Automatic Path Generation of USV Using Reinforcement Learning for Complex Coastal Areas

Sejin Kim, Mingyu Shin, Hyojun Ahn, Sunoh Byun, Eunseo Baek, and Yongjin Kwon*

Department of Industrial Engineering, College of Engineering, Ajou University, Suwon, South Korea
Email: jkn6873@ajou.ac.kr (S.J.K.); saycode99@ajou.ac.kr (M.G.S.); hyojunahn000713@ajou.ac.kr (H.J.A.); sunoh1020@ajou.ac.kr (S.O.B); eunseo5343@ajou.ac.kr (E.S.B.); *yk73@ajou.ac.kr (Y.J.K.)
*Corresponding author
Manuscript received May 30, 2023; revised June 15 date, 2023; accepted June 25, 2023; published April 15, 2024

***Abstract*—In order to derive local search path points for unmanned surface vehicles in complex coastal areas, the operator has to take the points manually. To solve this problem, we propose an algorithm for automatic generation of local search path points that combines reinforcement learning and the existing shortest path planning algorithm. The demonstration results of the algorithm show a reduction in travelling distance of about 34% and a reduction in redundant visit of 66%. It is expected that this developed algorithm can be more effectively used to automatically generate search path points for complex coastal areas in the future.**

***Keywords*—reconnaissance, reinforcement learning, A* Algorithm, path planning, automation, simulation**

I. Introduction

Unmanned surface vehicles (USVs) have recently been used in a variety of applications [1]. At the same time, advances in automated navigation technology are presenting many new possibilities. In particular, in the field of search operations, USVs are being used for various purposes, such as life-saving, marine resource exploration, and maritime surveillance [2]. In this regards, an efficient search path planning is increasingly becoming important. However, the existing search path planning method is not only time-consuming because it manually takes route points, but also has limitations in that it relies on the experience and intuition of experts. This makes it difficult to ensure the efficiency and effectiveness of the plan [3]. Therefore, there has been a demand for an automated search path generation algorithm.

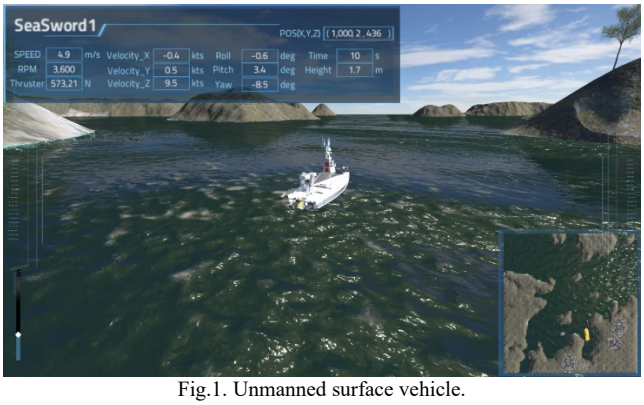

Fig.1. Unmanned surface vehicle.

In response, this paper presents an automatic search path generation algorithm to improve the efficiency of USV search operations. The main focus is to automatically generate the path routing for complex coastal areas, where many small islands, shallow waters, and other obstacles hinder the efficient generation of path plans. The algorithm does not rely on the experience of experts, provides a technical basis for safe and accurate operations, and is expected to increase the success rate of USV search operations. The results of this study are expected to make significant contributions to the research and industry of unmanned surface vehicles, further expanding the applications of USVs and providing a technical basis for safer and more efficient operations, as in Fig.1.

II. Literature Review

This section relates to an algorithm development and methods for automatically generating routes using reinforcement learning techniques for USVs to improve the efficiency of complex coastal search operations. The technology behind the invention is as follows. A schematic diagram of the prior work, entitled “method and system for optimal path finding in an unmanned autonomous vehicle using a sensor” is the prior patent technology in South Korea, as shown in Fig. 2.

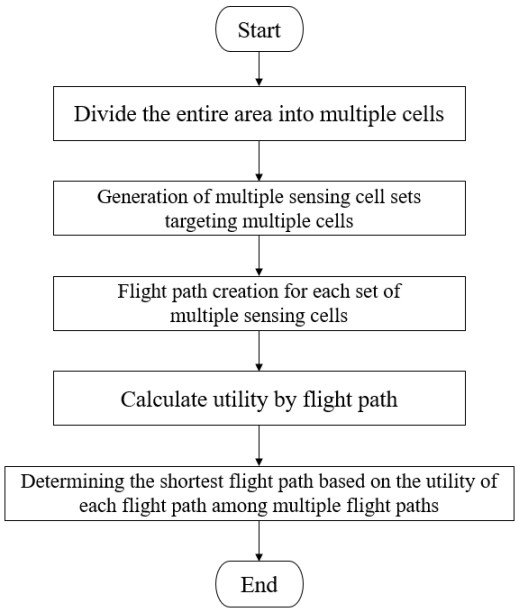

Fig. 2. Prior algorithmic schematic.
* This diagram is from prior patent in South Korea, entitled “method and system for optimal path finding in an unmanned autonomous vehicle using a sensor” *

In this flow, the entire region is divided into a plurality of cells, and a set of sensing cells is generated from which sensor information is collected. It then sets the shortest flight path based on the utility calculated from each flight path generated for each set of cells.

These techniques generate routes that visit only certain cells in a specific region, which is inappropriate for the goal

of this study. That is, the USVs have to visit all cells in the region at least once. In addition, the method of finding the optimal value by calculating the utility of each path is disadvantageous in terms of computational complexity. This is more obvious in a situation requiring many cells in a complex coastal area of Korea, as shown in Fig. 3.

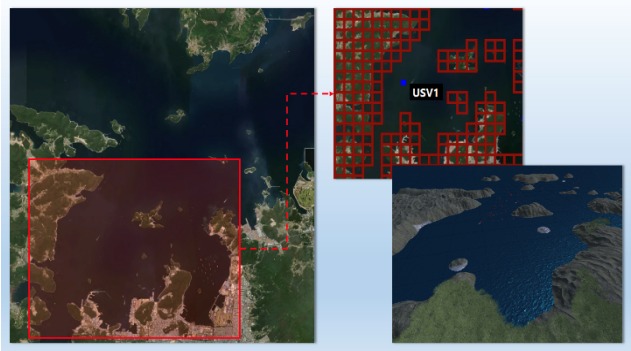

Fig. 3. Complex Korean coastal area.

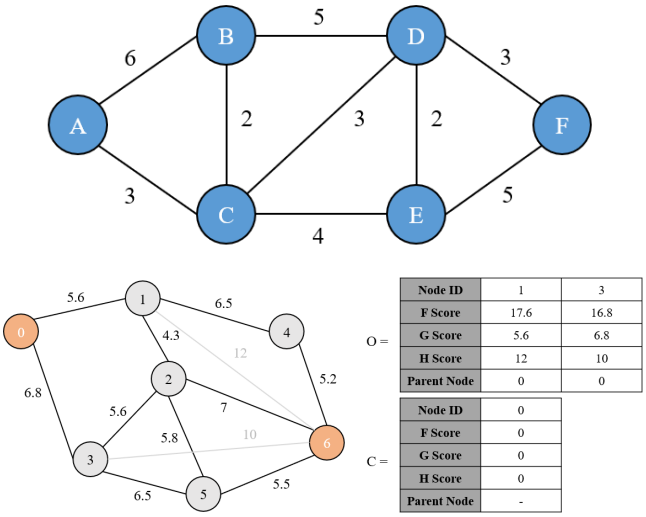

Fig. 4. Shortest path algorithm.

Additionally, the relevant prior work includes shortest path algorithms, such as Dijkstra [4] and A* [5], which derive paths between starting and goal points. This is illustrated in Fig. 4. However, they are limited in their ability to derive paths and waypoints for a search operation that visits all points within a region. In conclusion, prior algorithms for automatically generating routes for complex coastal search operations have limitations as shown in Table 1.

Table 1. Summary of limitations from prior work

Contents	Details
Not visiting all cells in a region	-Creating a path by selecting only a specific cell is unsuitable for missions that require navigating the entire area. -Selecting all cells to create a path results in excessive computation and time complexity.
Excessive time computational complexity	- Computes and compares the utility of all paths. -This creates a path through all cells in a complex port area, and comparing them requires a lot of computation and time.
A form that considers only points of departure and points of arrival	- Existing shortest path algorithms derive routes by considering only the departure and arrival points and are not suitable for area search operations.

In view of this, it is necessary to develop new technologies that overcome and improve the problems and limitations of existing technologies. Therefore, we propose an algorithmic system and method for automatically generating routes using reinforcement learning techniques for the efficiency of complex coastal search operations by unmanned underwater vehicles.

III. Materials and Methods

Concept Description: We combine reinforcement learning and A* algorithm to solve the existing problems. The new algorithm is expected to conduct the following: (1) visiting all cells in the region; (2) reducing the excessive time and computational and complexity; (3) making the path planning automatic and efficient. In order to improve the problems and limitations of existing techniques, we propose an algorithm for automatic generation of regional search path points by combining reinforcement learning and A* algorithm. First, the search area is divided into a graph with nodes of the size set by the user to reduce computation and memory, while improving the efficiency of the search. The core logic of the proposed algorithm is shown in Fig. 5.

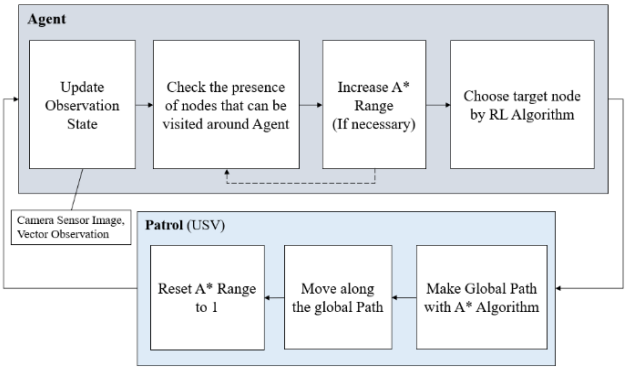

Fig. 5. Overall algorithm structure.

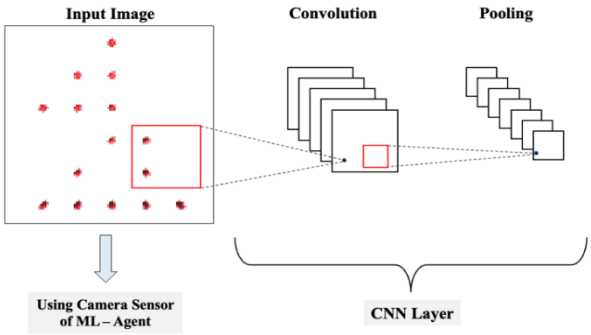

Fig. 6. Input image of agent.

First, the algorithm consists of two main components. The Agent, which collects state information and selects an action using a reinforcement learning algorithm, and the Patrol (USV), which actually moves using the selected action and the A* algorithm [6]. If we look at the internal logic of the Agent first, the Agent updates the image information based on the camera sensor and the vector observation to the observation state. At this time, the image information is delivered, as shown in Fig. 6, and the unexplored nodes implemented in the simulator that performs reinforcement learning training are captured and delivered. The captured image is delivered to the agent through the CNN Layer. This image information is used to recognize spatial information to the agent [7].

In addition to image information, the vector observation information is transmitted as shown in Table 2, including the current location of the agent, the number of times the agent has moved, the number of nodes visited, and whether each node has been visited. Each information is used to identify the current mission performance level and is related to the reward, so it leads to the efficient search path derivation.

Table 2. Vector observation

Vector State	Data Type
Current location of Agent (i)	int
Current location of Agent (j)	int
The number of times the agent moved	int
Number of nodes visited	int
Whether graph node is visited	Int (0,1)

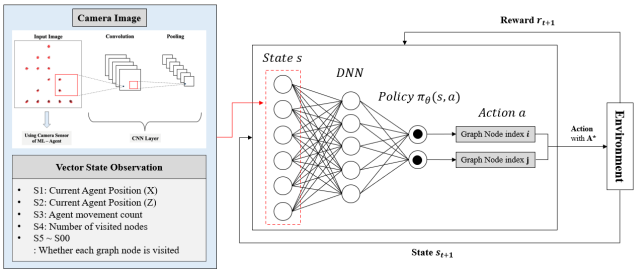

Fig. 7. Target node selection using reinforcement learning algorithm.

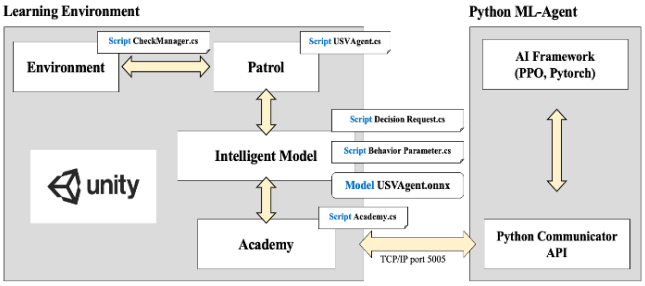

Fig. 8. Reinforcement learning algorithm training structure.

After updating its observations, the Agent sets a neighborhood range equal to the graph node index (i, j) corresponding to its current position plus A*-range, and checks if there are any unvisited nodes within the range. If no unvisited node exists, it checks the surrounding range by incrementing the value of A*-range by 1 until an unvisited node exists. If there are unvisited nodes in the neighborhood, select the target node as the action using a reinforcement learning algorithm for one of the unvisited nodes [8]. The reason for using A*-range to search for nearby unvisited nodes is to avoid inefficient travel. If we did not limit the range to nearby unvisited nodes, we would give the RL algorithm too many degrees of freedom. This would be difficult for the learning to converge [9]. The target node selection using Agent's reinforcement learning algorithm is shown in Fig. 7 and 8.

After receiving the aforementioned image and vector as observation state, the Agent derives an action using the policy. At this time, the action is used as the index (I, j) of the target node. Then, action (I, j) is passed to Patrol and used as a target node for the A* algorithm. Patrol derives the shortest path between the current location and the target node using the A* algorithm. And when the Patrol moves to the target node using the derived A* algorithm, the agent receives a reward. Reward is designed as shown in Table 3 and is designed to encourage the agent to complete the search in the shortest distance, while at the same time avoiding redundant trips. Here, "-" means negative awards, while "+" means positive awards given to the action.

Table 3. Reinforcement learning reward setting

Type of Reward	Value
The number of unvisited nodes	-
When agent visits node that already visited	-
Moving time	-
When agent visits unvisited node	+
When all nodes visited	++

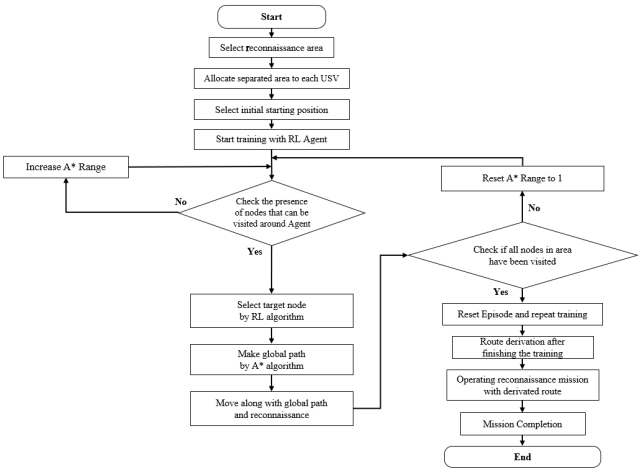

Fig. 9. Schematic of the unmanned surface vehicle's operational path point derivation and search process algorithm.

The first reward is a negative reward equal to the number of unvisited nodes currently remaining. This encourages exploration of the remaining unvisited nodes. The second is a negative reward for visiting a node that has already been visited, which encourages the agent to find the shortest path by reducing redundant visits. The third is a negative reward proportional to travel time, which encourages the agent to search as fast as possible. The fourth reward is a positive reward when an agent visits an unvisited node. This encourages visits to unvisited nodes. Finally, a large positive reward is given when all nodes have been visited to encourage all nodes to be visited. When the Patrol finishes travelling to the target node, the A*-range is reset to 1, and the process is repeated by updating the Agent's observation state again. Reinforcement learning training is performed until all nodes have been explored. Based on the core logic of the proposed algorithm, the path point derivation process for the USVs to perform search operations is shown in Fig. 9.

Initially, the search area is selected and assigned to each USV. After that, the starting position is determined and the training of the reinforcement learning agent is started, and the learning is performed using the A*-range, A* algorithm, and RL algorithm according to the core logic mentioned earlier [10]. After repeating the learning episodes, the search route points are derived, and the USV conducts search operations

using the derived route points. In the next part, we perform simulations using Unity3D and ML-Agent to demonstrate the effectiveness of the proposed algorithm. The effectiveness is demonstrated by comparing data such as the distance travelled by the agent before and after training, and the number of duplicated visits.

IV. Result and Discussion

In this section, we verify the proposed algorithm by simulating it in the Unity3D environment [11]. We implemented a simulation environment for reinforcement learning using Unity3D and ML-Agent. The selected search area is a coastal terrain in Taean, Chungcheongnam-do, South Korea, which has complex coastal boundaries. The simulation environment implemented in the Unity3D is shown in Fig. 10.

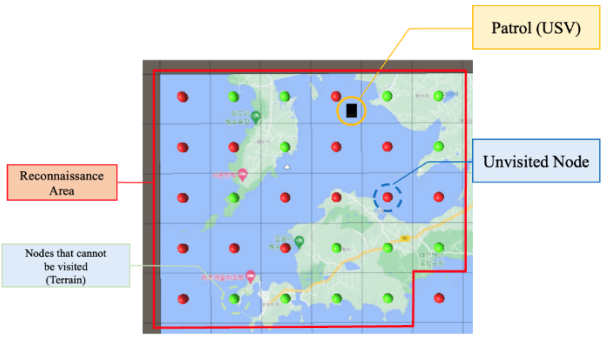

Fig. 10. Simulation environment setting.

Fig. 11. Patrol search process.

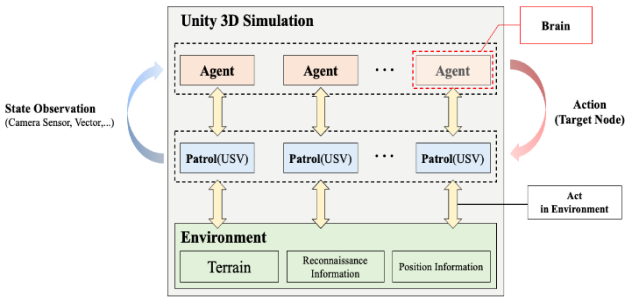

Fig. 12. Block diagram of the simulation environment inside Unity 3D.

The area outlined in red is the area to be searched. The search area consists of nodes of a certain size, as shown in the Fig.11. Red nodes indicate areas that can be searched, while green nodes indicate areas that cannot be searched by surface vessels due to terrain. The black square in the yellow circle represents a patrol that conducts search operations and visits red nodes to conduct search operations. At this time, the visited nodes are colored blue, as shown in Fig. 11.

In the simulation environment set up as above, Agent, Patrol, and Environment interact to perform reinforcement learning training, as shown in Fig. 12.

Patrol, implemented in Unity, interacts with the environment, and delivers image and vector information as state observations to the Agent, which acts as the brain. These states are used as input to the internal reinforcement learning algorithm. The algorithm used here is the PPO algorithm provided by ML-Agent. Once the Action (I, j) is derived, Partrol uses the A* algorithm to generate a global path, interact with the environment, and move around. It continues to learn by repeating this process. After setting up the simulation as above, the results of training with ML-Agent are illustrated in Fig. 13 and 14.

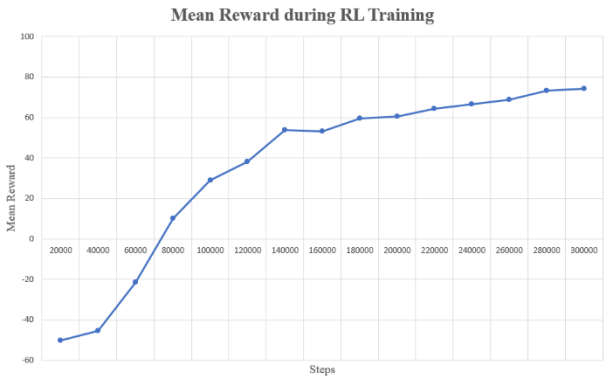

Fig. 13. Graph of reward average over training bins.

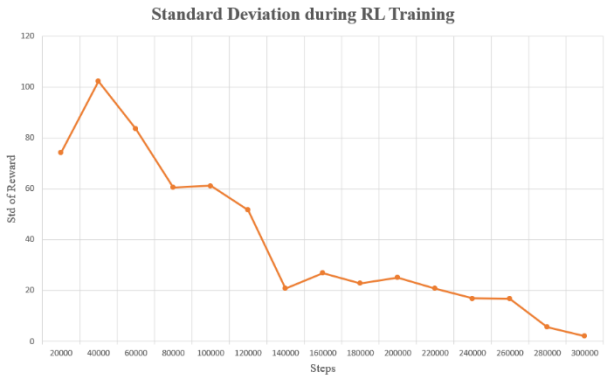

Fig. 14. Graph of reward standard deviation over training bins.

As shown in the figures above, the average reward received by the agent increases with each step, indicating that it is finding the optimal search path. The standard deviation of the reward also converges to zero, indicating that the agent is steadily finding the optimal path.

Table 4. Comparison of search efficiency before and after training

	Before Training	After Training
The number of movement by Agent	29	19
The number of repeated visitation	15	5
Spent time in simulation	145	95

Fig. 15 shows the search path obtained after training compared to before training. Before learning, the search path

is inefficient and the number of trip is high. However, after learning, it searches from the closest nodes one by one, and as a result, the optimal search path is automatically derived. Table 4 shows a result, comparing the search efficiency before and after training. As one can see, the number of trips and travel time of the agent have been reduced, and the number of duplicate visits has been dramatically reduced. This shows that we have derived an efficient search path.

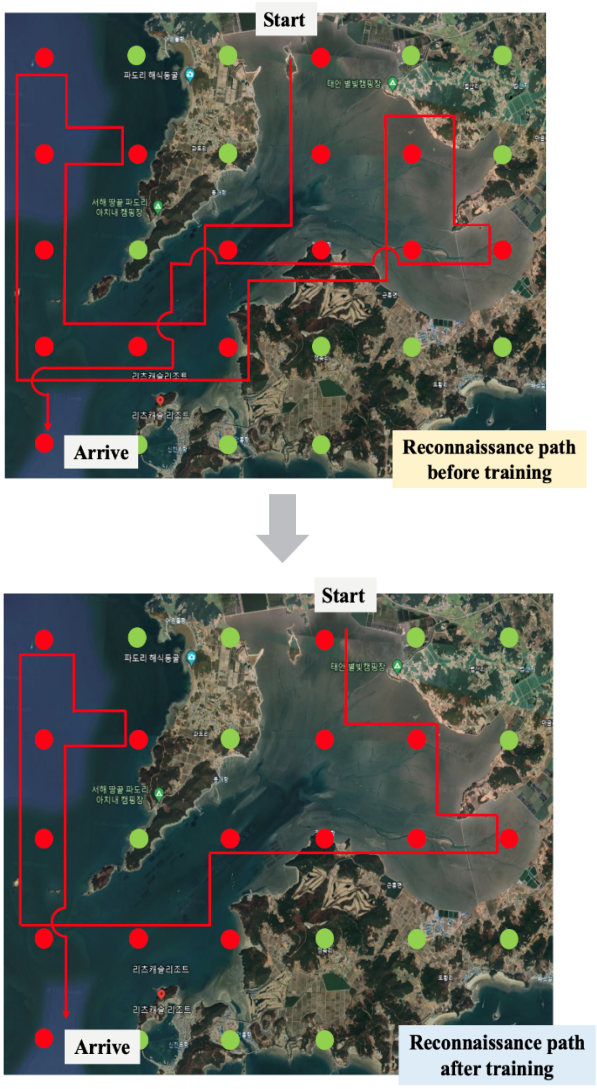

Fig. 15. Comparison of search paths before and after training.

V. Conclusion

The proposed algorithm was able to address limitations in deriving the route and route points of search operations visiting all points in the region of existing shortest-path algorithms such as Dijkstra and A*. First, when visiting all nodes, a large amount of Reward was given so that the agent (unmanned award) could visit all nodes, and the problem of not visiting all cells in the region was solved. In addition, rather than creating all possible paths and improving efficiency, reinforcement learning capable of learning through fast simulation was applied to find the path directly and converge to the optimal value by the concept of Trial & Error. Finally, since the shortest path algorithm only considers the starting point and the arrival point, as mentioned above, one episode is designed to end only when all nodes are searched, and the problems of the existing algorithm are solved by repeatedly applying the A* algorithm, one of the shortest path algorithms, and moving along the derived path.

In other words, the proposed algorithm, which combines reinforcement learning and the A* algorithm to improve the problems and limitations of existing technologies, was able to automatically derive path points that can be used for search operations rather than simply deriving shortest path points from one point to one. This is significant because it not only solved a problem that was difficult to derive previously, but also solved a practical problem using reinforcement learning. In addition, the algorithm for automatically generating search route points can be used to reduce the time and labor costs of conducting operations. By reducing the process of entering search path points one by one, the time can be reduced, and labor costs can be reduced in proportion to the reduced time. Furthermore, if a simulator is built that can scale this algorithm, it will be possible to run simulations in all areas of operation. This would allow for a large number of unmanned surface vessels to be searched at once, resulting in significant productivity gains and cost savings.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Authors conducted the research; analyzed the data; and wrote the paper. All authors had approved the final version.

Funding

This research was funded by Ajou Research Fund.

References

[1] S. Shik, "Current state of unmanned surface vehicle (USV) research at home and abroad," *Journal of the Society of Naval Architects of Korea*, vol. 55 no. 3, pp. 9–14, 2018

[2] Y. Yang, Y. Mao, R. Xie, Y. Hu, and Y. Nan, "A novel optimal route planning algorithm for searching on the sea," Cambridge University, January 2021.

[3] I. Hyeong and J. Y. Jae, "How the Army's 'unmanned watercraft' will be used with unmanned systems technology," 2020.

[4] D. Verma, D. Messon, M. Rastogi, and A. Singh, "Comparative study of various approaches of dijkstra algorithm," in *Proc. 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Feburary 2021

[5] Y. B. Li, Z. Xi. Wang and S. Y. Zhang, "Path planning of robots based on an improved a-star algorithm," in *Proc. 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, December 2022

[6] L. P. Kaelbling, A. Y. Ng, and S. J. Russell, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, 1996.

[7] C. Che, J. Huang, C. Y. Pan, and X. S. Yuan, "Military Image Scene Recognition Based on CNN and Semantic Information," in *Proc. 2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, September 2018

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[9] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *Foundations and Trends in Robotics*, 2018.

[10] M. Jaderberg *et al.*, "Reinforcement learning with unsupervised auxiliary tasks," in Proc. *ICLR*, 2017.

[11] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," arXiv, 2018.