alone robot to follow the optimal route to the goal point if the terrain is even slightly complicated.

### 6.3. Multi robots path planning

In multi-robot path planning, the ground robot utilizes the Occupancy Grid Map created by the drone to navigate along an optimal path. The ground robot relies on its map for observable points, while the drone's map is used for points that are not observable by the ground robot. the drone autonomously generates tasks and expands the BT to perform dynamic exploration. As part of the drone's tasks, when moving toward the landing site, it assumes the absence of obstacles around the AR marker and attempts to land in the vicinity of the AR marker.

Once the drone has created a map of the surrounding area, the ground robot initiates path planning. The resulting map is depicted in Fig.10(left). Consequently, it becomes feasible to perform an optimal route selection and exploration, as illustrated in Fig.10(right). When the ground robot reaches the goal location point and runs out of paths, the drone's BT grows a tree as shown in Fig.11. In BT, blue nodes are successful, red nodes are failures, and yellow nodes are running. It then begins to move onto

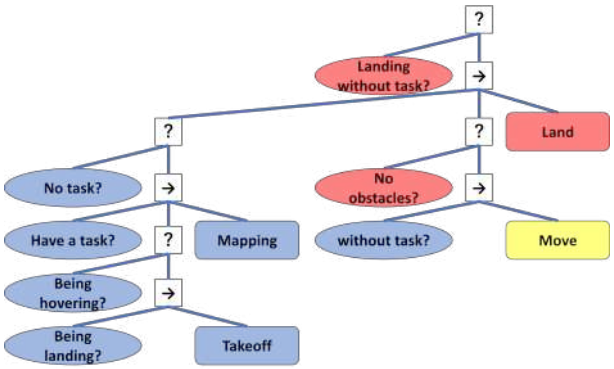


Fig. 11 BT for dynamic exploration with drones

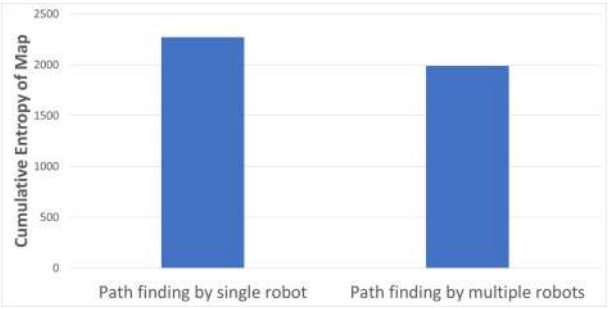


Fig. 12 Comparative analysis of accumulated entropy in path planning for single robot and multiple robots

the AR marker, which is assumed to be free of obstacles. By moving onto the AR marker and landing, BT considers all tasks completed. Based on the above results, it has been confirmed that the ground robot can make appropriate path selections by utilizing the drone's Occupancy Grid Map. We also confirmed that the drone can perform tasks autonomously using BT.

Furthermore, the cumulative entropy of each generated map was calculated using Eq.(5) and (6). Fig.12 illustrates the comparison of the cumulative entropy between the two maps. The left bar graph in Fig.12 represents the cumulative entropy of the map generated by the ground robot, as computed in Fig.9 (left), with a value of 2271. Similarly, the right bar graph represents the cumulative entropy of the map created through cooperative teamwork, as calculated in Fig.10 (left), with a value of 1987. Therefore, comparing the cumulative entropy, it can be observed that the map created by multiple robots resulted in a reduction of 284 in cumulative entropy, indicating a decrease in map uncertainty.

A challenge in creating an Occupancy Grid Map with a drone is incorporating the feature points captured by the ground robot into the map. This can lead to the ground robot perceiving obstacles in its current location based on the map's reflection, hindering the ability to perform complex tasks and preventing the ground robot from replanning its path. This issue poses a significant challenge in drone-based Occupancy Grid Mapping.

## 7. CONCLUSION AND FUTURE WORK

This paper presents a system where robots with varying specifications can request tasks from a real-time action planning system. The system includes a search path system, where a drone assists a ground robot in searching along a suitable route. The drone provides the ground robot with the entire map, allowing for a comprehensive check before the ground robot initiates the search. Furthermore, we demonstrate the autonomy of the drone and the automatic task generation capability of the real-time action planning system.

Detecting and correcting inconsistencies caused by incorrect recognition of conditions and action nodes in real-time action planning is a crucial challenge for future research. Moreover, if a ground robot can independently execute tasks, collaboration with other robots becomes unnecessary. Hence, it is essential to establish a system where robots can mutually request tasks and self-assess the complexity of each task, leading to natural collaboration. Additionally, users typically specify four waypoints in advance for the flight in current dynamic drone exploration. However, optimizing the drone's flight distance and efficient exploration is important due to concerns about battery consumption. Moreover, the generated map may contain unnecessary information for the ground robot. Thus, the drone needs to perform dynamic exploration to maximize mutual information for efficient exploration.

## REFERENCES

[1] J. Li, T. Sun, X. Huang, et al., "A Memetic Path Planning Algorithm for Unmanned Air/Ground Vehicle Cooperative Detection Systems," IEEE *Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2724-2737, 2022.

[2] E. H. C. Harik, F. Guérin, et al., "UAV-UGV Cooperation For Objects Transportation In An Industrial Area," IEEE *International Conference on Industrial Technology (ICIT)*, pp. 547-552, 2015.

[3] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*, pp. 6366-6373, 2015.

[4] M. Colledanchise, P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," *Artificial Intelligence and Robotics Series*, pp. 93-126, 2018.

[5] A. Yusefi, A. Durdu, and C. Sungur, "ORB-SLAM-based 2D Reconstruction of Environment for Indoor Autonomous Navigation of UAVs," *European Journal of Science and Technology*, pp. 466-472, 2020.

[6] S. Macenski, F. Martín, R. White and J. G. Clavero, "The Marathon 2: A Navigation System," IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*, pp. 2718-2725, 2021.