IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 January 2023, accepted 16 February 2023, date of publication 23 February 2023, date of current version 28 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3247730

RESEARCH ARTICLE

Path Planning for Mobile Robot Considering Turnabouts on Narrow Road by Deep Q-Network

TOMOAKI NAKAMURA, MASATO KOBAYASHI, (Member, IEEE), AND NAOKI MOTOI, (Member, IEEE)
Graduate School of Maritime Science, Kobe University, Kobe 658-0022, Japan
Corresponding author: Naoki Motoi (motoi@maritime.kobe-u.ac.jp)

ABSTRACT This paper proposes a path planning method for a nonholonomic mobile robot that takes turnabouts on a narrow road. A narrow road is any space in which the robot cannot move without turning around. Conventional path planning techniques ignore turnabout points and directions determined by environmental data, which might result in collisions or deadlocks on a narrow road. The proposed method uses the Deep Q-network (DQN) to obtain a control strategy for path planning on narrow roads. In the simulation, the robot learned the optimal velocity commands that maximized the long-term reward. The reward is designed to reach a target with a smaller change in robot velocity and fewer turnabouts. The success rate and the number of turnabouts in the simulation and experiment were used to evaluate the trained model. According to simulation and environmental data, the proposed strategy enables the robot to travel on narrow roads. Additionally, these outcomes demonstrate comparable performance on a number of roadways that are not part of the learning environments, supporting the robustness of the trained model.

INDEX TERMS Mobile robot, path planning, turnabout, reinforcement learning.

I. INTRODUCTION

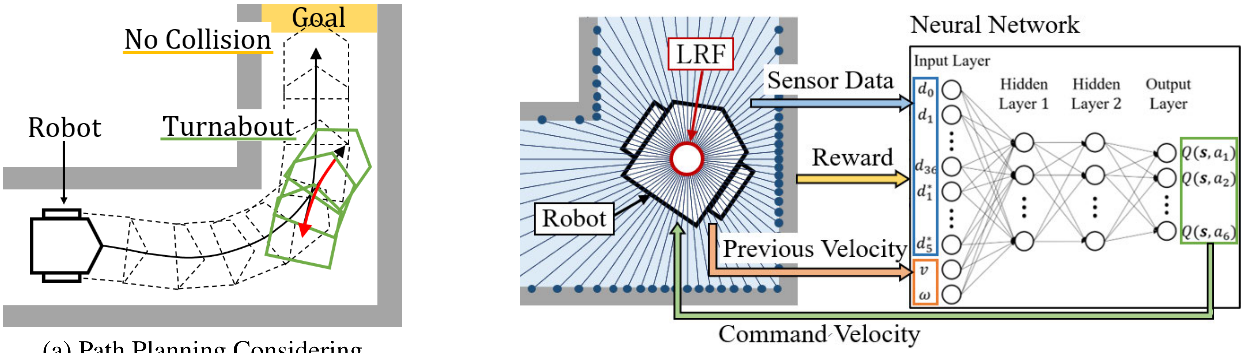
Through the development of robotics and computer technologies, autonomous mobile robots have greatly impacted numerous fields. Urban transit, industry, public health, and domestic settings all use mobile robots [1], [2], [3]. Recent studies on the autonomous control of robots include dynamic obstacle avoidance [4], [5], swarm robotic systems [6], [7], and autonomous parking systems [8], [9]. These studies were carried out to guarantee safe and efficient transportation in various residential areas [3].

Residential areas include several restrictions on road widths due to obstacles and crowds. In this work, two types of robots—holonomic mobile and nonholonomic—are used to categorize the ways for moving over restricted roads. The holonomic mobile robot can move in any direction regardless of its configuration. Numerous mechanisms have been reported, such as an omnidirectional wheeled robot [10], an omnidirectional crawler robot [11], a ballbot [12], and a floor tiling robot [13]. However, the exorbitant expense of the sophisticated structures restricts the application of holonomic robots.

For holonomic robots, a variety of path planning techniques have been suggested, including the artificial potential field method (APF) [14], velocity obstacle (VO) [15], and bug algorithm [16]. APF in particular has seen widespread application [17], [18], [19]. Using a potential field, this technique creates routes that direct the robot to the destination while dodging obstacles. A nonholonomic mobile robot, on the other hand, may not be able to go along the potential field's gradient; hence, the APF may generate infeasible trajectories.

Most mobile robots today are subject to nonholonomic restrictions. The smallest turning radius of the nonholonomic mobile robot, which determines the width of the road on which it can travel, was determined based on its specifications. We focus on a narrow road defined as any space in which the robot cannot move without turning around. On narrow roads, the robot cannot pass an obstacle if the total length

The associate editor coordinating the review of this manuscript and approving it for publication was Giulio Reina.



(a) Path Planning Considering Turnabouts

(b) Schematic Diagram of Proposed Method

FIGURE 1. Path planning on narrow road using proposed method.

of the obstacle and robot exceeds the road width. Therefore, this research does not assume general environments in residential areas such as dynamic environments.

This study examines path planning on narrow roads with regard to turnabouts. Numerous works have been published on path planning for nonholonomic robots [20], [21], [22], [23]. However, a large majority of these studies addressed forward motion alone from an efficiency standpoint. There have been reports of conventional path planning approaches that generate forward and reverse motions, such as the dynamic window approach (DWA) [24] and Reeds-Shepp curve [25]. The DWA calculates real-time translational and angular velocities based on the robot's configuration and obstacle data, making the robot move backwards whenever the translational velocity is negative. The Reeds-Shepp curve generates the shortest path using line segments and arcs. The path planning method using Reeds-Shepp curve and Rapidly-exploring Random Tree (RRT) [26] algorithm was reported [27]. But these methods may not be able to reach a goal because they don't take environmental information or turnaround points into account.

In traditional approaches, such as the DWA and Reeds-Shepp curve, the control strategy is planned ahead of time with information like the system model of the robot and its environment. Modern works have, however, developed path planning methods by the learning-based approach [28], [29], [30], [31]. This approach obtains the control strategy by repeatedly learning using machine learning (ML). Reinforcement learning (RL) is an area of ML. When using RL to plan a path, the control strategy is determined by how the robot interacts with its environment. Farias et al. came up with an algorithm to control the position of the robot using RL [32], by manipulating velocities based on the error between the target direction and heading angle. Chang et al. used RL to tune the DWA parameters [33]. Deep reinforcement learning (DRL), which combines RL with deep learning, has been used in many studies recently to plan paths [34]. Using DRL and imitation learning, Pfeiffer et al. came up with a way to plan a route for navigation without a map [35]. This method changes the speed of the robot based on what the sensors measure and where the target is in relation to the robot. Wen et al. wrote about path planning using DRL and simultaneous localization and mapping (SLAM) framework [36]. Yao et al. made the APF better by adding DRL [37]. But as far as we know, no studies on learning-based approaches have looked at moving backward when planning a path on a narrow road.

This paper suggests a method to plan a path that takes turnabouts into account and is based on learning. This study assumes that the proposed method can be used with a mobile robot that is not holonomic and travels on narrow roads. The proposed method lets the robot move on a narrow road with turnabouts, as shown in Fig. 1(a). Based on a trained model, the proposed method changes how fast a robot moves in both directions. A Deep Q-network (DQN) [38], which is a DRL method, is used to get a trained model for path planning that takes turnabouts into account.

As shown in Fig. 1(b), the proposed method selects velocity commands from six candidates based on sensor data and previous velocities. The reward is designed to achieve a goal with fewer turnabouts and a smaller change in robot velocity. The robot learns the best control strategy for maximizing the long-term reward. As a result, the proposed method generated paths that included turnabouts that corresponded to the surrounding environment.

The contributions of this study are summarized as follows.

- To allow the robot to travel on narrow roads, a path planning method that takes turnabouts is proposed for a nonholonomic mobile robot.
- Using RL, the proposed method generates a trained model for path planning on narrow roads. In real time, the trained model determines velocity commands with negative values based on environmental data.
- The proposed method enables the robot to travel on narrow roads, as evidenced by simulation and environmental results. The trained model's robustness was confirmed by performing path planning on several roads that differed from the learning environments.

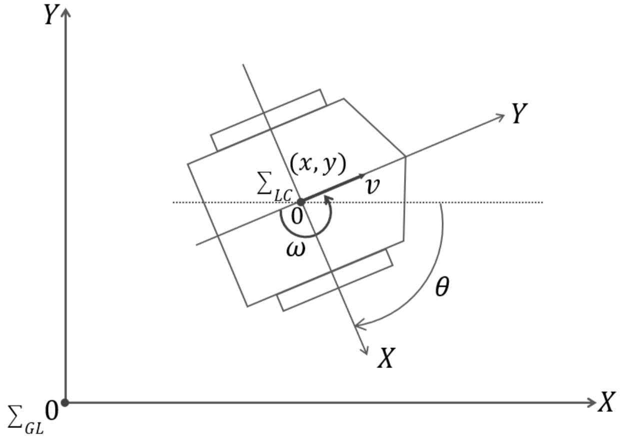


FIGURE 2. Coordinate system of mobile robot.

The rest of this paper is structured as follows. Section II describes the mobile robot's modeling. The DWA and Reeds-Shepp curves are presented as traditional path planning methods in Section III. Section IV proposes a path planning method that incorporates turnabouts by RL. Sections V and VI present simulation and experimental results to validate the proposed method's validity. Section VII concludes the paper and discusses future research.

II. MODELING OF MOBILE ROBOT

This section describes the modeling of the mobile robot. The global coordinate system $\sum_{GL}$ and the local coordinate system $\sum_{LC}$ are defined in Fig. 2. The robot has two driving wheels on the X-axis in $\sum_{LC}$. The center of the wheels is defined as the origin in $\sum_{LC}$. The Y-axis in $\sum_{LC}$ corresponds to the traveling direction of the mobile robot, whereas (x, y) denotes its position in $\sum_{GL}$. θ represents the angle between X-axes in $\sum_{GL}$ and $\sum_{LC}$.

The kinematics of the robot in $\sum_{GL}$ is described as follows [39].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

where v and ω are the translational and angular velocities of the mobile robot in $\sum_{LC}$.

III. CONVENTIONAL METHOD

This section describes the conventional methods of DWA and the Reeds-Shepp curve.

A. DYNAMIC WINDOW APPROACH (DWA)

This subsection discusses DWA [24]. DWA accounts for circular paths calculated from translational and angular velocities. The robot specifications first define the search space for the robot velocities. For the robot to stop before the closest obstacle on the circular path, velocity pairs were considered. If the search space contains negative translational velocity values, the robot can move backward.

Second, by maximizing objective function J, the optimal velocity pair is chosen from the dynamic window.

$$J = W_1 \cdot e_{head} + W_2 \cdot e_{dist} + W_3 \cdot e_{vel} \tag{2}$$

where e_{head}, e_{dist}, and e_{vel} represent evaluations of the angle along the target direction, the distance from the closest obstacle, and the translational velocity of the robot, respectively. W_1, W_2 and W_3 are weighting coefficients. The details of the DWA are further elaborated in [24].

B. REEDS-SHEPP CURVE

The Reeds-Shepp curve [25] is presented in this subsection. If the initial and final positions and directions are specified, the Reeds-Shepp curve finds the shortest path. The minimum turning radius of a robot is constrained by this method. This method divides paths into nine categories where arcs and line segments fit smoothly: $C|C|C$, $CC|C$, CLC, $CC_u|C_uC$, $C|C_uC_u|C$, $C|C_{\pi/2}LC$, $C|C_{\pi/2}LC_{\pi/2}|C$, $C|CC$, and $CLC_{\pi/2}|C$. C and L denote an arc of the minimum turning radius and a line segment, respectively, while | indicates a turnabout. The subscript $\bigcirc_u$ represents the length of L or the curvature angle of C. These categories have 48 paths, among which the shortest was selected. The details of the Reeds-Shepp curve are explained in [25].

IV. PROPOSED METHOD

This section proposes a path planning method that takes turnabouts into account. The traditional method, which generates forward and backward motions, is incapable of determining turnabout points and directions based on environmental data. Using a learning-based approach, the proposed method obtains a control strategy. The proposed method is called NT-DQN, which represents path planning on narrow roads considering turnabouts by DQN. The DQN is an RL method that employs a deep neural network (DNN) to estimate the state-action value function known as the Q-value. We implemented the DQN in the proposed method for two reasons.

- It is difficult to design a control strategy or obtain training datasets in advance when considering a path planning method with turnabouts based on environmental information. The RL-based path planning method obtains the control strategy through the interaction of the robot and its surroundings, with no prior training datasets.
- Using neural networks, the DQN can handle high-dimensional input data. As input data, the proposed method employs continuous values comprised of sensor data and robot velocities.

A. PROBLEM STATEMENT

The goal of this research is to generate paths that include turnabouts that correspond to the surrounding environment. The proposed method expresses velocity command $\boldsymbol{u}$ as follows:

$$\boldsymbol{u}_k = [v_k, \omega_k]^T = f\,(\boldsymbol{d}_k, \boldsymbol{u}_{k-1}) \tag{3}$$

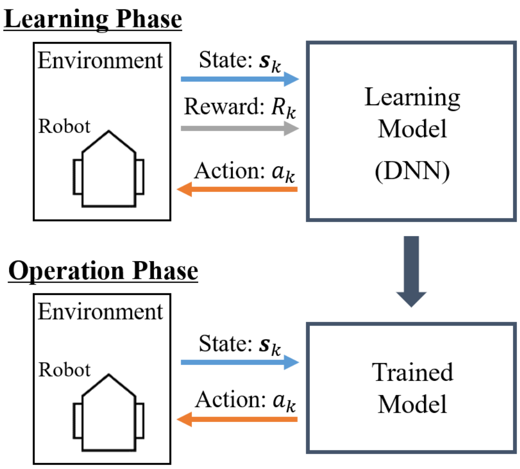


FIGURE 3. Diagram of path planning method using RL.

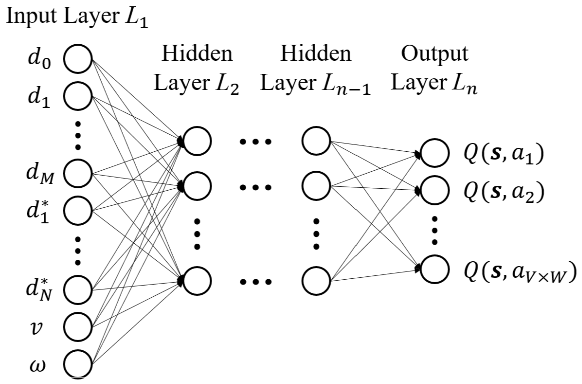


FIGURE 4. Model architecture.

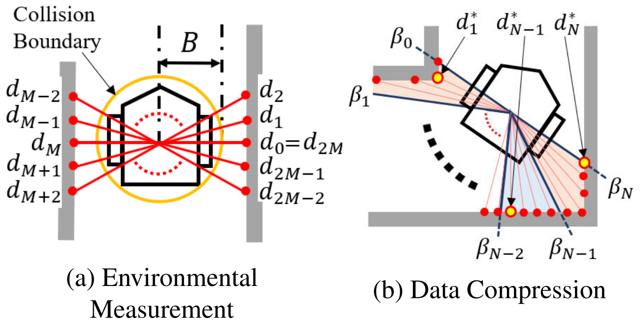
where $\boldsymbol{d}$ represents the set of sensor data measured using a laser range finder (LRF), while the subscript $\bigcirc_k$ denotes the time step and the function f represents the DQN learning model that determines the velocity command from sensor information and previous velocities.

The RL-based path planning method was divided into two phases, as illustrated in Fig. 3. During the learning phase, the robot used trials to update the learning model. The ϵ-greedy method [40] was used to determine velocity commands. The random action is chosen with probability ϵ, and the action that maximizes the Q-value among all velocity candidates is chosen with probability $1-\epsilon$. This study simulated the learning process. The robot performs actions during the operation phase based on the trained model acquired during the learning phase. At each time step, the action with the highest Q-value is chosen.

B. DESIGN OF LEARNING MODEL

1) MODEL ARCHITECTURE

Fig. 4 depicts the NT-DQN model, which determines the state-action value function $Q(\boldsymbol{s}, a; \Theta)$ for each action. Here, Q is the expected value of the long-term rewards for taking action a in state s, while Θ is the weight of the model. Adam, an optimization method, was used to update the weights of each layer [41]. The following are the definitions of state, action, and reward.



(a) Environmental Measurement

(b) Data Compression

FIGURE 5. State observation.

2) STATE DEFINITION

The state $\boldsymbol{s}_k$ at time step k is defined as follows.

$$\boldsymbol{s}_k = \left[\left(\boldsymbol{d}_k^{fwd}\right)^T, \left(\boldsymbol{d}_k^{bwd}\right)^T, v_{k-1}, \omega_{k-1} \right]^T \tag{4}$$

where $\boldsymbol{d}^{fwd}$ and $\boldsymbol{d}^{bwd}$ represent the sensor data in the forward and backward directions of the robot, respectively.

As shown in Fig. 5(a), $2M$ datapoints were measured around the robot. $d_i\,(0 \le i \le 2M-1)$ is the distance to the obstacles, as measured by the sensor. $\boldsymbol{d}^{fwd}$ is defined as follows:

$$\boldsymbol{d}^{fwd} = [d_0, d_1, \cdots, d_i, \cdots, d_M]^T \tag{5}$$

In addition, the minimum value of the sensor data is defined as follows.

$$d_{min} = \min_i \{d_i \mid i \in \mathbb{Z},\ 0 \le i \le 2M-1\} \tag{6}$$

Concerning the data in the backward area of the robot $d_i\,(M \le i \le 2M)$, the amount of data is compressed from $M+1$ to N by the following equation:

$$d_j^* = \min_i \left\{ d_i \,\middle|\, i \in \mathbb{Z},\ \frac{M}{\pi}\beta_{j-1} \le i \le \frac{M}{\pi}\beta_j \right\} \tag{7}$$

$$\beta_j = \left(1 + \frac{j}{N}\right)\pi \tag{8}$$

where M and N satisfy the relationship $M > N$. As shown in Fig. 5(b), the angular range of $\pm\pi/2$ [rad] to the negative in the negative direction of the Y-axis in Σ_{LC} is divided into N areas at equal intervals. The sensor data's minimum value for each divided area is extracted. Using compressed data, $d_j^*\,(1 \le j \le N)$, $\boldsymbol{d}^{bwd}$ is defined as follows:

$$\boldsymbol{d}^{bwd} = [d_1^*, d_2^*, \cdots, d_j^*, \cdots, d_N^*]^T \tag{9}$$

As shown in (4), the state $\boldsymbol{s}_k$ also contains velocity data v_{k-1} and ω_{k-1}, which are the robot velocities at the previous time step.

3) ACTION DEFINITION

The action a_k at time step k is defined as $a_l (1 \leq l \leq V \times W)$, with a_l denoting a pair of translational and angular velocities. The translational and angular velocities are divided into sets V and W, respectively. Thus, the number of actions is given by their Cartesian cross-product, $V \times W$. One pair from a_l was selected for each time step.

4) REWARD DEFINITION

The reward R_{k+1} for action a_k is defined by the following equation to avoid obstacles and reach the target position:

$$R_{t+1} = \begin{cases} K_{goal}, & \text{if } (x_{k+1}, y_{k+1}) \in \boldsymbol{G} \\ -K_{coll}, & \text{if } d_{min} \leq B \\ r_{dist} + r_{turn} + r_{act}, & \text{otherwise} \end{cases} \tag{10}$$

where K_{goal} and K_{coll} are the positive constants. $\boldsymbol{G}$ is the determined region to reach the target position. The collision check is carried out by comparing the minimum value of the sensor data d_{min} with the robot's radius B. If the robot arrived at the target position or collided, the robot position was reset and the next episode began. In addition, the following reward functions are provided to evaluate the episode's path planning process.

$$r_{dist} = -K_{dist} \left(\|\boldsymbol{x}_{goal} - \boldsymbol{x}_{k+1}\| - \|\boldsymbol{x}_{goal} - \boldsymbol{x}_k\| \right) \tag{11}$$

$$r_{turn} = \begin{cases} -K_{turn}, & \text{if } v_k < 0 \text{ and } v_{k-1} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$r_{act} = \begin{cases} K_{act}, & \text{if } \boldsymbol{u}_k - \boldsymbol{u}_{k-1} = \boldsymbol{0} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where K_{dist}, K_{turn} and K_{act} are positive constants. The vectors of the robot position at time step k and the target position are given by $\boldsymbol{x}_k$ and $\boldsymbol{x}_{goal}$, respectively. r_{dist} is the distance from the target position evaluation function. If the robot approaches the target position faster than the previous time step, a positive value is obtained. r_{turn} is a function for minimizing the number of turnabouts. When the participant returned, a negative value was obtained. r_{act} is the velocity change evaluation function. The higher the evaluation, the smaller the velocity change.

V. SIMULATION

This section presents the simulation results used to train the model and validate the proposed method's effectiveness. The DWA and Reeds-Shepp curves were used as the conventional methods in the comparison between the proposed method and the conventional methods. Path planning was performed in several environments using the proposed method to evaluate its robustness.

A. SIMULATION SETUP

Gazebo with Robot Operating System (ROS) Melodic was used to run the simulations. TurtleBot3 Burger [42], a ROS standard platform robot, was used in this study. The robot's minimum turning radius was set to 0.5 m.

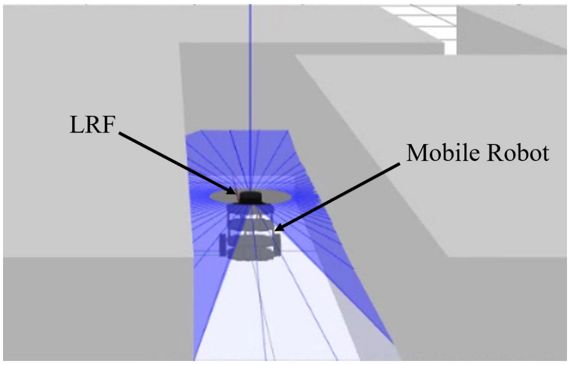


FIGURE 6. Gazebo simulator.

TABLE 1. Hyperparameter Settings.

Variables	Description	Value
M	Half number of sensor data around robot	36
N	Number of sensor data in backward range of robot	5
V	Number of translational velocity candidates	2
W	Number of angular velocity candidates	3
n	Number of layers in NT-DQN model	4
ϵ	Probability used in ϵ -greedy method	0.3
B	Radius of robot	0.125 m
K_{goal}	Reward to reach target position	10
K_{coll}	Reward to avoid collision	10
K_{dist}	Coefficient of reward function r_{dist}	10
K_{turn}	Coefficient of reward function r_{turn}	2
K_{act}	Coefficient of reward function r_{act}	0.2

TABLE 2. Action options.

Action	v [m/s]	ω [rad/s]
a_1	-0.1	-0.2
a_2	-0.1	0
a_3	-0.1	0.2
a_4	0.1	-0.2
a_5	0.1	0
a_6	0.1	0.2

Table 1 shows the NT-DQN model hyperparameters manually set prior to training. We set the hyperparameters considering the calculation cost and performance of the learning model (success rate and number of turnabouts). The LRF was used to measure 72 sensor data points around the robot. The sensor data's maximum value was set to 1.0m. The sensor data in the robot's backward area were compressed into five data points using (7). As a result, the NT-DQN model's input layer had 44 nodes, with 42 sensor data and two velocity data. The velocity candidates for the proposed method are shown in Table 2. At each time step, the robot chooses one of the six velocity pairs shown in Fig. 7. As a result, the NT-DQN model's output layer had six nodes. There were two hidden layers in the NT-DQN model, each with 50 nodes.

The simulation environment used during the learning phase is depicted in Fig. 8. $(x_{init}, y_{init}) = (0, 0)$ and $\theta_{init} = 0$ are the initial position and angle. The target position in

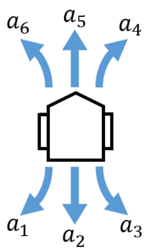
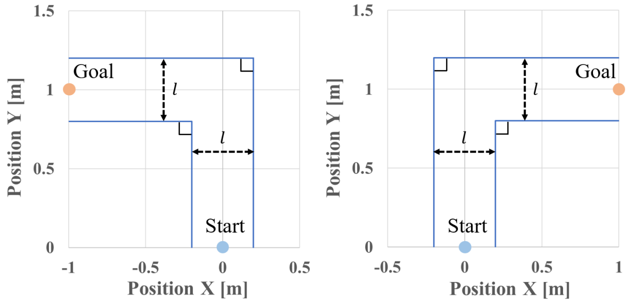


FIGURE 7. Movement direction.



(a) Environment 1 (b) Environment 2

FIGURE 8. Simulation environments (learning phase).

Environment 1 was set to $(x_{goal}, y_{goal}) = (-1, 1)$, as shown in Fig. 8(a), where (x_{goal}, y_{goal}) denotes the target position in Σ_{GL}. When $x_k \leq -1$, the robot was thought to have reached the target. The target position in Environment 2 was defined as $(x_{goal}, y_{goal}) = (1, 1)$ in Fig. 8(b). When $x_k \geq 1$, the robot was thought to have arrived at its destination.

During the learning phase, the simulation was carried out as follows: During each episode, the robot moved from its starting point. If the robot arrived at the target position or collided, its position was reset and the next episode began. Furthermore, the time step limit in each episode was set to 200. The learning procedure was divided into three steps. In the first step, only Environment 1 was used in the simulation. The width of the road, l, was set to 0.4 m. In the second step, the simulation was run using Environments 1 and 2, with l set to 0.4 m. The learning model was updated as a result of these steps to generate turnabouts at road corners. Furthermore, the third step was carried out using Environments 1 and 2, with l set to 0.4 or 0.45 m to improve the robustness of the learning model. The simulation environment was chosen at random at the start of each episode in the second and third steps.

The simulation environment used during the operation phase is depicted in Fig. 9. $(x_{init}, y_{init}) = (0, 0)$ and $\theta_{init} = 0$ are the initial position and angle. The target position was determined to be $(x_{goal}, y_{goal}) = (1, 2)$. When $y_k \geq 2$, the robot was thought to have reached the target.

There were two types of simulations in the operational phase. First, the proposed method was compared to conventional methods for path planning. The simulations were split into three groups: DWA (Case 1), Reeds-Shepp curve (Case 2), and NT-DQN (Case 3). These simulations were run in a trained environment with the road width l and bending angle α set to 0.4 m and 90°, respectively. The road condition in a trained environment is the same as that in a learning environment. Second, path planning using the proposed method was performed in untrained environments to validate the trained model's robustness. The road conditions in an untrained environment differ from those in a learning environment. The simulations were run with 12 road conditions (l, α), where l and α were chosen from 0.35, 0.4, or 0.45 m, and 75, 90, 105, or 120°, respectively.

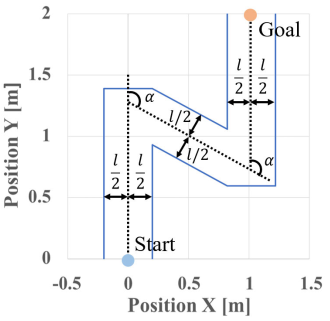
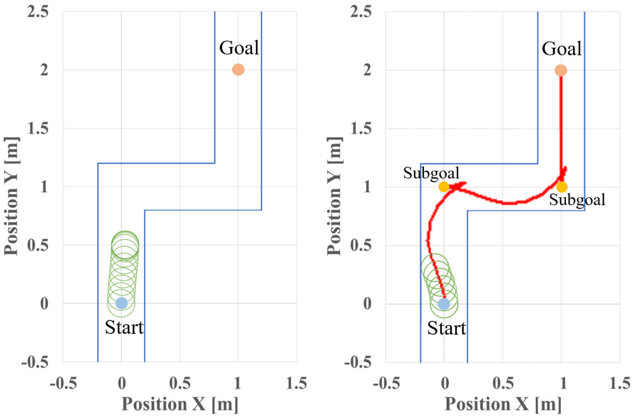


FIGURE 9. Simulation environment (operation phase).



(a) Case 1 (b) Case 2

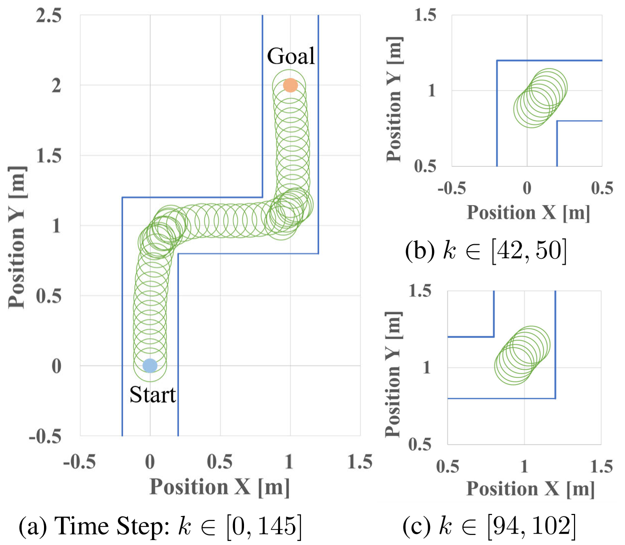
FIGURE 10. Simulation results in Cases 1-2 (conventional method).

B. SIMULATION RESULTS

1) COMPARISON OF PATH PLANNING METHODS

The simulation results for Cases 1–3 are shown in Figs. 10-11. The robot's path is indicated by green circles. In Case 1, DWA was used to plan the path. The robot repeated the forward and backward motions around (0.498, 0.035), as shown in Fig. 10(a). As a result, the robot does not reach the road's corners.

The Reed-Shepp curve was used to plan the path in Case 2. The two subgoals were set as $[x_{sub1}, y_{sub1}, \theta_{sub1}] = [0, 1, -\pi/2]$ and $[x_{sub2}, y_{sub2}, \theta_{sub2}] = [1, 1, 0]$ between the initial and target positions. The positions of the subgoals are



(a) Time Step: $k \in [0, 145]$

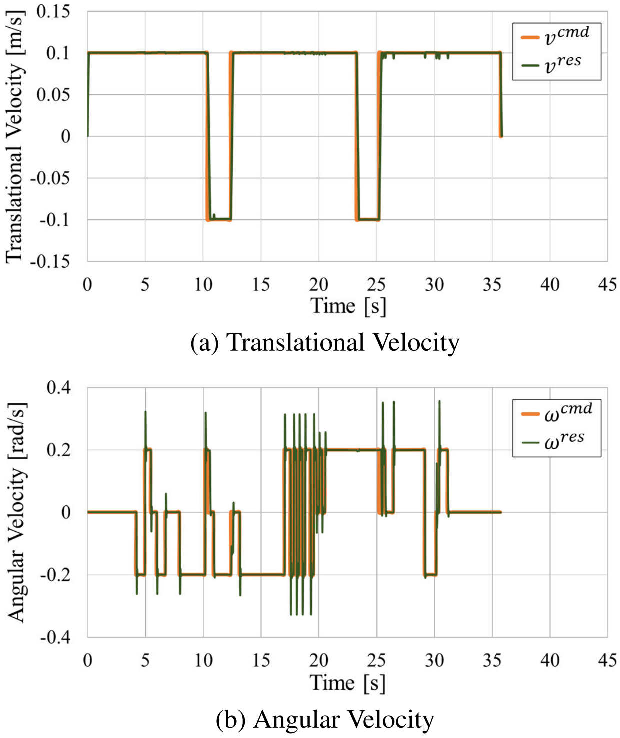
(b) $k \in [42, 50]$

(c) $k \in [94, 102]$

FIGURE 11. Simulation results in Case 3 (proposed method).

TABLE 3. Time to goal or collision.

Episode	T_{goal} [s]	T_{coll} [s]
5000	–	11.19
10000	–	25.10
15000	39.78	–
20000	35.80	–



(a) Translational Velocity

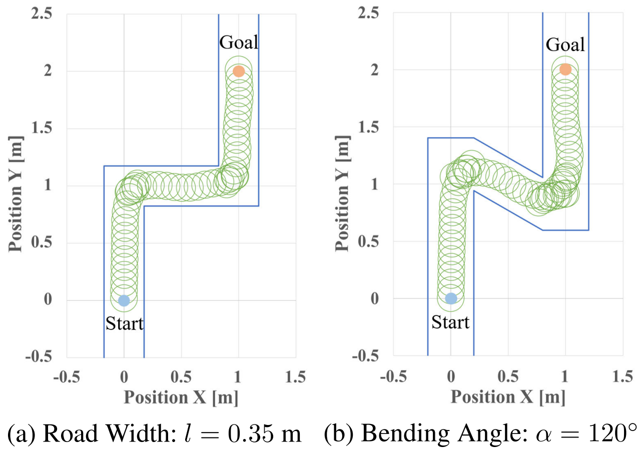
(b) Angular Velocity

FIGURE 12. Velocity response in Case 3.

represented by (x_{sub1}, y_{sub1}) and (x_{sub2}, y_{sub2}). The angles of the subgoals are denoted by θ_{sub1} and θ_{sub2}. In Fig. 10(b), the red line represents the shortest path generated by the Reeds-Shepp curve. The robot moved along the path at 0.1m/s. The robot, however, collided with the obstacles at $(0.332, -0.086)$.

Path planning was carried out in Case 3 using the proposed method. The NT-DQN model was used to confirm the effectiveness of the proposed method under the same road conditions as the learning environments after 20000 episodes in the first and second steps of the learning phase. The robot moved with a turnabout at each corner of the road, as shown in Figs. 11 and 12(a). Figs 11(b)-(c) show the robot's path when turning back at each corner. As a result, the robot arrived at the desired location on the narrow road. Furthermore, as shown in Fig. 12, the velocity responses v^{res} and ω^{res} followed the velocity commands v^{cmd} and ω^{cmd}. The superscripts $\bigcirc^{cmd}$ and $\bigcirc^{res}$ mean the command value and the response value.

The simulation times for Case 3 are listed in Table 3. T_{goal} is the amount of time required to reach a specific position. T_{coll} is the time required to collide or come to a halt. As shown in Table 3, the trained model for path planning with turnabouts was obtained through the DQN learning process. Furthermore, as the number of trials increased, the time required to reach the target position decreased. As a result, the robot achieved path planning by taking turnabouts into account.



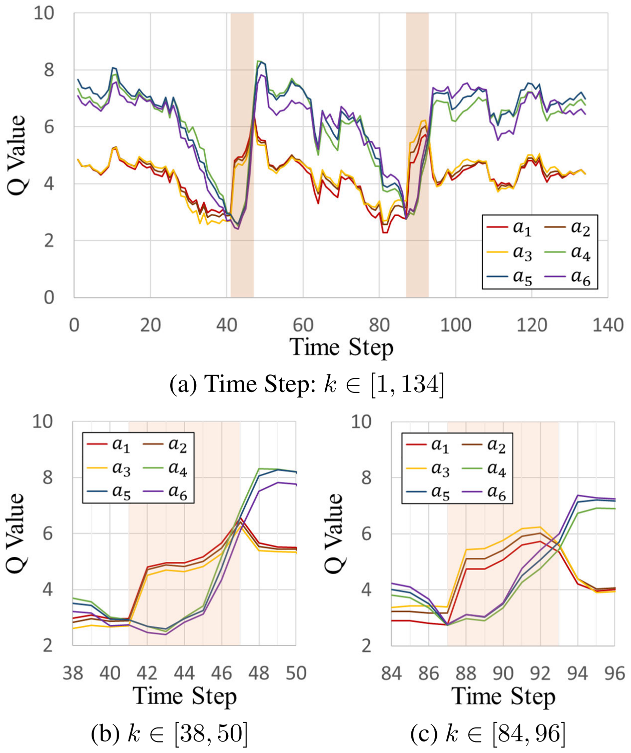
(a) Road Width: $l = 0.35$ m

(b) Bending Angle: $\alpha = 120°$

FIGURE 13. Simulation results in untrained environments.

2) PERFORMANCE EVALUATION OF PROPOSED METHOD

The simulation results for several untrained environments are shown in Fig. 13. The NT-DQN model developed from 30000 episodes during the learning phase was used to validate the proposed method under road conditions that differed from the trained environment. The road width in Fig. 13(a) was set to 0.35 m, which was narrower than the trained environment. The bending angle of the road in Fig. 13(b) was set to 120°, which was greater than in the trained environments. Under these road conditions, the trained model generated paths to the target position, including turnabouts, as shown in Fig. 13.

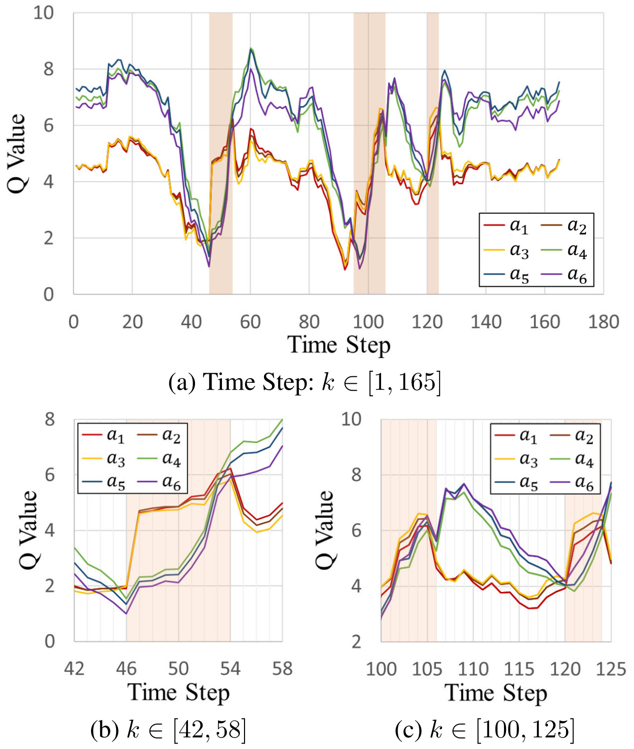


(a) Time Step: $k \in [1, 134]$

(b) $k \in [38, 50]$

(c) $k \in [84, 96]$

FIGURE 14. Transition of Q-value regarding result in Fig. 13(a).



(a) Time Step: $k \in [1, 165]$

(b) $k \in [42, 58]$

(c) $k \in [100, 125]$

FIGURE 15. Transition of Q-value regarding result in Fig. 13(b).

Fig. 14 depicts the Q-value transition for each action in relation to the simulation results in Fig. 13(a). The time step is represented by the horizontal axis, and the Q-value is represented by the vertical axis. As shown in Table 2, a_l $(1 \leq l \leq 6)$ denotes the velocity candidate. The red hatched areas represent the time when the robot retreated. The trained model exhibits the following tendencies in Fig. 14.

1) Except when turning back, the forward motion had a higher Q-value than the backward motion. As illustrated in Fig. 14(a), the trained model preferentially selects forward motion.
2) As shown in Figs. 11(b)-(c), the relationship between the Q-value for a backward motion and the Q-value for a forward motion was reversed just before the robot collided with the wall, as shown in Figs. 14(b)-(c). This implies that the trained model chooses to move backward to avoid collisions.
3) As shown in Figs. 14(b)-(c), the Q-value for backward motion was consistently greater than that for forward motion while turning back. This means that the trained model will continue to choose a backward motion based on sensor data in the backward area in order to reduce the number of turnabouts.

Fig. 15 depicts the transition of the Q-value in relation to the simulation result in Fig. 13(b). The same tendencies described in sections I)-3) are seen in Fig. 15. Furthermore, the results in Fig. 15 indicate that the trained model adapted to an environment with a steep bending angle when compared to the learning environments. The trained model generated backward motion for a greater number of time steps than the result depicted in Fig. 14. According to road conditions, the number of turnabouts increased. We confirmed that the features of the trained model for path planning on narrow roads can be extracted using DQN based on these results.

TABLE 4. Evaluation of simulation result for trained model.

α [°]	Success Rate (Number of Turnabouts)			
	l [m]			Total
	0.45	0.4	0.35	
75	1.00 (1.00)	1.00 (1.00)	1.00 (2.00)	1.00 (1.33)
90	1.00 (2.00)	1.00 (2.00)	1.00 (2.00)	1.00 (2.00)
105	1.00 (2.00)	1.00 (2.20)	0.96 (2.72)	0.99 (2.31)
120	0.50 (3.80)	0.68 (3.88)	0.34 (3.58)	0.51 (3.75)
Total	0.88 (2.20)	0.92 (2.27)	0.83 (2.58)	0.87 (2.35)

Table 4 displays the average success rate and number of turnabouts for 50 trials per road condition. The grey hatched areas represent the trained width (0.4 m, 0.45 m) and bending angle (90°). The success rate and number of turnabouts on the road with the untrained width (0.35 m) were the same as in trained environments when α was set to 90°. This means that the NT-DQN model learns features to account for variations in road width. Furthermore, as the bending angle of the road increased for each road width, the trained model increased the

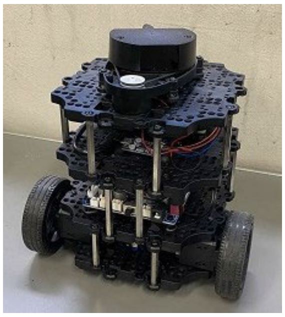


FIGURE 16. TurtleBot3 burger.

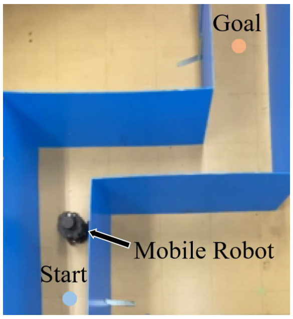


FIGURE 17. Experimental environment.

number of turnabouts. This means that the trained model can generate paths, including turnabouts, based on the bending angle. When α was set to 120°, however, the success rate decreased when compared to the other road conditions of the bending angle. This was due to missing the target direction from the road's corner. This paper does not discuss the results for road conditions where α is less than 75° because the robot traveled on roads without turnabouts.

VI. EXPERIMENT

This section displays the experimental results, which confirm the efficacy of the proposed method.

A. EXPERIMENTAL SETUP

The experiment was carried out with the trained model obtained from 30000 simulation episodes. For the experiment, the simulation parameters listed in Table 1 were used. The mobile robot used in this experiment is depicted in Fig. 16 as the TurtleBot3 Burger. The LRF was attached to a robot. Fig. 17 depicts the experimental settings. The environmental settings were identical to those used in the simulation environment, as shown in Fig. 9. The road conditions (l, α) for the trained environment were set to (0.4, 90), while for the untrained environments it was (0.35, 90), (0.4, 120).

The proposed method's performance was experimentally validated. To the best of our knowledge, providing a theoretical explanation of path planning performance using a learning-based approach is difficult. As a result, the majority of papers on the learning-based approach that have been published have experimentally evaluated trained models [28], [29], [30], [31], [32], [33], [34], [35], [36], [37]. The success rate and number of turnabouts were used as indexes in this study to assess the robustness of the trained model.

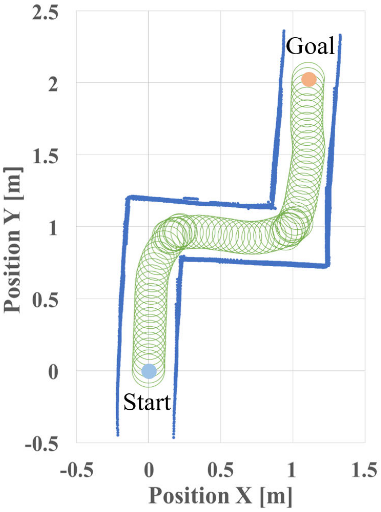
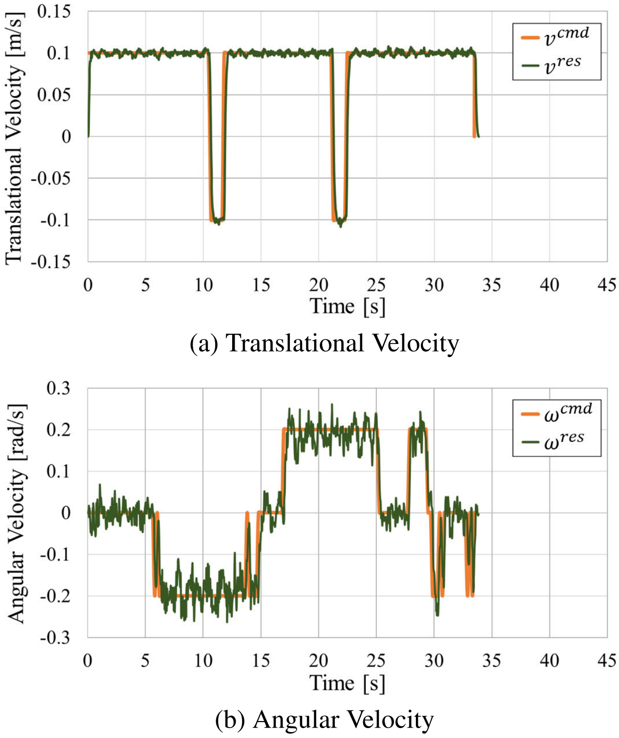


FIGURE 18. Experimental results in trained environments.



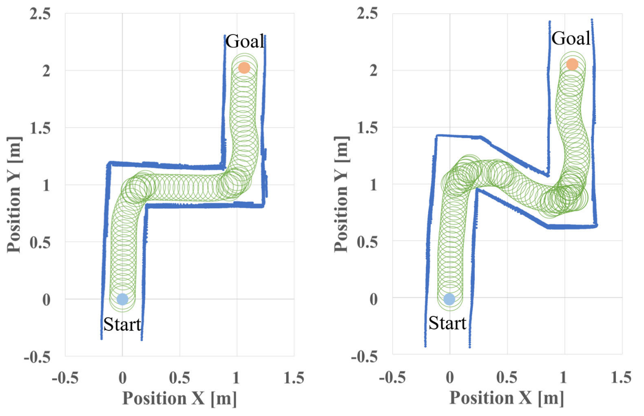
(a) Translational Velocity

(b) Angular Velocity

FIGURE 19. Velocity response regarding result in Fig. 18.

B. EXPERIMENTAL RESULTS

The experimental results for the trained environment are shown in Fig. 18. The green circles and blue spots represent the LRF's measurements of the robot's trajectory and environment. Fig. 19 depicts the velocity response in relation to



(a) Road Width: $l = 0.35$ m (b) Bending Angle: $\alpha = 120°$

FIGURE 20. Experimental results in untrained environments.

the results in Fig. 18. The robot traveled with a turnabout at each corner of the road, as shown in Figs. 18 and 19(a). Furthermore, as shown in Fig. 19, the translational and angular velocity responses followed the velocity commands. The average success rate and number of turnabouts for the five trials were 1.00 and 2.20, respectively.

The experimental results for the untrained environments are shown in Fig. 20. The road width in Fig. 20(a) was set to 0.35m, which was narrower than the trained environment. As shown in Fig. 20(a), the robot traveled with a turnabout at each corner of the road. The average success rate and number of turnabouts for the five trials were 1.00 and 2.20, respectively. The bending angle of the road was set to 120°, as shown in Fig. 20(b), which was greater than that in the trained environments. As shown in Fig. 20(b), the robot traveled four times with turnabouts. For the five trials, the average success rate and number of turnabouts were 0.60 and 3.60, respectively. Based on these findings, the trained model implements path planning by incorporating real-world turnabouts. Furthermore, other road conditions not depicted in this study confirmed results that were similar to the simulation. The experimental results confirmed the validity of the proposed method.

VII. CONCLUSION

This study proposed a path planning method for a nonholonomic mobile robot that takes turnabouts on narrow roads into account. Using a learning-based approach, the proposed method generates a trained model for path planning on narrow roads. In real time, the trained model determined the velocity commands based on sensor data and previous velocities. Simulations and experiments were used to validate the proposed method's effectiveness.

The future works are summarized as follows.

- The robot may be unable to travel in environments that differ significantly from the learning environment, such as branching and blind roads. These are environments where the LRF cannot detect a road. Future work will improve the architecture of the learning model and learning process to improve performance even further.
- Because the proposed method used a discrete velocity command, the velocity command changed abruptly. In the future, RL algorithms designed for learning continuous actions will be used to control robot velocities continuously.
- The proposed method can be improved to avoid losing the target at a road bend. The target data is combined with a map-based path planning method.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Jun. 2016.

[2] F. Zaccaria, A. Baldassarri, G. Palli, and M. Carricato, "A mobile robotized system for depalletizing applications: Design and experimentation," *IEEE Access*, vol. 9, pp. 96682–96691, 2021.

[3] M. A. K. Niloy, A. Shama, R. K. Chakrabortty, M. J. Ryan, F. R. Badal, Z. Tasneem, M. H. Ahamed, S. I. Moyeen, S. K. Das, M. F. Ali, M. R. Islam, and D. K. Saha, "Critical design and control issues of indoor autonomous mobile robots: A review," *IEEE Access*, vol. 9, pp. 35338–35370, 2021.

[4] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109544–109554, 2019.

[5] H. S. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, pp. 353–365, 2022.

[6] J. Zhang, Y. Lu, L. Che, and M. Zhou, "Moving-distance-minimized PSO for mobile robot swarm," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9871–9881, Sep. 2022.

[7] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Trans. Hum.-Mach. Syst.*, vol. 46, no. 1, pp. 9–26, Feb. 2016.

[8] J. Zhang, H. Chen, S. Song, and F. Hu, "Reinforcement learning-based motion planning for automatic parking system," *IEEE Access*, vol. 8, pp. 154485–154501, 2020.

[9] S. Ma, H. Jiang, M. Han, J. Xie, and C. Li, "Research on automatic parking systems based on parking scene recognition," *IEEE Access*, vol. 5, pp. 21901–21917, 2017.

[10] Z. Lu, M. Lin, S. Wang, Y. Zhang, and Y. Yu, "Research on a new-style under-actuated omnidirectional mobile robot based on special coupling drive system," *IEEE Access*, vol. 7, pp. 152138–152148, 2019.

[11] K. Tadakuma, E. Takane, M. Fujita, A. Nomura, H. M. K. Konyo, and S. Tadokoro, "Planar omnidirectional crawler mobile mechanism—Development of actual mechanical prototype and basic experiments," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 395–402, Jan. 2018.

[12] C. Cai, J. Lu, and Z. Li, "Kinematic analysis and control algorithm for the Ballbot," *IEEE Access*, vol. 7, pp. 38314–38321, 2019.

[13] R. Parween, A. V. Le, Y. Shi, and M. R. Elara, "System level modeling and control design of hTetrakis—A polyiamond inspired self-reconfigurable floor tiling robot," *IEEE Access*, vol. 8, pp. 88177–88187, 2020.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, pp. 500–505.

[15] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[16] J. Ng and T. Bräunl, "Performance comparison of bug navigation algorithms," *J. Intell. Robotic Syst.*, vol. 50, no. 1, pp. 73–84, Aug. 2007.

[17] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022.

[18] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.

[19] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1124–1138, Oct. 2017.

[20] A. S. Lafmejani, H. Farivarnejad, and S. Berman, "Adaptation of gradient-based navigation control for holonomic robots to nonholonomic robots," *IEEE Robot. Autom. Lett.*, vol. 6, no. 1, pp. 191–198, Jan. 2021.

[21] T. Xu, S. Zhang, Z. Jiang, Z. Liu, and H. Cheng, "Collision avoidance of high-speed obstacles for mobile robots via maximum-speed aware velocity obstacle method," *IEEE Access*, vol. 8, pp. 138493–138507, 2020.

[22] Z. Li, J. Deng, R. Lu, Y. Xu, J. Bai, and C.-Y. Su, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 6, pp. 740–749, Jun. 2016.

[23] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors*, vol. 21, no. 23, p. 7898, Nov. 2021.

[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[25] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that Goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.

[26] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Tech. Rep., Oct. 1998, pp. 98–11.

[27] A. Choudhary, Y. Kobayashi, F. J. Arjonilla, S. Nagasaka, and M. Koike, "Evaluation of mapping and path planning for non-holonomic mobile robot navigation in narrow pathway for agricultural application," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2021, pp. 17–22.

[28] C.-Y. Tsai, H. Nisar, and Y.-C. Hu, "Mapless LiDAR navigation control of wheeled mobile robots based on deep imitation learning," *IEEE Access*, vol. 9, pp. 117527–117541, 2021.

[29] R. Akabane and Y. Kato, "Pedestrian trajectory prediction based on transfer learning for human-following mobile robots," *IEEE Access*, vol. 9, pp. 126172–126185, 2021.

[30] B. Liu, X. Xiao, and P. Stone, "A lifelong learning approach to mobile robot navigation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1090–1096, Apr. 2021.

[31] J. Yuan, H. Wang, C. Lin, D. Liu, and D. Yu, "A novel GRU-RNN network model for dynamic path planning of mobile robot," *IEEE Access*, vol. 7, pp. 15140–15151, 2019.

[32] G. Farias, G. Garcia, G. Montenegro, E. Fabregas, S. Dormido-Canto, and S. Dormido, "Reinforcement learning for position control problem of a mobile robot," *IEEE Access*, vol. 8, pp. 152941–152951, 2020.

[33] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Auto. Robots*, vol. 45, no. 1, pp. 51–76, Jan. 2021.

[34] H. Sun, W. Zhang, R. Yu, and Y. Zhang, "Motion planning for mobile robots—Focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69061–69081, 2021.

[35] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4423–4430, Oct. 2018.

[36] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active SLAM based on deep reinforcement learning under unknown environments," *Intell. Service Robot.*, vol. 13, pp. 263–272, Jan. 2020.

[37] Q. Yao, Z. Zheng, L. Qi, H. Yuan, X. Guo, M. Zhao, Z. Liu, and T. Yang, "Path planning method with improved artificial potential field—A reinforcement learning perspective," *IEEE Access*, vol. 8, pp. 135513–135523, 2020.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[39] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 1, pp. 47–62, Feb. 1996.

[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[42] *TurtleBot3 Burger*. Accessed: Feb. 24, 2023. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/features/

TOMOAKI NAKAMURA received the B.E. degree in marine engineering from Kobe University, Japan, in 2021. He is currently pursuing the M.E. degree with the Graduate School of Maritime Sciences, Kobe University, Japan. His current research interests include machine learning, robotics, and motion control.

MASATO KOBAYASHI (Member, IEEE) received the bachelor's and master's degrees in maritime sciences and the Ph.D. degree in engineering from Kobe University, Japan, in 2017, 2019, and 2022. From 2019 to 2021, he was a Engineer and a Researcher with the Technology Development Division, Seiko Epson Corporation, Japan. From 2021 to 2022, he was a Research Intern with OMRON SINIC X Corporation, Japan. Since 2022, he has been with Kobe University, where he is currently an Academic Researcher. He is also with the Yutaka Matsuo Laboratory, The University of Tokyo, Tokyo, Japan, where he researches and develops robotics and AI. His current research interests include robotics, mechatronics, motion control, haptics, and artificial intelligence.

NAOKI MOTOI (Member, IEEE) received the B.E. degree in system design engineering and the M.E. and Ph.D. degrees in integrated design engineering from Keio University, Japan, in 2005, 2007, and 2010, respectively. In 2007, he joined the Partner Robot Division, Toyota Motor Corporation, Japan. From 2011 to 2013, he was a Research Associate with Yokohama National University, Japan. From 2019 to 2020, he was a Visiting Professor with the Automation and Control Institute (ACIN), TU Wien, Austria. Since 2014, he has been with Kobe University, where he is a currently an Associate Professor. His current research interests include robotics, motion control, and haptics.