Designing a Simulator Structure for Validating Swarm Robot's Dynamic Mission Planning

Mingyu Shin[1], Sejin Kim[1], Hyojun Ahn[1], Sunoh Byun[1], Eenseo Baek[1], Sungjun Shim[2], Jooyoung Kim[2], and Yongjin Kwon[1,*]

[1]Department of Industrial Engineering, College of Engineering, Ajou University, Suwon, South Korea
[2]Intelligent/Autonomous Control SW Team, LIG Nex1, Sungnam, South Korea
Email: saycode99@ajou.ac.kr (M.G. S.); jkn6873@ajou.ac.kr (S.J. K.); hyojunahn000713@ajou.ac.kr (H.J. A.); eunseo5343@ajou.ac.kr (E.S. B.); sungjun.shim@lignex1.com (S.J. S.); jooyoung.kim@lignex1.com (J.Y. K.); yk73@ajou.ac.kr (Y.J. K.)
[*]Corresponding author

Manuscript received Month date, 2023; revised Month date, 2023; accepted Month date, 2023; published February 6, 2024

***Abstract*—As swarm-based robotic missions become increasingly important, efficient mission planning for multiple robots in a swarm has become one of the key issues. When deploying swarm robots for indoor search tasks, it is necessary to have a dynamic mission plan that can respond to the changing environment in real time, rather than being limited to the static mission plan initially set. However, there is no standardized method for swarm robots to effectively respond to the changing environment after entering a building with an initially assigned mission plan. Therefore, this paper proposes a new type of simulator structure that can respond and validate dynamic mission planning algorithms by considering swarm robots with multiple unit missions. Two decision tree-based mission planning algorithms were designed and reflected in the simulator based on the proposed simulator structure. In there, two algorithms were compared and tested. In addition, an inheritance hierarchy for swarm unit missions was provided, so that newly required missions among the multiple unit missions of the robot could be easily implemented and reflected in the simulator. As a result, the proposed simulator structure is expected to have a wide applicability in that it can be used for research and verification of dynamic mission planning algorithms that can respond to the environment not only in indoor navigation tasks but also in other environments where robots with multi-unit missions.**

***Keywords*—mission planning, unmanned systems, SW architecture, swarm robots, dynamic mission planning**

I. Introduction

In disaster situations, robots are becoming increasingly important, especially in situations where direct human intervention is difficult or dangerous, such as exploring buildings and searching for survivors. While robots can be more effective when swarmed and working as a team, assigning tasks to each robot individually is a very complex and challenging task, and requires dynamic mission planning that is not limited to the initially set plan but can respond to the changing environment in real time. The complexity of dynamic mission planning has led to the need for this research, where multiple robots perform tasks in groups to maximize the efficiency of their assigned tasks.

Currently, there is no standardized way for robots to effectively respond to the changing environment after entering a building with an initially assigned mission plan. Therefore, in this thesis, we design and propose the structure of a simulator that can validate mission planning algorithms to ensure that robots can operate efficiently in dynamic environment. Furthermore, we use this simulator to validate the behavior of mission planning algorithms and propose ways for swarm robots to perform their missions more effectively [1, 2].

To further enrich this research, the authors present additional contributions based on their understanding of recent trends in robotics. In the work provided by Savkiv et al. [3], an adaptive gripping device for industrial robots is proposed. The device combines the ability to grasp manipulated objects of different shapes with the ability to control deviations in the shape of the object. This approach helps to increase the flexibility and precision of the robot. In other work done by Mykhailyshyn *et al*. [4], an approach to the classification of gripping devices for industrial robots is analyzed and supports the need for a systematization of pneumatic gripping devices. This study provides an in-depth understanding of the diversity and segmentation of robotic grip devices, which opens new directions for improving the ability of robots to perform various tasks. These two studies demonstrate the latest trends in robotics that can respond to the changes and unexpected variations that can frequently occur in real world situations. Even though the application areas are different, this study and the above two studies carry the similar approach in view of the changing environment where the robotic applications often come in to contact. By reflecting the recent trends and by considering other related research works, the validation methods of the simulator and mission planning algorithm proposed in this study can be further improved and become more effective and practical.

II. Literature Review

The objective of this research is to design the structure of a simulator that can validate the mission planning algorithm of multiple robots. The technologies behind this research are as follows. Table 1 shows the basic concepts required for mission planning of swarm robots. With these organized concepts, one can define what mission planning is, and get a sense of how missions are created and controlled in a cluster, which will lead to the design of a simulator structure.

In this section, we envision and robustly design how the robot's state changes due to the mission plan, how it will react to those changes, and how it will structure the flow of data between the logics. In most cases, the operator establishes a static mission plan before deploying the robots, and the operator controls the robot in real-time during operation.

DOI: 10.7763/IJMO.2024.V14.845

Table 1. Mission planning terminology

Terminology		Definition
Cluster objects		Swarm Robot
Platforms		A generic term for swarm robots, transportation robots, and communication relay robots.
Cluster Subgroup		Swarm organizational units that perform missions.
Task		Role of a swarm object during a mission. What the swarm group will do.
(Swarm) Mission Planning		Tactics. Planning how to accomplish a mission using cluster subgroup composition, tasks, objective coordinates, range of operations, etc.
(Swarm) Mission Planning	Static	The operator enters mission information such as 'target coordinates, cluster subgroups, and tasks' into the operation control SW to obtain a mission plan.
	Dynamic	Automatically create & execute mission plans based on event occurrences or situational / environmental information.
(Cluster) Mission Planning How to Create	Manual	Operators manually enter mission plans
	Automatic	The operator enters the information required for the mission plan and generates the mission plan based on the conditions.
(Swarm) Mission Planning Control Methods	Centralized	Swarm objects execute missions according to the commands passed in the Mission Plan SW.
	Distributed Control	Create & execute dynamic mission plans by communicating between leaders of cluster subgroups (without Mission Planning SW control)
Route Planning		Set waypoints to get to the target coordinates. Tasks and target coordinates correspond to
Telecom shaded areas		Areas where radio waves cannot be received.

However, there are limitations to real-time responses due to communication breakdowns and system failures. Recently, with the development of high-performance sensors and the advancement of artificial intelligence technology, research is being conducted to automate the mission planning of swarm robots. Since robots cannot be operated autonomously without human intervention, a hybrid mission planning method also exists.

The advantages of centralized mission planning are that it's easy to manage data centrally, and the cost of configuring the system is lower than other mission planning methods. The disadvantages are that data from all robots must be centralized, which can lead to data bottlenecks. The operational radius is also narrowed because they can only operate where they can communicate with the center. In addition, if the centralized command center is destroyed, mission planning for the remaining robots becomes impossible. Fig. 1 illustrates the centralized mission planning approach for an unmanned surface vehicle [1, 2].

The solution to the centralized problem is to decentralize the mission planning, which is often used when communications between robots are not possible. In the case of the distributed mission planning method, the mission is planned through the information collected by the individual robots without sharing situational awareness information between the robots. However, it is very expensive and time-consuming to set up the system. Fig. 1 illustrates the distributed mission planning approach for unmanned surface vessels [5, 6]. Fig. 1 shows a schematic of a hybrid mission planning technique. A centralized control center and a swarm leader platform control the unmanned platforms that are followers. In the hybrid mission planning method, which combines the disadvantages of centralized and decentralized mission planning methods, the leader platform of each swarm plans the missions of its followers [7].

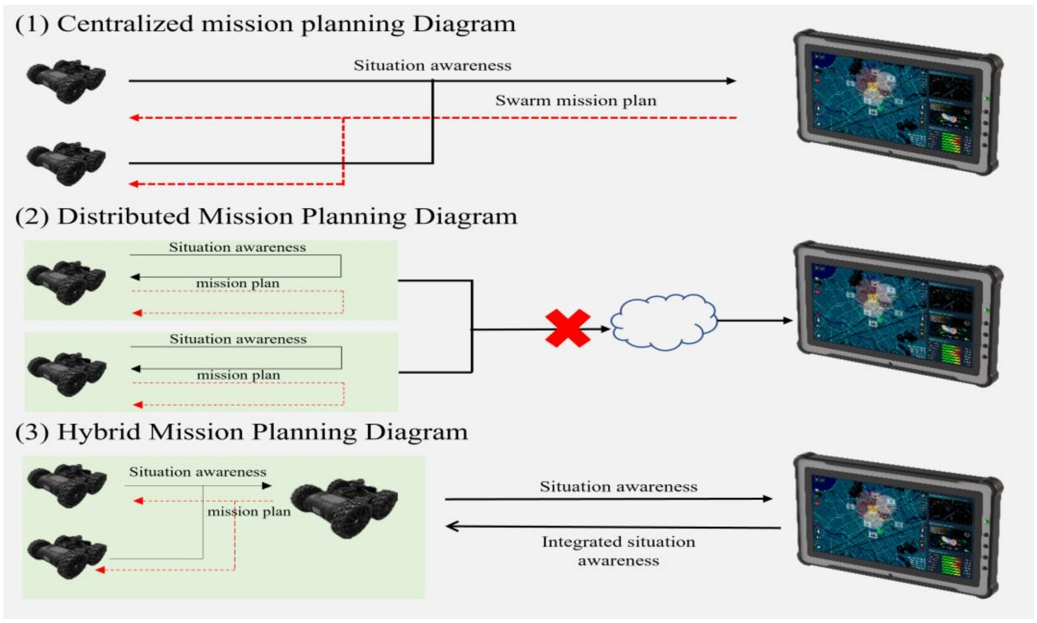


Fig. 1. Mission planning diagram.

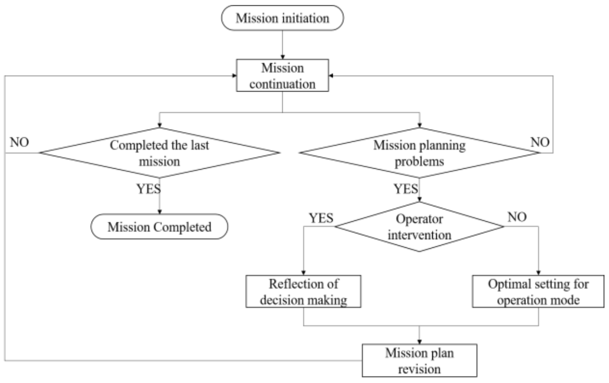

Mission initiation
Mission continuation
Completed the last mission
Mission planning problems
NO
NO
YES
YES
Mission Completed
Operator intervention
YES
NO
Reflection of decision making
Optimal setting for operation mode
Mission plan revision

Fig. 2. The process of mission management being performed.

In Fig. 2, one can see that the mission plan is modified by the optimal operational mode setting, even if the operator's decisions are not reflected [2]. To date, most mission planning algorithm research has been focused on path planning, especially based on CBBA algorithms. They mainly demonstrate techniques for assigning multiple UAVs to different traveling missions by calculating the cost of traveling paths. However, this technique is inappropriate for indoor search, which requires multiple unit missions. Examples include surveillance missions to detect object targets, communication relays to organically send and receive status and situation data, as well as mapping and pathfinding missions required for indoor search. To overcome and improve the limitations of existing technologies, it is necessary to develop a new technology for autonomously planning multiple missions for swarm robots. Therefore, we design a simulator structure that can verify the mission planning algorithm, and proceed with the verification through a simple swarm mission planning algorithm [8–11].

III. Materials and Methods

We propose a simulator structure to validate the mission planning algorithms, as shown in Fig. 3. The structure of the proposed simulator is mainly divided into the Simulation Environment, User Interface, and Data Management Module. Simulation Environment literally refers to the environment in which the mission planning algorithm can be simulated and is divided into Mission Planning Module and Virtual Environment.

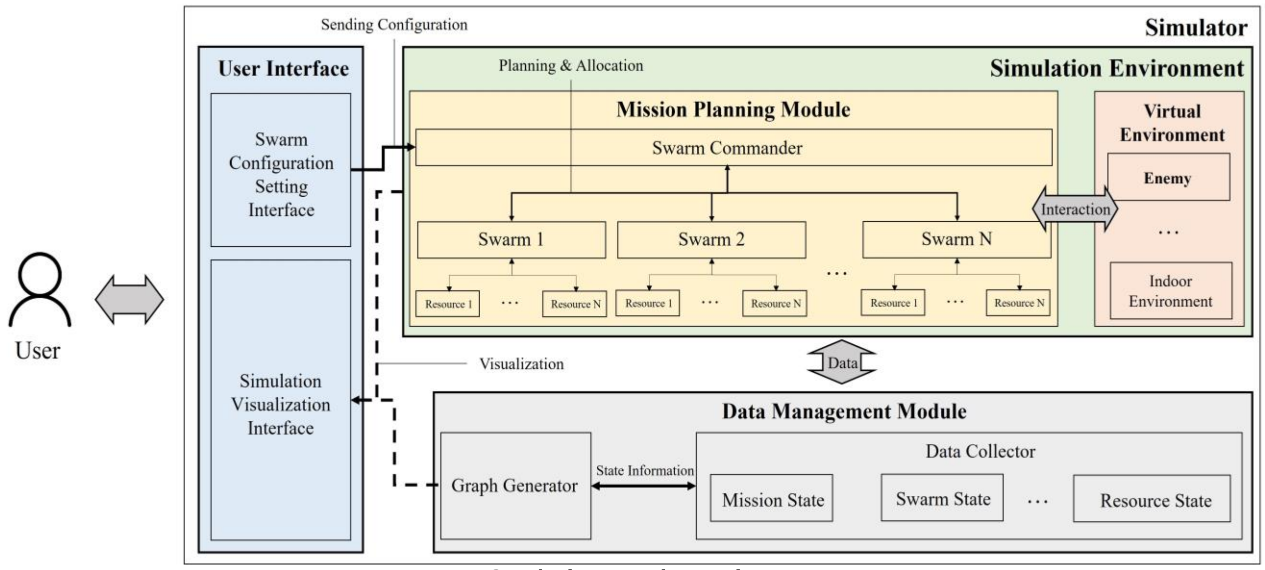


Fig. 3. Hybrid mission planning diagram.

1) *Mission planning module*

The Mission Planning Module is where the mission planning algorithm that the operator wants to validate is implemented and validated in the simulator. The Mission Planning Module is organized in a hierarchical structure of Swarm Commander, Swarm, and Resource, and mission planning is centered on the Swarm Commander.

'Swarm Commander' is composed of two main parts, 'Mission Planning', which plans the mission, and 'Swarm List', which references the swarms it belongs to, as shown in Fig. 4. 'Mission Planning' is the part where the actual mission planning algorithm is implemented and driven. It is divided into 'Static Mission Planning', which plans the mission immediately when the initial user sets the mission, and 'Dynamic Mission Planning', which modifies the mission plan in real-time, as the planned situation changes. Simulator users can easily verify the algorithm by implementing the mission planning algorithm they want to verify in this part. Then, the swarm-specific missions planned by the Mission Planning algorithm are assigned to the swarms referenced in the ‘Swarm List’. The assigned mission is delivered in the form of a ‘Swarm Unit Mission’. ‘Swarm Unit Mission’ is an abstract class object that expresses the mission that the swarm should perform, and detailed unit missions inherit from 'Swarm Unit Mission' and are implemented in a polymorphic form to suit the characteristics of each mission.

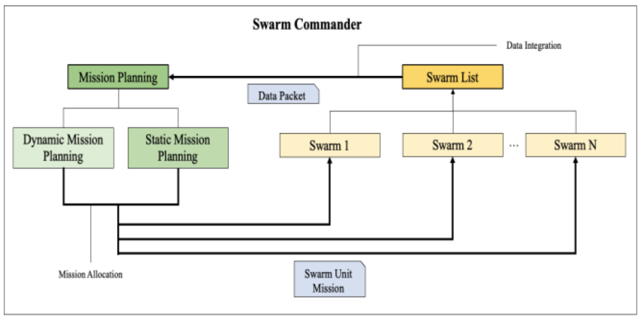


Fig. 4. Swarm commander internal structure.

The inheritance structure of Swam Unit Mission and its subunits is shown in Fig. 5. Swam Unit Mission, implemented as an abstract class, has common member variables and methods. The name variable, which represents the mission's name of all the detailed missions, and the duration variable, which represents the duration of the mission, are field members.

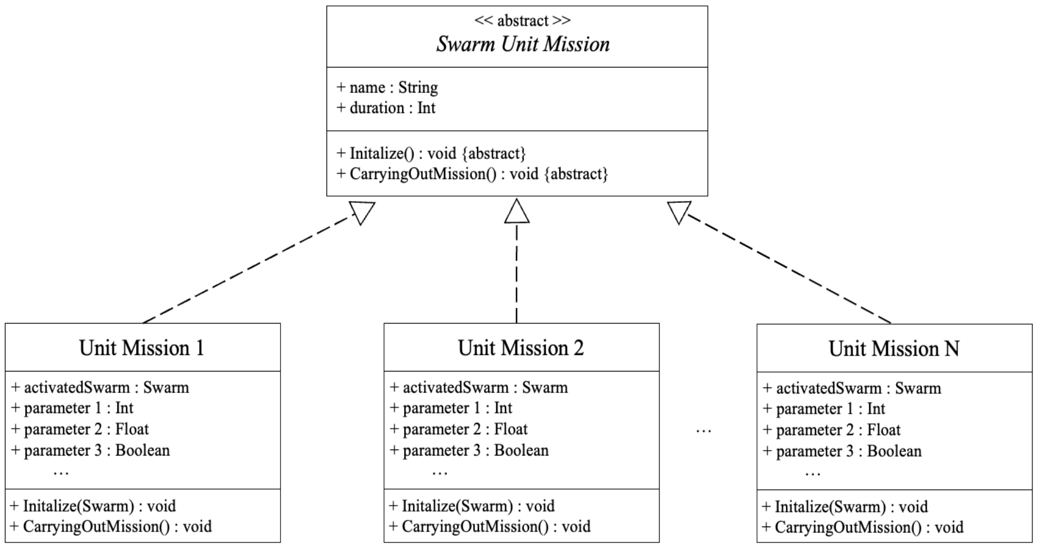


Fig. 5. Swarm unit mission and detail unit mission class inheritance structure.

In addition, there are two abstract methods, 'Initialize()', which performs a common initialization before starting all missions, and 'CarryingOutMission()', which actually carries out the mission, and each subdivided unit mission can override these methods to implement a polymorphic form. The overridden 'Initialize(Swarm)' method registers the 'Swarm' passed in as an argument with 'activatedSwarm', receives the swarm to carry out the unit mission, and performs the initial setup. The 'CarryingOutMission()' method is where the registered 'activatedSwarm' actually carries out the mission, and the specific details will vary depending on the type of mission. The structure of 'Swarm', which receives unit tasks from 'Swam Commander', is illustrated in Fig. 6. After receiving the unit task from the Swarm Commander, the Swarm assigns the task to the various 'Resources' referenced in the Robot List (Swarm). The method of assigning tasks to 'Resources' utilizes various predefined rules or algorithms. The 'Robot List' (Swarm) collects the status information of the 'Resources' in real-time for dynamic mission planning and delivers it to the 'Swarm Commander' in data packets.

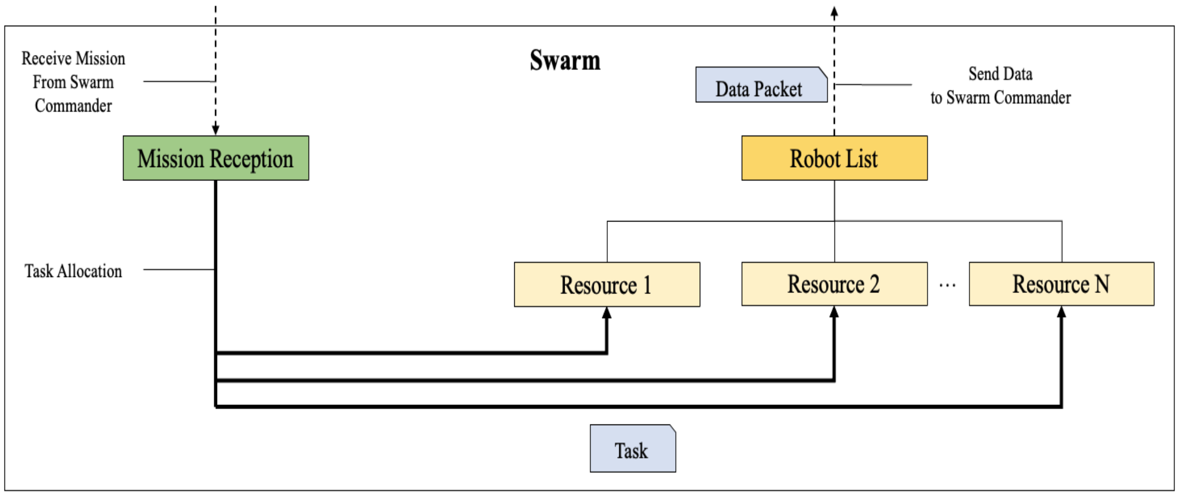


Fig. 6. Swarm internal structure.

2) *Virtual environment*

'Virtual Environment' refers to the environment in which multiple robots operated by the Mission Planning Module operate and interact. The 'Virtual Environment' includes the indoor environment where the mission takes place with interactive enemies. The operator implements the scenario to be simulated in this 'Virtual Environment' and verifies it with the robots performing the mission planning algorithm.

3) *User interface*

The 'User Interface' is literally the part that directly communicates with the user and exchanges necessary information. It consists of the 'Swarm Configuration Setting Interface' for initial setup and the 'Simulation Visualization Interface' for monitoring the simulation.

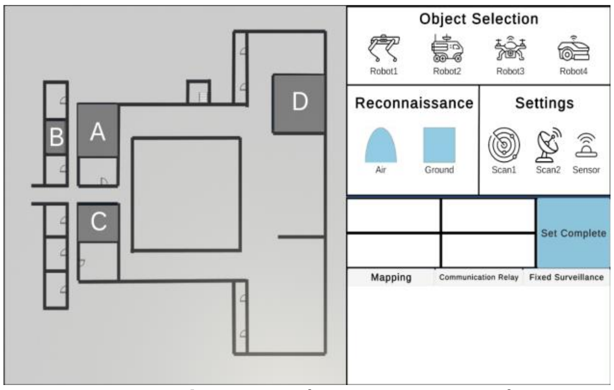
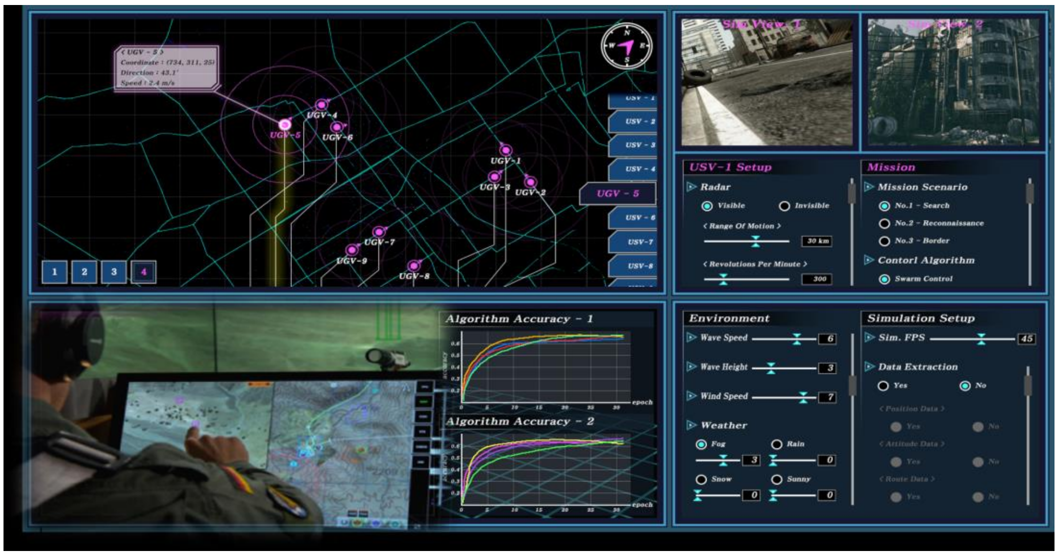


Fig. 7. Example swarm configuration settings interface.

The ‘Swarm Configuration Setting Interface’ is the part that sets the mission goal of ‘Swarm Commander’ and performs initial settings such as the number of swarms and robot types. The initial settings utilize a drag-and-drop based GUI (Graphical User Interface) to make it easy and intuitive for the operator to set up. Fig. 7 shows an example of the swarm configuration setup interface, which utilizes multiple setup elements on the right to set initial information by dragging and dropping. The 'Simulation Visualization Interface' is literally an interface where you can see the simulation taking place in the actual simulation environment. The interface consists of several panels to check the status information of the robot, mission performance information, etc. and an interface such as buttons for the operator to change the desired panel. Fig. 8 is an example of the Simulation Visualization Interface.


< UGV - 5 >
Coordinate : (734, 311, 35)
Direction : 43.1°
Speed : 2.4 m/s

USV-1 Setup
Radar: [✓ Visible] Invisible
< Range Of Motion > 30 km
< Revolutions Per Minute > 300

Mission
Mission Scenario: [✓ No.1 - Search] No.2 - Reconnaissance No.3 - Border
Contorl Algorithm: [✓ Swarm Control]

Environment
Wave Speed 6
Wave Height 3
Wind Speed 7
Weather: [✓ Fog] 3 Rain 0 Snow 0 Sunny 0

Simulation Setup
Sim. FPS 45
Data Extraction: Yes [✓ No]
< Position Data > Yes No
< Attitude Data > Yes No
< Route Data > Yes No

Fig. 8. Example of simulation visualization interface.

4) *Data management module*

Finally, the ‘Data Management Module’ is a module that collects data generated between simulations and processes it for analysis. First, the ‘Data Collector’ collects mission progress information, swarm status information, and robot status information. Then, after the simulation is in progress or over, it creates graphs based on the data collected through the ‘Graph Generator’ to provide various information to the operator. Graphs that provide various information such as mission execution time and enemy detection count are provided. With this feature, it will be easy to verify the completeness and effectiveness of the mission planning algorithm. Using the above structure of the proposed simulator, it will be easy to implement and verify various mission planning algorithms. In the following Results and Discussion section, to verify the effectiveness and efficiency of the proposed structure, we will implement a simple simulator based on the structure and verify the mission planning algorithm.

IV. Result and Discussion

Fig. 9 shows an environment in which you need to perform an indoor navigation task. The indoor environment contains 10 randomly placed objects. We assume that the doors in each room represent entryways and that the indoor map information is unknown. Initially, the navigation task starts when the user statically assigns the task, and the task duration is assumed to be 12 minutes.

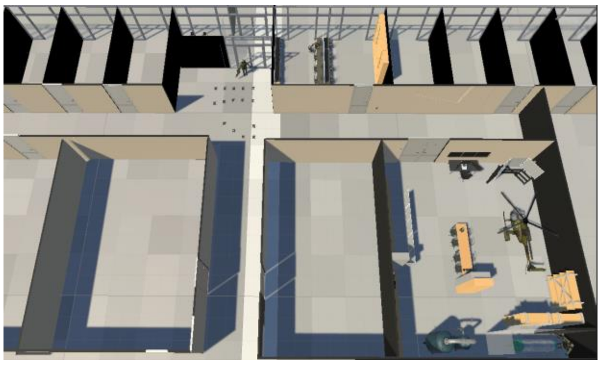


Fig. 9. Setting up the environment based on the proposed simulator structure.

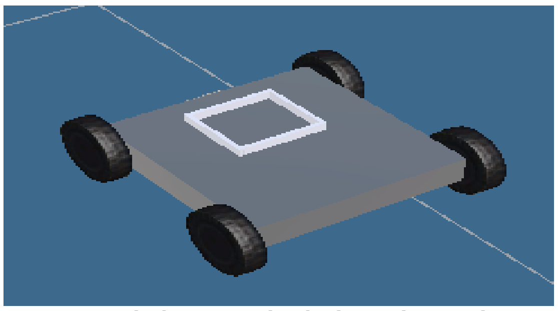


Fig. 10. A ground robot equipped with a low-end monocular camera modeled in the simulator.

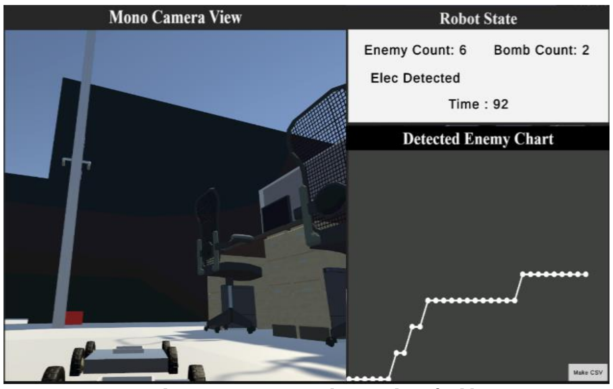


Fig. 11. Simulation environment being identified by sensors.

As shown in Fig. 11, one can check whether an object is detected by the sensors onboard the robots. In the upper right corner, one can see detection information, operation time, and contextual notifications. The screen can be switched for each robot to view the simulation environment from different angles, as shown in Fig. 12. Additionally, the number of detected objects can be graphed, and the simulation result data can be extracted to a CSV file by pressing the button at the bottom right. The above simulation can be validated by selecting and changing only the dynamic mission planning algorithm and comparing the number of object detections for the same operation time. To simplify the validation of the algorithm based on the proposed simulator structure, we use a decision tree as a mission plan [7], as shown in Fig. 13. When the mission first starts, it is given the task of either finding an entryway or mapping it, depending on whether it has map information or not. If it takes more than 150 seconds to find an on-ramp, you are assigned a surveillance mission.

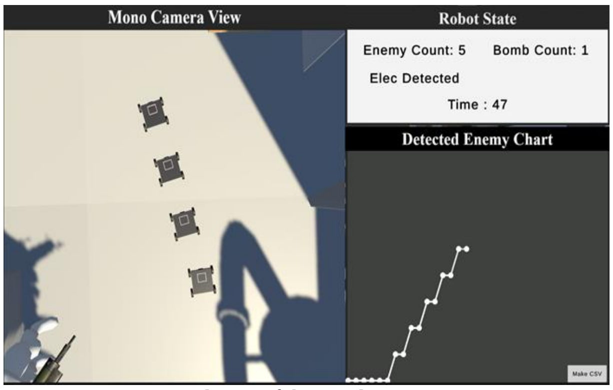


Fig. 12. Aerial view of the simulation environment.

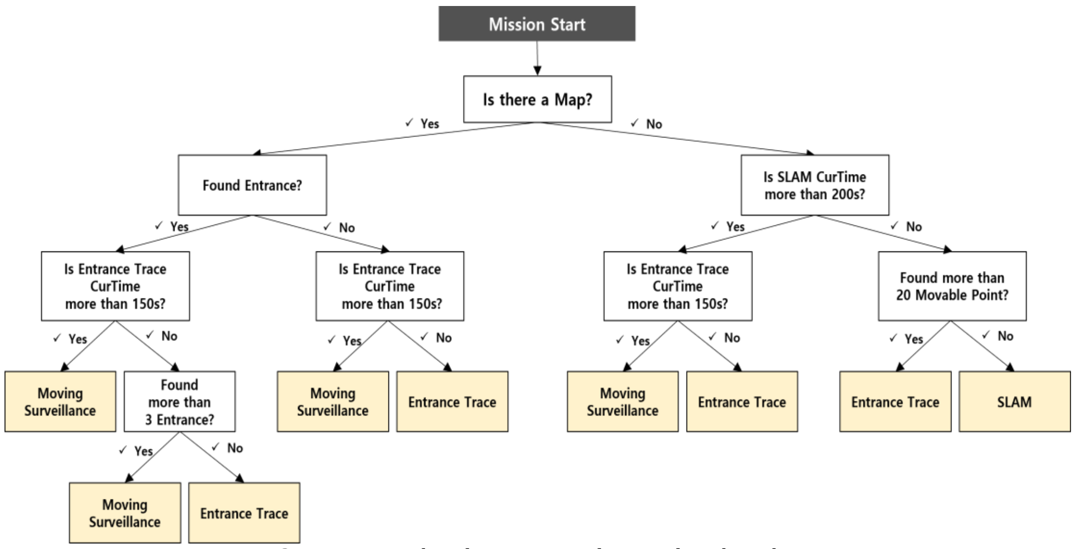


Fig. 13. Decision tree-based unit mission planning algorithm schematic.

Exceptionally, if three or more driveways are found, a surveillance mission is also planned. A map generation mission is planned if there is no map information at the start of the scenario or if more than 20 coordinates to move to are not found within 150 seconds of the start of the map generation mission. Finally, a ramp finding mission is planned if the opposite conditions of surveillance missions are met. As an exception, at least 20 coordinates within 150 seconds of the start of the ramp finding mission are found.

The robot's swarm mission is planned and executed by the algorithm being set up, as shown in Fig. 14. At 44 seconds into the simulation, all the swarms are mapped by the decision tree we designed, and if any swarm has found more than 20 movable points, the mission is planned to find an entry point.

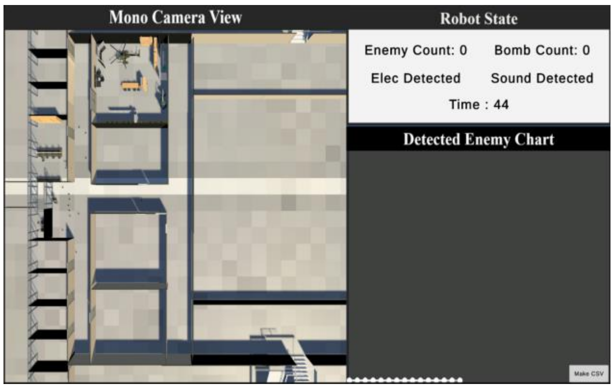


Fig. 14. Simulated situations with mission planning algorithms.

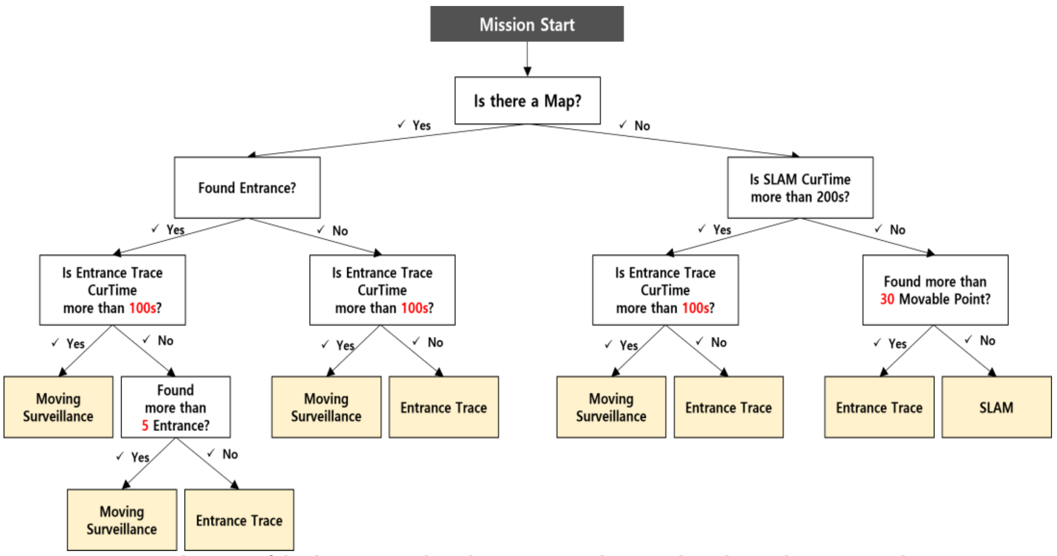
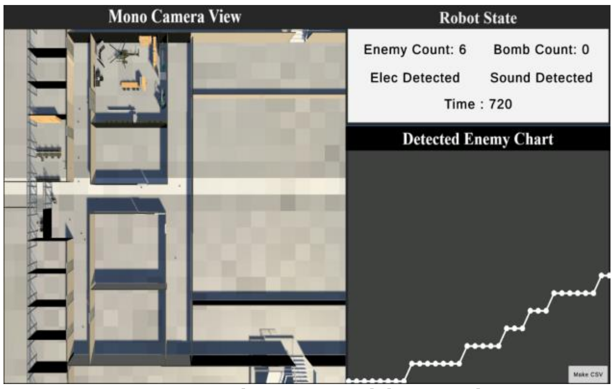

Mission Start
Is there a Map?
✓ Yes
✓ No
Found Entrance?
Is SLAM CurTime more than 200s?
Is Entrance Trace CurTime more than 100s?
Found more than 30 Movable Point?
Moving Surveillance
Found more than 5 Entrance?
Entrance Trace
SLAM

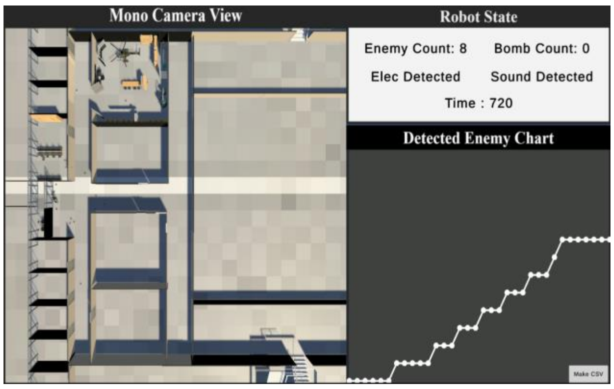
Fig. 15. Schematic of the decision tree-based unit mission planning algorithm with parameter changes.

To validate the algorithm, partially change the parameters of the previously used decision tree, as shown in Fig. 15. First, in mission planning, change the parameter from 150 seconds to 100 seconds based on the execution time to find the ramp. Second, if the time to find the ramp is less than the given execution time, re-plan the mission based on the number of ramps found, changing the parameter from 3 to 5. Finally, for mapping, we set the simulator to re-plan the mission by changing the initial execution time from 150 seconds to 200 seconds and changing the number of points that can be moved within the given execution time to 30.



Enemy Count: 6 Bomb Count: 0
Elec Detected Sound Detected
Time : 720

Fig. 16. First decision tree validation result.



Enemy Count: 8 Bomb Count: 0
Elec Detected Sound Detected
Time : 720

Fig. 17. Second decision tree verification result.

Fig. 16 and Fig. 17 are user interfaces that show the validation results for each decision tree. Table 2 shows the results of the comparative validation of the decision tree-based mission planning algorithm. You can see that after changing the parameters of the nodes, the number of object detections increases within the same operation time.

Table 2. Validate a comparison between two algorithms

	Before Change Parameter	After Change Parameter
Spent Time in Simulation(s)	720	720
Number of detections	6	8
First detection time(s)	170	127
Last detection time(s)	672	588
Detection success rate(%)	60	80

Although the mission planning algorithm under comparison was a simple decision tree with only the parameter values changed, by customizing the desired unit mission and outcome metrics based on this simulator structure, and applying the mission planning algorithm to the method called 'DynamicMissionPlanning()' inside the structure, you can perform comparative validation of the algorithm and select a better mission planning algorithm.

V. Conclusion

Through the proposed simulator structure, we were able to apply and verify algorithms for planning multiple robot missions that can be utilized in search operations. It is significant in that it solved the problem of having to develop a simulator each time to verify the mission planning algorithm because it is concentrated on the research of path planning-based mission planning algorithms, and solved a practical problem in that it was only necessary to design missions in classes according to the purpose. It is also expected to be scalable in that it is easy to develop AI-based mission planning algorithms and useful for developing verification simulators for autonomous unmanned systems in the future if only an environment for learning AI is additionally built.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Mingyu Shin, Sejin Kim, Hyojun Ahn and Sunoh Byun conducted the research; Eunseo Baek analyzed the data; and Sungjun Shim, Jooyoung Kim, and Yongjin Kwon wrote the paper. All authors had approved the final version.

Funding

This work was supported by Korea Research Institute for defense Technology planning and advancement (KRIT) grant funded by the Korea government (DAPA (Defense Acquisition Program Administration)). (No. KRIT-CT-22-006-005, Control technology for collective operation of military ultra-small ground robots).

References

[1] H. M. Park, "Design and implementation of interface system for swarm USVs simulation based on hybrid mission planning," *Journal of Sensors*, 2018.

[2] S. Hong, K. Kim, and H. Kim, "Unit mission based mission planning and automatic mission management for robots," *Journal of Intelligent and Robotic Systems*, vol. 70, pp. 1–4, 2013.

[3] V. Savkiv, R. Mykhailyshyn, P. Maruschak, V. Kyrylovych, F. Duchon, and L. Chovanec, "Gripping devices of industrial robots for manipulating offset dish antenna billets and controlling their shape," *Transport*, vol. 36, no. 1, pp. 63–74, Mar. 2021.

[4] R. Mykhailyshyn, V. Savkiv, P. Maruschak, and J. Xiao, "A systematic review on pneumatic gripping devices for industrial robots," *Transport*, vol. 37, no. 3, pp. 201–231, Aug. 2022.

[5] I. Hwang, H. Kim, and S. Kim, "Modified consensus based auction algorithm for task allocation of multiple unmanned aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 86, no. 2, pp. 361–377, 2017

[6] F. Pecora, A. Bernardino, and L. Iocchi, "Planning under Uncertainty for robotic tasks with mixed observability," *IEEE Transactions on Robotics*, 2019.

[7] H. I. Christensen and A. T. Fiore, "A review of task planning in robotics," *Robotics and Autonomous Systems*, 2013.

[8] N. Michael and V. Kumar, "A survey of planning algorithms in multi-robot systems," *Robotics and Autonomous Systems*, 2014.

[9] A. Farinelli, L. Iocchi, and D. Nardi, "Multi-robot task allocation in uncertain environments: A survey," *Autonomous Robots*, 2016.

[10] S. Lee, S. Lee, and S. H. Choi, "A task planning method for cooperative unmanned aerial vehicle systems," *Journal of Intelligent and Robotic Systems*, 2018.

[11] H. Zhu, Y. Zhang, and D. Wu. "A multi-objective optimization Method for multi-robot task planning," *IEEE Transactions on Industrial Informatics*, 2019.

[12] T. J. Choi and C. W. Ahn, "A swarm art based on evolvable boids with genetic programming," *Journal of Advances in Information Technology*, vol. 8, no. 1, pp. 23–28, February 2017.

[13] S. Geetha, G. Poonthalir, and P. T. Vanathi, "A hybrid particle swarm optimization with genetic operators for vehicle routing problem," *Journal of Advances in Information Technology*, vol. 1, no. 4, pp. 181–188, November, 2010. doi:10.4304/jait.1.4.181-188