The Hidden Object Searching Method for Distributed Autonomous Robotic Systems

Han-Ul Yoon, Dong-Hoon Lee, and Kwee-Bo Sim

School of Electrical and Electronics Engineering, Chung-Ang University, 221 Heukseok-Dong, Dongjak-Gu, Seoul, 156-756 Korea
(Tel: +82-2-820-5319; E-mail: huyoon@wm.cau.ac.kr, kbsim@cau.ac.kr)

Abstract: In this paper, we present the strategy of object search for distributed autonomous robotic systems (DARS). The DARS are the systems that consist of multiple autonomous robotic agents to whom required functions are distributed. For instance, the agents should recognize their surrounding at where they are located and generate some rules to act upon by themselves. In this paper, we introduce the strategy for multiple DARS robots to search a hidden object at the unknown area. First, we present an area-based action making process to determine the direction change of the robots during their maneuvers. Second, we also present Q learning adaptation to enhance the area-based action making process. Third, we introduce the coordinate system to represent a robot's current location. In the end of this paper, we show experimental results using hexagon-based Q learning to find the hidden object.

Keywords: DARS, area-based action making process, Q learning, object recognition

1. INTRODUCTION

Recently, the robots are replacing human's work in dangerous fields, such as rescue jobs at fire-destroyed buildings or at gas-contaminated sites; information retrieval from deep seas or from space; and weather analysis at extremely cold areas like Antarctica. Multiple robots are especially needed to penetrate into hard-to-access areas, such as underground ant nests, to collect reliable and solid data. They can send data through cooperation and communication and can make independent decisions to act.

Distributed autonomous robotic systems (DARS) are systems with multiple autonomous robotic agents, assigned with required functions. The most unique and important feature of DARS is that each system is a distributed system composed of multiple agents or robots [1]. With this feature, DARS can be applied for a wide range of application. DARS is now applied to multi-robot behavior, distributed control, coordinated control, cooperative operation, etc. DARS has received much attention since it can offer a new way of controlling multiple agents more flexibly and robustly. Parker proposed the heuristics approach algorithm for multiple robots and applied it to cleaning tasks [2]. Ogasawara used this system for several robots to transport a large object [3]. In this paper, we propose an area-based action making (ABAM) process to control multiple robots against collision and guide individual robot to search through its own trajectory.

Reinforcement learning allows an agent to actively determine an action policy based on explorations of its environment. During exploration of an uncertain state space with reward, an agent can learn what to do by continuous tracking of its state history and appropriately propagating rewards through the state space [4]. In our research, we focused on Q-learning as a reinforcement learning technique. Because Q-learning is a simple way to solve Markovian action problems with incomplete information and on the basis of the action-value function Q that maps state-action pairs to expected returns [5]. In addition to this simplicity, Q-learning can adopt to the real world situation. For example, the state space can be matched with the physical space of the real world. An action also can be regarded as physical robot movement. In this paper, we propose the hexagon-based Q-learning to enhance the area-based action making process so that the learning process can better adapt to real world situations.

The organization of this paper is as follows. In the next chapter, the area-based action making process is introduced. In chapter 3, hexagon-based Q learning adaptation is presented. The coordinate system to represent robot's current location in the hexagon-based Q learning is explained in chapter 4. In chapter 5, experimental results of the hidden object search using hexagon-based Q learning with multiple robots are presented. Chapter 6 is the conclusion of the paper.

2. AREA-BASED ACTION MAKING PROCESS

Area-based action making (ABAM) process is a process that determines the next action of a robot. The reason why this process is referred to ABAM is that a robot recognizes surrounding not by distances, from itself to obstacle, but by areas around itself. The key idea of the ABAM process is to reduce the uncertainty of its surrounding. It is similar with the behavior-based direction change, to control the robots [5][6]. The robots recognize the shape of its surrounding, then take an action (turn and move forward) to where the widest space will be guaranteed. Consequently, each robot can avoid an obstacle and collision with other robots. Figure 1 depicts the different actions taken by distance-based action making (DBAM) and by ABAM in the same situation [7]. Our small mobile robot has the six emitter-detector infrared sensor pairs, which are placed at an angle of 60 degrees with one another to cover 360 degrees.

The advantage of ABAM is illustrated by the following example. Figure 2 presents the result of each action making process by DBAM and ABAM. In both case, the robot is surrounded by 4-obstacles. By DBAM, the robot will be confused because it perceives that there is no obstacle in the southeast direction, and then it will try to keep tracking to the southeast. Finally, it will get stuck between two obstacles. This scenario is shown in the left picture in Fig. 2. By ABAM, however, the robot will calculate the areas of its surrounding, and then it will recognize that an action to the northeast will guarantee the widest space. Therefore, the robot will change its direction to the northeast. This scenario is presented in the right picture in Fig. 2.

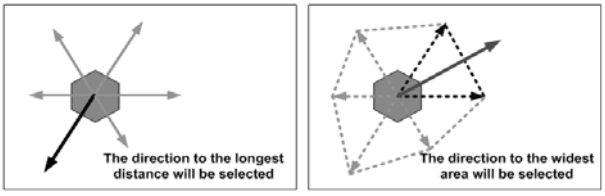

Fig. 1 The difference of action results by DBAM and by ABAM.

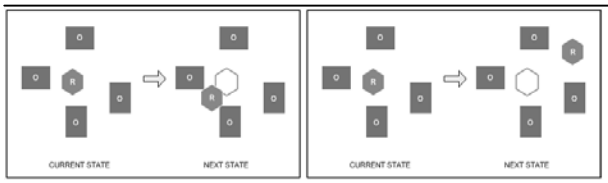

Fig. 2 An illustrative example of robot maneuvers by DBAM (left) and by ABAM(right).

In addition to the obstacle avoidance, ABAM also make the robots to search their own space [8]. This feature is advantageous when 2 or 3 robots meet at the same place. When they face each other, each robot will try to find more wide space. Consequently, the robot will change its direction to avoid the other robots and start to search in its own space again.

3. HEXAGON−BASED Q LEARNING

Q learning is a well-known algorithm for reinforcement learning. It leads the agent to acquire optimal control strategies from delayed rewards, even when the agent has no prior knowledge of the effects of its actions on the environment [9][10]. Q learning algorithm is presented in Table 1, where s is a possible state, a is a possible action, r indicates the immediate reward value, and γ is the discount factor.

Table 1 Q learning algorithm.

For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero

Observe the current state s

Do forever

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \qquad (1)$$

- $s \leftarrow s'$

Figure 3 is an illustrative example to explain Q-learning algorithm more clearly. Each grid square presents the possible states. The 'R' stands for a robot or agent. The values upon the arrows are relevant $\hat{Q}$ values with the state transition. For example, the value $\hat{Q}(s_1, a_{right}) = 72$, where a right refers to the action that moves R to its right [9].

If the robot takes the action to the right, the value will be updated for this entry where $r = 0$, $\gamma = 0.9$ are predetermined values. The formula is presented below.

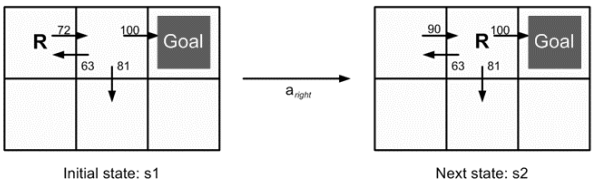

Fig. 3 An illustrative example of Q learning.

$$\begin{aligned} \hat{Q}(s,a) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \\ &\leftarrow 0 + 0.9 \max\{63, 81, 100\} \\ &\leftarrow 90 \end{aligned} \qquad (2)$$

The Q learning for our robot system was adapted to enhance the ABAM process. The adaptation can be performed with a simple and easy modification, named hexagon-based Q learning. Figure 4 is an illustrative example of hexagon-based Q learning. In Fig. 4, intuitively, we know that the only thing that was changed is the shape of state space. We changed the shape of the space, from a square to a hexagon, so that the robot can recognize its surrounding by 6-areas. According to this adaptation, the robot takes an action to 6-direction and has 6-table entry $\hat{Q}$ value. In the left of Fig. 4, the robot is in the initial state. Now, if the robot decides that +60 degree guarantee the widest space after calculation of its 6-areas of surrounding, the action of the robot would be $a_{+60°}$. After the action is taken, if *Area6'* is the widest area, the value of $\hat{Q}(s_1, a_{+60°})$ will be updated by the formula (1) in the Q learning algorithm as

$$\begin{aligned} \hat{Q}(s_1, a_{+60}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow 0 + \gamma \max\{Area1', Area2', \cdots, Area6'\} \\ &\leftarrow \gamma Area6' \end{aligned} \qquad (3)$$

where 0 is the predetermined immediate reward. After the movement from the initial state to the 1st next state, immediate reward becomes the difference between *'the sum of total area before action is taken'* and '*the sum of total area after action is taken.'* Thus,

$$r = \sum_{j=1}^{6} Area'_j - \sum_{i=1}^{6} Area_i \qquad (4)$$

where $Area_i \in s_1$ and $Area_j \in s_2$, respectively.

Ultimately, the robot can determine its trajectory by learning this $\hat{Q}$ value. In the real world experiment, however, battery consumption is a problem. If the robot has to perform infinite iterations to complete task, total system will fail. Therefore, a system must be set up to cancel the former action and move back to the earlier state, when the former action causes any bad reward or result. However, to determine *"how many states would be canceled?"* is critical problem that needs another heuristic approach to solve it. For out experiment, we just set the robot to retreat one-states backward against any misjudged action. The hexagon-based Q-learning algorithm is presented in Table 2.

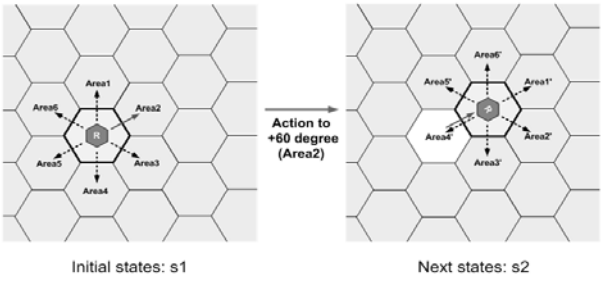

Fig. 4 An illustrative example of Q learning.

Table 2 Hexagon-based Q learning algorithm.

For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero

Calculate each 6-areas at the current state s

Do until task is completed.

- Take an action a to the widest area
- Receive immediate reward r
- Observe the new state s'

If $\hat{Q}(s',a')$ is greater or equal than $\hat{Q}(s,a)$

- Update the table entry for $\hat{Q}(s,a)$
- $s \leftarrow s'$

If $\hat{Q}(s',a')$ is too less than $\hat{Q}(s,a)$

- Move back to the previous state
- $s \leftarrow s$

4. COORDINATE SYSTEM TO REPRESENT ROBOT'S LOCATION IN THE HEXAGON−BASED Q LEARNING

Robot has to identify its location in the environment to work cooperatively, or to store the object's location what it found. With an advantage of hexagon-based Q learning, we can formalize a simplified X-Y coordinate system to represent the robot's location.

Figure 5 shows every possible movement (action) that can be taken by the robot at some state. Let's denote the longest side of a right-angled triangle with $2R$. By the geometrical analysis, the base side is $\sqrt{3}R$ and the height is R. Therefore, x can be $0, +\sqrt{3}R, -\sqrt{3}R$ in the case of +0, +60, −60 degrees, and y can be $+2R, +R, -R$ in the case of +0, ± 60, ± 120 degrees respectively, where $x \in X$, $y \in Y$. Move to the ± 180 degree (turn around) means the state cancellation caused by a critical result. Table 3, involved look-up-table inside the program, shows the relation between x, y value and $\theta_{S[N]}$ by direction was taken at the N^{th} state.

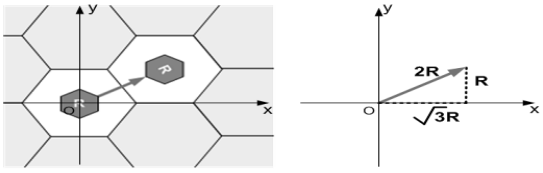

Fig. 5 Geometrical vector analysis for robot movement.

Table 3 Look-up-table to determine $disp\_x, disp\_y$ by the $\theta_{S[N]}$.

$\theta_{S[N]}$	$displacement\_x$	$displacement\_x$
0	0	2R
+60	$\sqrt{3}R$	R
−60	$-\sqrt{3}R$	R
+120	$\sqrt{3}R$	−R
−120	$-\sqrt{3}R$	−R
180	0	−2R

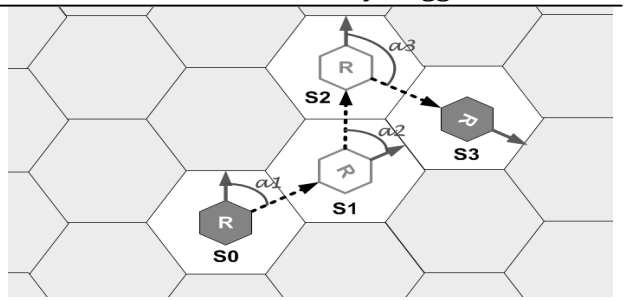

Fig. 6 An illustrative example of current location calculation.

The current position can be calculated by vector addition. Figure 6 shows an illustrative example. The angle, between Y-axis and a robot's forward direction, starts from Y-axis and clockwise direction is positive. At the initial state, *S[0]*, $\theta_{S[0]}$ is 0 degree. It means that the robot is heading on the northern-side in the figure. If the robot takes an action to the +60 degree, $\theta_{S[1]}$ will be +60 degree and the robot is heading on +60 degree direction. If the robot takes an action to the –60 degree, it will be heading on the north again. Consequently, $\theta_{S[N]}$ can be generated as follows:

$$\theta_{S[N]} = \sum_{k=0}^{N} \alpha_k = \begin{cases} \alpha_0 = 0, & \\ \alpha_1, & \\ \alpha_k + \theta_{S[k-1]} & \text{if } 1 < k \le N \end{cases} \quad (5)$$

where α is the angle which presented in Fig. 6. The current position is updated by following procedure.

Calculate $\theta_{S[N]}$ at each state

Determine $disp\_x, disp\_y$ values by look-up-table

Then,

- $x \leftarrow x + disp\_x$
- $y \leftarrow y + dist\_y$ (6)

For example, $\theta_{S[1]} = \alpha_1 = +60$ at the 1st state in Fig. 6. Therefore, the position of the robot, $P_{S[1]}$, is $(\sqrt{3}R, R)$. At the 2nd state, $\theta_{S[2]} = \alpha_2 + \theta_{S[1]} = +60 + (-60) = 0$ and $P_{S[2]}$ is $(\sqrt{3}R+0, R+2R) = (\sqrt{3}R, 3R)$ respectively. Finally, $\theta_{S[3]} = \alpha_3 + \theta_{S[2]} = 0 + 120 = 120$ and $P_{S[3]}$ is $(\sqrt{3}R + \sqrt{3}R, 3R + (-R)) = (2\sqrt{3}R, 2R)$ at the 3rd state.

When the task is completed, this value can be used to represent the object location. If one robot found the object, the others could get around the object by receiving the location via communication network.

5. EXPERIMENTAL RESULTS

We performed experiments by using three different control methods: random search, ABAM only, and ABAM with Hexagon-based Q-learning.

The task of the robots is a follows: "Find the hidden object while tracking through an unknown hallway". We set up the color of the object as green. The object was a stationary. It was a located at a hidden place near the obstacle. The 5-robots, which try to search the object, recognize the object by the object's color and shape. The 5-robots will decide whether they have finished the task by detecting the object after each action is taken.

First, we used the random search control method to find the hidden object. The main controller generated a random number and decided the next action corresponding to this number. Random search is not so strong method to control the robot efficiently. Therefore, random did not perform well. Moreover, it is very time and power consuming in the real world situation. The result showed that random search is a horrible method to adopt to a real robot system. In Fig. 7, the white arrow points out the object (same in Fig. 8 and Fig. 9). During random search, even though the robots are within near vicinity, the robots failed to find the object.

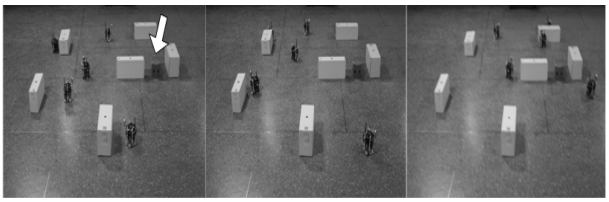

Fig. 7 Five robots are searching the object using random search.

Second, we applied ABAM to the robots. With the feature of ABAM, the robots sense their environment by 6-infrared sensors and calculate 6-area with these values. When the calculation is done, each robot tries to move to where the widest area will be guaranteed. In our 2nd experiment, after the robots started to move, each robot spread out into the environment. Consequently, the ABAM performed really better than random search. Figure 8 shows the robots on processing. The task completion took around 10 minutes.

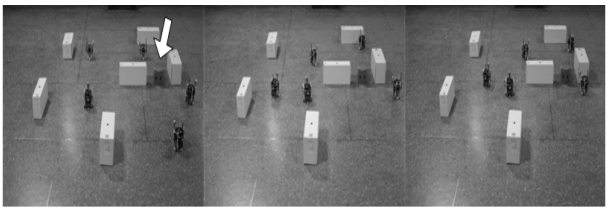

Fig. 8 Five robots are searching the object using area-based action making (ABAM).

Finally, we adopted the hexagon-based Q-learning to ABAM as a modified control method. This method allowed the robots to reduce the probability of wrong judgment and compensated wrong judgment by reinforcement learning. By using the hexagon-based Q-learning adaptation to ABAM, the task completion time reduced to around 6 minutes at the first time. As the hexagon-based Q learning is reinforcement learning, the time reached out 4 minutes and 20 seconds. The search with hexagon-based Q-learning is presented in Fig. 9.

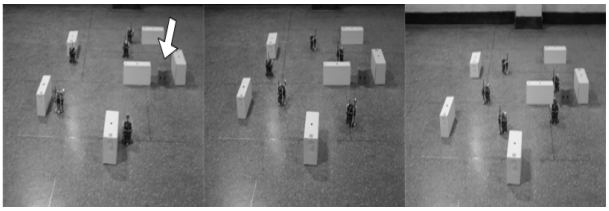

Fig. 9 Five robots are searching the object using hexagon-based Q learning.

6. CONCULSION AND FUTURE WORKS

In this paper, we presented the area-based action making process and hexagon-based Q-learning to search the object, hidden in unknown space, for 5 of our self-made small mobile robots. The experimental results from the application of the three different control methods in the same environmental situations were presented. The area-based action making process and hexagon-based Q-learning can be a new way for robot to search an object in unknown space. This algorithm also makes the agents avoid obstacles during their search.

In our research, first, we need to clarify the problem of accessing to the object. This means that if multiple robots are to carry out a task such as object transporting or block stacking, the robots need to recognize the object then approach to it. Therefore, we need to develop the robust accessing algorithm. Naturally, some grippers need to be attached to both sides of the robot. Second, our robot systems should be improved so that the main part and the sub-parts adhere more strongly. In addition, stronger complex algorithms such as Bayesian learning or TD(λ) method should be adapted. Third, a self-organizing Bluetooth communication network should be built so that robots can communicate with each other robustly even if one or more robots are lost. Finally, the total system should be refined.

ACKNOWLEDGEMENTS

This research was supported by the project of Developing SIC (Super Intelligent Chip) and its Applications under the program of Next generation technologies in 2000 of Ministry of Commerce, Industry and Energy.

REFERENCES

[1] L. Parker, “Adaptive action for cooperative agent teams,” *Proc. of 2nd Int. Conf. on simulation of Adaptive Behavior*, pp 442-450, 1992.

[2] G. Ogasawara, T. Omata, and T. Sato, “Multiple movers using distributed, decision-theoretic control,” *Proc. of Japan-USA Symp. On flexible Automation*, Vol. 1, pp. 623-630, 1992.

[3] D. Ballard, *An Introduction to Natural Computation*, The MIT Press, Cambridge, 1997.

[4] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, New Jersey, 1997

[5] W. Ashley and T. Balch, “Value-based observation with robot teams (VBORT) using probabilistic techniques,” *Proc. of Int. Conf. on Advanced Robotics*, 2003.

[6] W. Ashley and T. Balch, “Value-based observation with robot teams (VBORT) for dynamic targets,” *Proc. of Int. Conf. on Intelligent Robots and Systems*, 2003.

[7] J. B. Park, B. H. Lee, and M.S. Kim, “Remote control of a mobile robot using distance-based reflective force,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, Vol. 3, 3415-3420, 2003.

[8] P. Ögren and N. E. Leonard, “Obstacle avoidance in formation,” *Proc. of IEEE Int. Conf. on Robotics and Automation*, Vol. 2, pp. 2492-2497, 2003.

[9] T. Mitchell, *Machine Learning*, McGraw-Hill, Singapore, 1997.

[10] C. Clausen and H. Wechsler, “Quad-Q-learning,” *IEEE Trans. On Neural Network*, Vol. 11, pp. 279-294, 2000.