An Advanced Motion Planning Algorithm for the Multi Transportation Mobile Robot

Nguyen Van Vui¹ and Yong-Tae Kim²

¹Smart Device Solutions Co., Ltd, Ho Chi Minh City, Vietnam

²Department of Electrical, Electronic and Control Engineering, Smart Logistics Technology Institute, Hankyong National University, Anseong, Korea

Ijfis

Abstract

In this paper, an advanced motion planning algorithm for the multi transportation mobile robots is presented. The proposed algorithm is developed based on the D* Lite path planning algorithm to avoid dynamic obstacles and a potential collision between the transportation robots. In the multi-robot environment, an operation server manages all planned paths of the robots by using the condition of the robot and obstacle environment. The server updates continuously database of all robots in a real-time. The proposed algorithm checks and predicts a safe point and then decides the lower and higher priority robot. The lower priority robot will be re-planned the path to avoid a potential collision with higher priority robot. The experimental results show that the server can quickly update the path of two robots in the real-time. It can also check and re-plan correctly the path by using D* Lite algorithm to avoid a potential collision of two robots.

Keywords: Motion planning algorithm, D* Lite algorithm, Path planning algorithm, Collision avoidance, Transportation robot, Multi-robot

1. Introduction

Logistics companies are putting a lot of effort into reducing logistics costs and streamlining product supply with various IT and new technologies. Especially, the application of automation equipments, unmanned transportation vehicles, transfer robots and drone is increasing in the distribution center due to the increase of labor costs. Logistics robots are generally used in the semiconductor and automobile fields because of the high initial investment cost in general. However, logistics robots have recently been applied to logistics centers [1, 2]. As the demand of the transportation mobile robot is increasing in the logistic center, a large number of robot can be applied to carry and transfer packages of goods on a map of a floor [3, 4].

The navigation of the robots applied to a logistics center can be largely divided into running along predetermined routes or carrying goods from the origin to the destination as autonomous travel without designated route. Amazon's KIVA system uses QR code on the bottom surface to display each absolute position and transports the logistics rack, and efficiently handles small quantity items of various kinds. A QR code-based navigation algorithm of the logistics robot was proposed to transport the goods shelves to the packaging workshop [4, 5].

Due to a common map environment of the logistic center, the transportation robot needs to know about the position and the planned paths to avoid a potential collision with other robots

Received: May 1, 2019
 Revised : May 20, 2019
 Accepted: Jun. 18, 2019

Correspondence to: Yong-Tae Kim
 (ytkim@hknu.ac.kr)
 ©The Korean Institute of Intelligent Systems

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

and freight racks. Besides that, a management server needs to get the navigation conditions and the optimal paths to manage all robots. In order to solve this problem, the path planning algorithm is used to plan and re-plan a path of each of robot on a server [6–8].

In the paper, we propose a QR code alignment method and an advanced motion planning method of the transportation robots in the logistic environment. The QR code alignment method controls the robot to locate on the QR code using QR code's angle from the robot and calculates the error zone to help the robot to turn left or right exactly. We also propose an advanced motion planning algorithm to generate an optimal path in the multi-robot environment and avoid a potential collision between the robots in the navigation status. The collision avoidance algorithm is designed to navigate the robots without collision using decision of higher and lower priority robot, and D* Lite algorithm. The advanced motion planning algorithm helps the operation server to manage all real positions, planned points and parameters of paths of the robots by continuously updating a database in the real time. The proposed methods are verified by using the navigation and collision avoidance experiments.

This paper is organized as follows. Section 2 explains about the system configuration of multi transportation mobile robot in a Wi-Fi networking environment and the grid map for navigation of mobile robots. Section 3 explains an advanced motion planning algorithm with the D* Lite path planning and the collision avoidance algorithm in detail. Section 4 shows the experimental results of QR code alignment methodology, the navigation results of one robot on the static obstacle and the grid map of the static environment, and the experimental results of the proposed motion planning algorithm applied on two transportation robots. Section 5 provides the conclusion of the paper.

2. Overall Configuration of Multi-Robot System

2.1 Configuration of Multi Transportation Robot

In the multi-robot environment, the management server needs to generate an optimal path of robot by using motion planning algorithms based on the robot status, obstacles and the given tasks. The transportation robots must carry the freight rack and pallets on the floor. So, the robots need to be designed to communicate with the server via a wireless networking. The multi-robot control configuration is proposed as shown in Figure 1.

The multi-robot system is designed to operate based on 3 main parts as shown in Figure 2 including path planning on the

Figure 1. The multi-robot control configuration in the Wi-Fi networking [5].

Figure 2. Path planning, localization, navigation part of the multi-robot system.

server, localization and navigation on each robot. In the path planning part, the management server is a master to manage and control all the paths of all robots. It is used to solve the question: “where should the transportation robot move?”. The path planning system is a brain of the robot in the multi-robot system. In the localization part, a Raspbian environment of a Raspberry Pi 3 board is used to decode a coordinate of the QR code and allow the robot to track the line in the grid map, and move the robots to the goal position on the floor. The robot not only communicate data with the server via a TCP/IP protocol but also communicate and control the navigation of each robot via an RS-232 protocol in the real time.

In the navigation part, the RTOS environment is applied on the rotational robot and the linear robot to control two wheels

Figure 3. Configuration of the grid map with 10 × 10 in scale.

and the lifter. It receives and analyses the commands from the localization part by using the RS-232 protocol and controls motors of the robot. This navigation part is used to solve the question: “What is the robot doing?”. So it is also called the body of each robot.

2.2 Grid Map Configuration

A grid map with n row × m column in scale for the path planning part of the server is configured including: nodes, robot, goal, obstacle, mobile rack or storage shelf as shown in Figure 3. Each position of the nodes matches with a coordinate of a QR code on the map of the floor. The map of coordinates can be changed by modifying an Excel file with a configuration data.

2.3 Navigation Rule and Data

The value of nodes and the current angle of robot are used to calculate and convert to a format for the navigation of the robot. We use character formats for the navigation: ‘F’ is forward, ‘B’ is backward, ‘R’ is turn right, ‘L’ is turn left, ‘E’ is stop, ‘U’ is lift up, ‘D’ is lift down. The formats help the localization part to control and navigate the robot correctly when the robot is moving on the map. After a path is planned by using the planning algorithm, a moving sequence is generated including many nodes from a start point to a goal point such as: $N_i, N_{i+1}, \dots, N_n$. A next node N_{i+1} and a current node N_i become 2 inputs of the following equation:

$$\Delta N_j = N_{i+1} - N_i. \quad (1)$$

Combination of ΔN_j of Eq. (1) and a current angle of the robot θ_j are used to set a rule for the navigation data as shown in

Table 1. Navigation rule of the robot

ΔN_j	θ_j			
	-90°	+90°	0°	+180°
-1	F	B	L	R
+1	B	F	R	L
-Row	R	L	F	B
+Row	L	R	B	F

Table 1.

Control mode of the lifter and control type of the fork-type lifter are decided using data of the map. The server uses all of data from the planned path, navigation data, mode of the lifter and type of the fork-type lifter to convert to simple data points. The format data includes: (X_i, Y_i, D_i, M_i) where X_i, Y_i are the coordinates of QR code at point i , D_i is the result of navigation data to navigate the robot at point i , M_i is the mode to control the lifter at point i .

2.4 The Structure of Common Database

The advanced motion planning algorithm checks the collision of multi robot by using a common database on the server. The server writes and reads continuously the database in the real-time. The structure of database should be formatted as below:

*	IP Address	:	Sxx	Pxx	dxx	lxx	Lxx	Gxx
---	------------	---	-----	-----	-----	-----	-----	-----

where IP address is each robot has a known IP address in the Wi-Fi networking. The formatted IP address is 192.168.c.d with value of c and d from 0 to 255. Sxx and Pxx are value of node of the start point and the next planned point. S and P are a formatted character and xx is value. dxx is value of real distance. d is character and xx is value. If the robot moving in the line tracking statement between 2 points, value of distance will be received from the real robot. Value of real distance is equal to 0 when the robot standing on QR code. lxx is length value of the path. l is character and xx is value. After the path is planned, the length value is calculated from start point to goal point to define a shorter path after compared with another robot’s path. Lxx and Gxx are node value of the mobile rack or storage shelf and the goal point. L and G are character and xx is value.

The server reads data continuously from the common database and defines the robot via IP address. Value of the start point Sxx, the next planned point Pxx, real distance dxx, length of path lxx and the goal point Gxx are used to predict and calculate the collision avoidance. Value of Lxx helps the server manage

Figure 4. The error after the robot turns on the QR code [4].

all positions of the mobile racks or storage shelves on the map of the floor.

3. Advanced Motion Planning Algorithm

3.1 QR Code Alignment Methodology

When the robot is moving on the floor map by tracking the line from the start point to the next planned point [9], errors from the line tracking make error between the robot's frame and the QR code's image after the robot meet the next planned point. The error will not impact to the navigation of the robot when the next planned navigation is forward or backward. Unfortunately, when the next planned navigation is turn left or turn right, the error easily makes image of the QR code be lost after the robot is turning as show in Figure 4 [4].

The robot start turning at a position as shown in Figure 4(a) with $+90^\circ$ or -90° of angle. Figure 4(b)-4(d) show that the camera continuously reads angle of the QR code when the robot is turning to define the angle of goal. Due to the error occurred at the start position, the image of the QR code is lost after the robot continuously turns as show in Figure 4(e) and 4(f). At this time, the angle of goal is not defined correctly, so the robot cannot stop at the correct position on the QR code to move continue on its path. In order to solve this problem, a good position of robot when the robot is standing on the QR code is defined as a center or in a safe zone of the image's frame. The size of image's frame in the configuration of camera is 320×240 in the pixels to capture the area of $7.3 \text{ cm} \times 6 \text{ cm}$ and the size of QR code is $20 \text{ mm} \times 20 \text{ mm}$ in its scale as shown in Figure 5.

The safe zone (SZ) and unsafe zone (UZ) are established

Figure 5. Frame's image of QR code in the Descartes coordinate [4].

by 2 points (X_1, Y_1) and (X_2, Y_2) as shown in Figure 5. The UZ is divided by 2 small zones: left unsafe zone (LUZ) and right unsafe zone (RUZ). There are three marker points in a QR code such as: $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$ as shown in Figure 5. Let's consider the marker point $A(x_A, y_A)$. In the case, the point A is detected in the SZ, the robot is able to turning without any error. Conversely, if the point A is detected in the UZ, then a QR code alignment methodology is applied to control the robot forward or backward with a small velocity to modify the error. The angle data of the QR code is sampled continuously. The angle data of the QR code at time t_i is α_i . $\Delta\alpha$ is the average value of the QR code angle as following equation,

$$\Delta\alpha = \frac{\sum_{i=0}^n \alpha_i}{n}, \quad (2)$$

where n is the sampling time.

The value of sampling time depends on the turning velocity of the robot. The position of the robot at time t_n is predicted correctly with the angle of goal. The value of angle α_0 is defined at time t_0 when the camera starts losing the image. When $\Delta\alpha = \alpha_0$, the robot stops and does a forward or backward action based on the start position of the marker point A and the turning command of the robot [4] (Table 2). Conversely, if $\Delta\alpha \neq \alpha_0$, it means that the image is not lost and the robot is turning in fine condition.

Table 2. Rule of QR code alignment

Command	$A(x_A, y_A)$		
	LUZ	SZ	RUZ
Turn Left	B	None	F
Turn Right	F	None	B

Figure 6. The path planned by using D* Lite planning algorithm.

3.2 D* Lite Planning Algorithm

When the robots are moving on the floor map in the dynamic environment, the obstacles can appear accidentally and a potential collision is easy to occur between the robots together. So, the robot needs to be planned or re-planned with a shortest path in a fast time by using the path planning algorithm [10–12]. D* Lite planning algorithm as shown in Figure 6 can solve the goal-directed navigation problems in unknown terrain. This planning algorithm computes the values of the inconsistent nodes when a node changes. The algorithm uses two definitions of nodes that are successor and predecessor. A successor is defined as node ID_2 and node ID_1 is a predecessor if there is a directed edge from node ID_1 to node ID_2 in the graph.

A node is an inconsistent node when it updates all of its neighbors and itself again [11]. It is defined as the following equation:

$$g(u) \neq rhs(u), \quad (3)$$

where $g(u)$ is the estimated distance from the goal vertex to the current vertex (u), rhs is value of a node and is computed based on the g values of its successors in the graph and the transition costs of the edges to the successors as following equation:

$$rhs(u) = \min_{u' \in Succ(u)} (c(u, u') + g(u')), \quad (4)$$

where $u' \in Succ(u)$ means u is a predecessor and u' is a successor, $0 < c(u, u') < \infty$ is the cost of moving from vertex u to vertex u' .

A shortest path from a start to a goal is searched by using D* Lite algorithm based on the current vertex u and u' until u_{goal} is reached as shown in Figure 7 [8].

Figure 7. D* Lite planning algorithm searching a shortest path.

Figure 8. The paths planned (a) before and (b) after matching a static obstacle.

3.3 Multi-Robot Navigation Algorithm

A multi-robot navigation algorithm is designed using path planning and collision avoidance algorithm. In the multi-robot working environment, each robot plans its own path independently. The D* Lite planning algorithm is used to generate a path on the map of each robot. The robot needs to know the position of other robots that are considered the obstacles on its grid map. By using the D* Lite algorithm and the grid map, a shortest path is planned in a fast time. Figure 8 shows the results of 2 situations: a path planned without a static obstacle and a path planned after detecting a static obstacle.

In the dynamic working environment, the robot and the obstacle always move on the map of the floor. The robot R_i is an obstacle of the robot R_j on its map. Conversely, the robot R_j is an obstacle of the robot R_i on its map. A common database is used to continuously synchronize the position of robot in the real time [13–15]. The robot becomes the dynamic objectives on the map by updating the changed positions from database [16]. This method helps the robot update the position

Figure 9. The path of robot R_i planned by using multi-robot algorithm after matching robot R_j .

of another robot immediately and then re-plan the path by using D* Lite algorithm as shown in Figure 9 [5].

Let's consider that there are two robots moving on planned paths with difference controlled speeds on the grid map of the floor in the warehouse. A potential collision between two robots is able to occur at an “unsafe” point when the robots are moving as some situations shown in Figure 10. The unsafe point can be found on the planned path of each robot by reading data from the database on the server. It is a next planned point of a robot which its value is able to equal to value of a next planned point of another robot in the planned path. So a potential collision as shown in Figure 10 is able to occur when these robots move to the unsafe point at the same time.

In order to solve the problem, an unsafe point P_i from start point S_i of the robot R_i and an unsafe point P_j from start point S_j of the robot R_j on the planned path are considered. These points are used to find a higher priority robot and a lower priority robot on the server as shown in Figure 11. When $P_i = P_j$, P_i and P_j are changed to a “predict” point. The decision rules are needed based on the predict point to detect and decide that what is a higher priority robot. The decision rules are as follows:

Case 1: both robots are standing on two QR code points S_i and S_j . It means that $d_i = 0$ and $d_j = 0$. Then, the higher priority robot is the robot with the nearest goal point by comparing l_i and l_j . If $l_i \leq l_j$, the higher priority robot is R_i and R_j is low priority. Conversely, if $l_i > l_j$, the higher priority robot is R_j and R_i is low priority.

Figure 10. Potential collision situations when the robots moving on the map.

Case 2: there are any robot that moved pass over the QR code with a distance and another robot standing on the QR code. It means that $\{d_i \neq 0 \text{ and } d_j = 0\}$ or $\{d_i = 0 \text{ and } d_j \neq 0\}$. Then the robot passed over the QR code is decided as the higher priority robot.

Case 3: both of robots moved pass over the QR code. It means that $d_i \neq 0$ and $d_j \neq 0$. Then, the higher priority robot is the robot which the distance is longer. If $d_i > d_j$, the higher priority robot is R_i and R_j is low priority. Conversely, if $d_i \leq d_j$, the higher priority robot is R_j and R_i is low priority.

In Case 1 and Case 2, the higher priority robot is enabled to continue moving to the next point and the lower priority robot is not enabled to move or must to re-plan a new path. The predict point of the higher priority robot will be changed to a “safe” point on the map. The server orders the robot to move to the safe point on its path and at the same this time, the predict point of the lower priority robot will be changed to a “re-plan” point. The server also sends a stopping command to the lower priority robot when matched the re-plan point and then the server re-plan a new path by using D* Lite algorithm when the higher priority robot matched its safe point. In the waiting time for the new path, the lower priority robot still stands on its start point on the grid map. Especially, Case 3 is a hard and urgent case

Figure 11. The collision avoidance in the multi-robot dynamic working environment.

Figure 12. Collision avoidance algorithm for colliding robots.

that both of robots passed over the QR code and were standing on the line of the map. Therefore, the higher priority robot in Case 3 is enabled to move to its next point, the lower priority robot will be stopped immediately on the line. After found the next point, the predict point of the higher priority robot will be considered again and changed to the re-plan point if it's value is equal to value of the start point of the lower priority robot or changed to the safe point when it's value is not same with value of the start point of the lower priority robot. The server just sends the running command to the lower priority robot after the higher priority robot passed over the predict point of the lower priority. Figure 12 shows a collision avoidance algorithm for colliding robots. These statements are repeated until the goal is met.

4. Experimental Results

Figure 13. Experimental results of QR code alignment method when the robot is turning.

4.1 Results of QR Code Alignment Methodology

The experimental results in Figure 13 show that the robot is turning left and the start position of the point A is out of the safe zone (Figure 13(a)). So, the QR code alignment methodology is applied to detect the time when the camera lost image and the position of the point A is corrected by using sampling data with value of the QR code's angle as shown in Figure 13(b) and 13(c). After finishing data analyzing and deciding the correct position in Figure 13(d), the robot is stopped and then the robot moves backwards with a small velocity. When the image is detected again and the value of angle is changed, at that time, the error is fixed successfully as shown in Figure 13(e). The robot continues modifying the angle to find the goal angle and finish the turning processing as Figure 13(f).

4.2 Navigation Results of One Robot in Static Obstacle Environment

Figure 14 shows the experimental results of the robot navigation in static obstacle environment. The robot can move from the start point (0, 1) on the planned path (0, 1F-1, 1F-2, 1F-3, 1F-4, 1E) and meet the goal point (4, 1) if there is no any obstacle on its path (Figure 14(a)). Unfortunately, a static obstacle can appear at any point on its path. In the Figure 14(b), two static obstacles at the point (1, 0) and (2, 1) are put on the path of robot. After checking the next planned point by using the proposed motion planning algorithm, the path is re-planned. The new planned path is (1, 1R-1, 2L-2, 2F-3, 2F-4, 2L-4, 1E). At this time, the start point is changed to the point (1, 1) and the next planned point is point (1, 2) as shown in Figure 14(c). The

Figure 14. Experimental results of the robot navigation in static obstacle environment.

robot tracks and follows the line on the floor to move to the next planned point on the new planned path (Figure 14(d)). Same processing continues until the robot meets a static obstacle at the points (4, 3) and (3, 1); the new paths after re-planning

successfully are (3, 3F-3, 2F-3, 1L-4, 1E) and (3, 2F-4, 2L-4, 1E) (Figure 14(e) and Figure 14(f)). Finally, the robot stops at the goal point (4, 1) and finish its navigation without any collision (Figure 14(g)).

Figure 15. Experimental results of the navigation method for two transportation robots.

4.3 Navigation Results of Two Robot

Figure 15 shows experimental results that the management server controls two transportation mobile robots which moving on its planned paths. The planned path of robot 1 is (2, 0F-2, 1F-2, 2E) and the path of robot 2 is (0, 1F-1, 1F-2, 1F-3, 1F-4, 1E). Two robots start moving from the start point (2, 0) and (0, 1) after receiving a command from the server as shown in Figure 15(a). Two robots track the line continuously to move on the floor and then find the next point (2, 1) and (1, 1). The proposed motion planning algorithm continuously checks the overall condition of the robots in the real-time. The algorithm is used to re-plan the path for robot 2 when it meet the point (1, 1) and also meet an unsafe point (2, 1). The first re-planned path of robot 2 is (1, 1R-1, 2L-2, 2F-3, 2F-4, 2L-4, 1E) as shown in Figure 15(b). After receiving the re-planned path from the server, the robot 2 turns right to avoid collision with the robot 1 and moves to the next point (1, 2) in Figure 15(c). At that time, the robot 1 meet the point (2, 1) on its path. The collision avoidance is continuously checked and is detected one more unsafe point (2, 2) when the robot 2 meet the point (1, 2) on its new path. Therefore, the path is re-planned by using the D* Lite path planning algorithm again. The second re-planned path

of the robot 2 is (1, 2F-1, 3L-2, 3F-3, 3F-4, 3L-4, 2F-4, 1E) as shown in Figure 15(d). After receiving the second path, the robot 2 move forwards to the point (1, 3) without turning left to the point (2, 2) as the old path as Figure 15(e). Robot 1 stops at its goal point (2, 2). Robot 2 moves to its goal point (4, 1) with the second re-planned path if there are no any unsafe point in its path when detecting by using the proposed motion planning algorithm as shown in Figure 15(f)–15(h).

5. Conclusion

In the paper, the QR code alignment method and the advanced motion planning method are proposed to manage and control the transportation robots in the dynamic logistic environment. For the navigation of the robot, the QR code alignment methodology is developed to modify the position of robot on the exact QR code position automatically when it is near the QR code. The proposed alignment method uses the real-time QR code's angle from the robot and the sampling time. The method calculates the error zone and helps the robot to turn left or right exactly. The experimental results show that the position of robot is aligned correctly and the robot moves continuously on its path without error. In the multi-robot environment, the proposed motion

planning algorithm is applied to allow the management server to control and avoid a potential collision between the robots in the real-time. The collision avoidance algorithm checks a next planned point continuously, predicts exactly a safe point and then decides a higher priority robot. The lower priority robot must be re-planned the path by using D* Lite algorithm to avoid the higher priority robot. The proposed planning method is designed to control the robots without collision in dynamic environment. The experimental results show that the proposed motion planning algorithm for the multi-robots can check and re-plan the path by using D* Lite algorithm to avoid a potential collision correctly. Also, the management server updates the data of the transportation robots quickly and exactly in the real-time.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgments

This work was supported by the GRRC program of Gyeonggi Province (No. GRRC-Hankyong 2017-B01, Smart Logistics Technology Institute).

References

- [1] N. T. Truc, E. H. Sun, Y. M. Kim, and Y. T. Kim, "Navigation method using fuzzy line tracking for the transportation robot," in *Proceedings of 2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, Kitakyu-shu, Japan, 2014, pp. 819-822. <https://doi.org/10.1109/SCIS-ISIS.2014.7044874>
- [2] Y. M. Kim and Y. T. Kim, "Design of logistics transportation robot based on modular conveyor rack and path planning in logistics center," *Journal of Institute of Control, Robotics and Systems*, vol. 22, no. 2, pp. 83-88, 2016. <https://doi.org/10.5302/J.ICROS.2016.15.0194>
- [3] J. H. Cho and Y. T. Kim, "Design of autonomous logistics transportation robot system with fork-type lifter," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 3, pp. 177-186, 2017. <https://doi.org/10.5391/IJFIS.2017.17.3.177>
- [4] J. H. Cho, E. H. Jung, J. Y. Kim, M. H. Park and Y. T. Kim, "A study on improved navigation algorithm of logistics transportation robot for freight moving on the rack," *Journal of Korean Institute of Intelligent Systems*, vol. 28, no. 5, pp. 436-442, 2018. <https://doi.org/10.5391/JKIIS.2018.28.5.436>
- [5] N. V. Vui and Y. T. Kim, "Motion planning algorithm for the transportation robot using QR code alignment," in *Proceedings of the 18th International Symposium on Advanced Intelligent Systems*, Daegu, Korea, 2017, pp. 130-134.
- [6] H. U. Yoon and D. W. Lee, "Subplanner algorithm to escape from local minima for artificial potential function based robotic path planning," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 4, pp. 263-275, 2018. <https://doi.org/10.5391/IJFIS.2018.18.4.263>
- [7] K. Lee, D. Y. Im, B. Kwak, and Y. J. Ryoo, "Design of fuzzy-PID controller for path tracking of mobile robot with differential drive," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 3, pp. 220-228, 2018. <https://doi.org/10.5391/IJFIS.2018.18.3.220>
- [8] S. Koenig and M. Likhachev, "D*Lite," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, Edmonton, Canada, 2002, pp. 476-483.
- [9] G. H. Lee and J. Jung, "Line tracking control of a two-wheeled mobile robot using visual feedback," *International Journal of Advanced Robotic Systems*, vol. 10, no. 3, 2013. <https://doi.org/10.5772/53729>
- [10] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933-945, 2009. <https://doi.org/10.1177/0278364909340445>
- [11] S. Koenig, M. Likhachev, Y. Liu, and D. Furey, "Incremental heuristic search in AI," *AI Magazine*, vol. 25, no. 2, pp. 99-112, 2004.
- [12] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93-146, 2004. <https://doi.org/10.1016/j.artint.2003.12.001>

- [13] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping," in *Proceedings of IEEE International Conference on Robotics and Automation (Cat. No.00CH37065)*, San Francisco, CA, 2000, pp. 321-328. <https://doi.org/10.1109/ROBOT.2000.844077>
- [14] L. Dozio and P. Mantegazza, "Real time distributed control systems using RTAI," in *Proceedings of the 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Hokkaido, Japan, 2003, pp. 11-18. <https://doi.org/10.1109/ISORC.2003.1199229>
- [15] T. Choi, H. Do, D. Park, C. Park, and J. Kyung, "Real-time synchronisation method in multi-robot system," *Electronics Letters*, vol. 50, no. 24, pp. 1824-1826, 2014. <https://doi.org/10.1049/el.2014.2959>
- [16] M. Deng, A. Inoue, K. Sekiguchi, and L. Jiang, "Two-wheeled mobile robot motion control in dynamic environments," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 3, pp. 268-272, 2010. <https://doi.org/10.1016/j.rcim.2009.11.005>

Nguyen Van Vui received his B.S. degree in Physics and Electronic Engineering, University of Science, Ho Chi Minh City, Vietnam in 2010. He was a firmware engineer at Greystone Data System, Ho Chi Minh City

in 2010 and 2011. In 2011 and 2012, he was a hardware engineer at Elek Electronic, Binh Duong province. From 2013 to 2015, he was a software engineer in Robert Bosch Engineering Vietnam, Ho Chi Minh City. He received M.S. degrees in Department of Electrical, Electronic and Control Engineering at Hankyong National University, Anseong, Korea in 2017. Since 2017, he has been CEO of Smart Device Solutions Co. Ltd, Ho Chi Minh City, Vietnam. His present interests include intelligent system, smart home system and its applications.

E-mail: vuinguyen.qn@gmail.com

Yong-Tae Kim received his B.S. degree in Electronic Engineering from Yonsei University, Seoul, Korea in 1991, and his M.S. and Ph.D. degrees in Electrical and Electronic Engineering from KAIST, Daejeon, Korea in 1993 and 1998, respectively. From 1998 to 2000, he was a senior engineering in Samsung. In 2006, he was a visiting scholar at Department of Electrical Engineering in University of Illinois at Urbana-Champaign. Since 2002, he has joined the faculties of the Department of Electrical, Electronic and Control Engineering at Hankyong National University, where he is currently a professor. His current research interests include intelligent robots, logistics automation, intelligent control and its applications.

E-mail: ytkim@hknu.ac.kr