Intelligent Service Robotics (2021) 14:663–677
https://doi.org/10.1007/s11370-021-00387-2

ORIGINAL RESEARCH PAPER

Reinforcement learning-based dynamic obstacle avoidance and integration of path planning

Jaewan Choi[1,2] · Geonhee Lee[1,3] · Chibum Lee[4]

Received: 10 April 2021 / Accepted: 9 September 2021 / Published online: 6 October 2021

Abstract
Deep reinforcement learning has the advantage of being able to encode fairly complex behaviors by collecting and learning empirical information. In the current study, we have proposed a framework for reinforcement learning in decentralized collision avoidance where each agent independently makes its decision without communication with others. In an environment exposed to various kinds of dynamic obstacles with irregular movements, mobile robot agents could learn how to avoid obstacles and reach a target point efficiently. Moreover, a path planner was integrated with the reinforcement learning-based obstacle avoidance to solve the problem of not finding a path in a specific situation, thereby imposing path efficiency. The robots were trained about the policy of obstacle avoidance in environments where dynamic characteristics were considered with soft actor critic algorithm. The trained policy was implemented in the robot operating system (ROS), tested in virtual and real environments for the differential drive wheel robot to prove the effectiveness of the proposed method. Videos are available at https://youtu.be/xxzoh1XbAl0.

Keywords Mobile robot · Navigation · Collision avoidance · Reinforcement learning · Deep learning

1 Introduction

With the development of sensing and computer technologies, autonomous driving systems, such as autonomous vehicles and mobile robots, have gained large attention of researchers, and experienced increased commercialization. The mobile robots are used in logistics centers as logistics robots, which are in-charge of transportation of goods, and in various other fields such as serving in restaurants and cafes, delivering food to home, and providing information for guidance or cleaning at airports. Very recently, to prevent the spread of COVID-19 infection, the biggest problem today, mobile robots have been started to be used for quarantine purposes such as body temperature measurement and sanitizing. As working mechanism, the mobile robots use internal sensors such as encoder and IMU to measure their speed, posture and travel distance, and external sensors such as lidar and camera to measure the presence and distance of any surrounding walls and obstacles. With these sensors, they explore the environments to locate themselves and autonomously drive to the destination. The goal is that these mobile robots travel safely and quickly to their destination without colliding with any obstacles.

The navigation of these mobile robots typically comprises locating the robots first based on maps followed by generating paths from the robot's current position to the target position. The generated path is then followed by the robot. For new obstacles that are not displayed on the map, the robot itself detects the obstacles and avoids them using the distance measuring sensor, namely lidar, and finally, reaches to the target position. In many working scenarios, the movement of dynamic objects is irregular, and it is a challenge for a mobile robot to avoid the dynamic obstacles while traveling and reach the destination without collision. This is mainly because the mobile robots do not know the behavior of the dynamic obstacles or where the obstacles would finally head to, i.e., the destination. The bypassing of these dynamic obstacles has been studied actively, using both the

✉ Chibum Lee
chibum@seoultech.ac.kr

1 Department of Mechanical Design and Robot Engineering, Graduate School, Seoul National University of Science and Technology, Seoul, Korea

2 Present Address: Hyundai-Rotem Co., Uiwang-si, Korea

3 Present Address: Syscon Co., Inchon, Korea

4 Department of Mechanical System and Design Engineering, Seoul National University of Science and Technology, Seoul, Korea

traditional analytical techniques [1,2] and empirical methods [3–5].

However, the conventional obstacle avoidance methodologies such as dynamic window approach (DWA) [6] and timed elastic band (TEB) [7] are specific for static obstacle avoidance. Therefore, it becomes a challenge to bypass the dynamic obstacles by predicting the next movement of the obstacles, which requires sharing information with dynamic obstacles or detecting a new dynamic object. The velocity obstacle (VO) series method [8–11] has been used by several researchers to address the dynamic obstacle method problem. While it is advantageous for dynamic obstacle avoidance to predict and avoid its next moving position through movement path of the object, the conventional methods have the disadvantage of requiring heavy computing power due to the combination of complex conditions and equations.

Empirical information-based learning methodologies can simplify formulas represented by combinations of complex conditions in traditional methods. Reinforcement learning (RL) has been actively studied because of the advantage that it can solve irregular problems which are otherwise not solvable by traditional methods [12,13]. The representative obstacle avoidance methodologies based on RL include CADRL [14] and MRCA [15]. CADRL aims to drive by avoiding people and requires the process of obtaining information on a person's position, speed, and size for avoidance. Therefore, when a person is detected, it has excellent avoidance performance, and has the advantage of being able to learn from human movements to social rules such as left-hand or right-hand traffic. However, if a person is not detected, avoidance cannot be performed, and since detection is not carried out on non-human dynamic objects, it has a disadvantage that collision cannot be avoided. On the other hand, MRCA can drive without collision in a multi-agent environment, and unlike CADRL, performs the task of predicting and avoiding movements of dynamic obstacles through information on distance from the lidar without detecting the dynamic objects. Therefore, it is possible to avoid various dynamic obstacles, not only specific dynamic objects, and has the advantage of being able to learn quickly because multiple robots learn the policy each as a respective agent. Each robot is applied as a dynamic object with a collision avoidance policy but presented as irregular movement of dynamic objects to the other robots. However, this advantage may be a disadvantage. Since they are all assumed to use the same avoidance policy, it is relatively difficult to avoid dynamic obstacles with different avoidance policies or without avoidance policies. In addition, since it was trained in an environment where dynamic factors were not considered, it has the disadvantage of some differences in performance of driving and obstacle avoidance in the real-world and training environments.

In the context of the above scenario, in the present study, in this study, for decentralized dynamic obstacle avoidance, we propose a method of predicting the next movement through the movement of an obstacle based on RL and avoiding collision. The dynamic object movement was predicted through distance information from lidar without detecting the objects to perform avoidance of various obstacles. Furthermore, to reduce the differences between driving in real and training environments, the policy was trained in the environment where inertia and friction dynamics were considered. In addition, a multi-robot environment was also configured to enable fast learning, and dynamic objects which do not have obstacle avoidance policies other than robots were also placed in the training environment to enable effective avoidance of dynamic obstacles running with other policies. However, RL-based obstacle avoidance alone caused the problem of not finding a path in a specific situation. To tackle this problem and impose the path efficiency, a path planner was integrated with the reinforcement learning-based obstacle avoidance.

We implemented the reinforcement learning-based obstacle avoidance to differential drive mobile robot and performed the task of driving to the target position by avoiding obstacles. To train the policy and evaluate the performance, we constructed simulation environments, and evaluated the efficiency of obstacle avoidance and driving through simulation and conducting real-world experiments. Section 2 describes the differential driving robot used in the present study and the parameters of the model. Section 3 details the reinforcement learning-based collision avoidance method. The state, behavior, reward function, termination condition, training environment, training algorithm are described in detail and the train results along with the problem of the method are also discussed. In Sect. 4, the modified method which includes the path planner has been proposed and explained. Section 5 evaluates the driving and obstacle avoidance performance of the proposed method through real-world experiments in comparison with dynamic window approach (DWA), timed elastic band (TEB). Section 6 contains analysis and discussion of the results. Finally, conclusions are drawn in Sect. 7.

2 Description of mobile robot

The methodology used in the current study can also be applied to build any type of mobile robots, but the differential drive robot, which is widely used for indoor services, was adopted for application and validation. The industrial logistic robot, SR7 (Syscon co., Inchon, Korea), is shown in Fig. 1 and its specifications are listed in Table 1.

The kinematics of the differential drive robot is represented by two-dimensional coordinates as indicated in Fig. 2. The yaw angle ψ represents the robot's travel direction about the x-axis at the center of the two wheels. The velocities of the left and right wheels are v_L and v_R, respectively, while the for-

Fig. 1 Differential drive wheeled SR7 robot

Table 1 Specification of SR7 robot

Proper	Value
Type	Differential drive
Dimension	0.8 × 1.0 × 0.32 m
Weight	150 kg
Wheel radius	0.55 m
Tread	0.75 m
Lidar angle	360 deg
Lidar minimum distance	0.5 m
Lidar maximum distance	20 m
Lidar resolution	0.25 deg

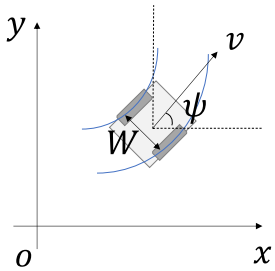


Fig. 2 Coordinate system for a differential drive robot

ward velocity of the mobile robot is given by $v = (v_L + v_R)/2$ and the yaw rate is $\dot{\psi} = \omega = (v_R - v_L)/W$, where W is the wheel tread.

The required rotation speeds of the motors ω_R and ω_L can be obtained from the input of the robot which are the forward velocity v and the yaw rotation velocity ω, under the assumption that there is no slip,

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \frac{1}{R} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{W} & -\frac{1}{W} \end{bmatrix}^{-1} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

where R is the radius of the drive wheels.

The equation of motion for global coordinates is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{2}$$

The kinematics of the robot is used to convert the linear and angular velocity derived from the algorithm presented in this paper to the rotation speed of left and right motor.

3 RL-based dynamic obstacle avoidance

In this section, we will develop an obstacle avoidance model, named as Mobile robot Collision Avoidance Learning (MCAL) based on reinforcement learning. In MCAL, the agent level decentralized collision avoidance paradigm was adopted similar to the previous study multi-agent collision availability (MRCA) [15]. However, to reduce the difference in obstacle avoidance performance between simulation and real-world environments and to achieve high sample efficiency and fast learning speed, MCAL was trained in the environment with dynamics considered using the value-based learning method, soft actor critic (SAC) [16].

Figure 3 shows the framework for MCAL, RL-based obstacle avoidance method with continuous behaviors, and each block performing the following tasks.

- The *robot* obtains the position and speed data (/Odometry) and the lidar data (/Scan) via interactions with the external environment, *world*.
- For *localization* [17], map information (/Map) and lidar data (/Scan) were compared, and the robot's position (/Global Pose) was derived based on the map.
- The relative difference between the target position (/Goal) and the position of the robot based on the map (/Global Pose) is the distance from the robot to the target position (/Relative Goal), which becomes the input of the *RL agent*.
- The *RL agent* gathers information on the distance between the robot and the target position (/Relative Goal), 3 steps of lidar scan data (/Scan), and the speed of the robot (/Odometry.speed), and outputs the forward and rotational velocity (/Velocity) of driving to the target point without collision through the trained deep neural network of reinforcement learning.
- The *Robot* moves by controlling the forward and rotational velocity obtained through the *RL agent*.

This process repeats until the target position is reached.

3.1 Reinforcement learning formulation

3.1.1 State

To formulate the collision avoidance problem within a reinforce learning framework, the state consists of lidar data, which are the distance information from the surrounding environment, the forward velocity v and the rotational velocity ω of the robot, and the relative distances of x and y from the robot to the target position as

$$s_t = [s_t^{\text{lidar}}, s_t^{\text{goal}}, s_t^{\text{speed}}] \tag{3}$$

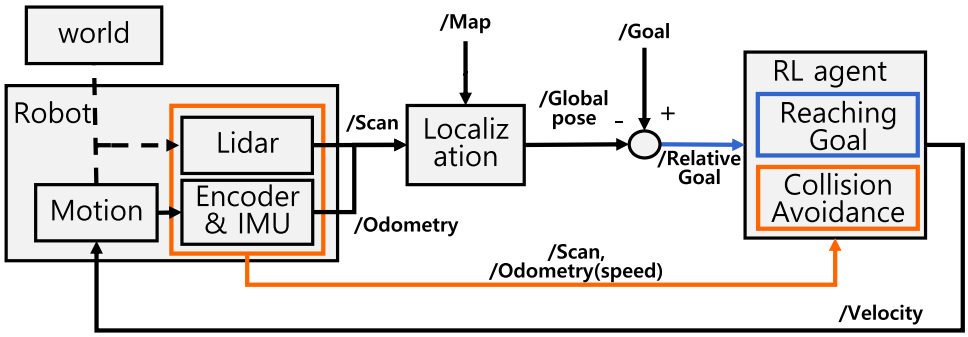


Fig. 3 Structure of mobile robot collision avoidance learning

where $s_t^{\text{lidar}} = [o_1^{t-2}, o_1^{t-1}, o_1^t]$, $s_t^{\text{goal}} = o_g^t = [x, y]$, and $s_t^{\text{speed}} = o_s^t = [v, \omega]$.

s_t^{lidar} are the lidar data which indicate the relationship between the obstacles and the robot by measuring the distance and the recognizing objects. Furthermore, the lidar data on three consecutive time-steps was used to implicitly predict the movement direction and speed of the object. s_t^{goal} is the relative distance from the robot to the target position, and driving direction can be effectively obtained from the clear information provided by s_t^{goal} on whether the moving direction is right. s_t^{speed} provides the velocity information of the robot. Additionally, and the limiting speed and inertia of the robot and avoidance method according to the speed can be known from s_t^{speed}.

3.1.2 Action

The behavior of the mobile robot can be defined as continuous behavior to enable smooth movement and avoidance in various ways, and consists of two-dimensional information with the forward velocity v and the rotational velocity ω as follows.

$$a_t = [v, \omega] \tag{4}$$

where v, ω are continuous values and have the limiting velocity constraints of $v \in [0.00, 0.55]$, $\omega \in [-0.60, 0.60]$.

3.1.3 Reward

In this study, the robot aims to reach the target position (P_x, P_y) and the orientation was not concerned since the differential drive robot can rotate easily in place. To reach the target position, driving with obstacle avoidance through reinforcement learning without colliding with obstacles and remaining within the performance limit is necessary. Therefore, the reward was also considered separately. The total reward function is the sum of these three reward functions,

$$R = R_g + R_c + R_\omega. \tag{5}$$

If the mobile robot reaches the target position, the agent receives a large reward of 10. Additionally, while moving to the target position, if the distance to the target becomes shorter than before, then also reward is given as the robot is moving in the right direction.

$$R_g = \begin{cases} 10 & \text{if } \text{dis}_{\text{curr}} < 0.5 \\ \text{dis}_{\text{pre}} - \text{dis}_{\text{curr}} & \text{otherwise} \end{cases} \tag{6}$$

where the distance is given by

$$dis = \sqrt{(p_x^{\text{goal}} - p_x^{\text{robot}})^2 + (p_y^{\text{goal}} - p_y^{\text{robot}})^2}. \tag{7}$$

Moving in a direction away from the target position will incur a penalty corresponding to the distance traveled in one step, and moving in the direction closer to the target position will result in a reward corresponding to the distance traveled in one step.

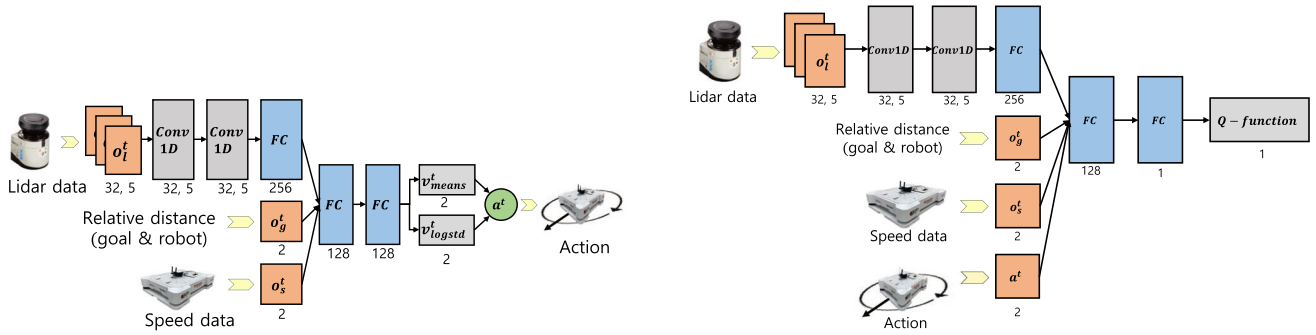
The reward R_c imposes a large penalty of -10 when there is a collision with an obstacle, a policy that the robot needs to be taught to avoid collisions with obstacles.

$$R_c = \begin{cases} -10 & \text{if Collision} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

The last reward R_ω concerns the robot's performance limit as a penalty, instead of constraint equation. When the 150 kg heavy robot SR7 rapidly rotates, it is difficult to control due to inertia, so R_ω imposes a large penalty on rotational velocity beyond the threshold value in order to prevent the problem.

$$R_\omega = \begin{cases} -0.1|\omega| & \text{if } |\omega| > 0.6 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The weight of each reward was determined after trial and error through experiment so that the robot learns the various methods of obstacle avoidance according to the situation, such as avoidance through acceleration and deceleration, avoidance through stopping, and avoidance via directional change.



(a) Actor network **(b)** Critic network

Fig. 4 Actor and critic networks for mobile robot collision avoidance learning

3.1.4 Termination conditions

The termination condition for stopping an episode in the training was designed for 3 cases, namely when the robot reaches the target position, when it collides with an obstacle, and when the number of steps to proceed the episode exceeds 2000.

$$T = \begin{cases} \text{True} & \text{if (Reached goal } or \text{ Collision } or \text{ step} > 2000) \\ \text{False} & \text{otherwise} \end{cases} \tag{10}$$

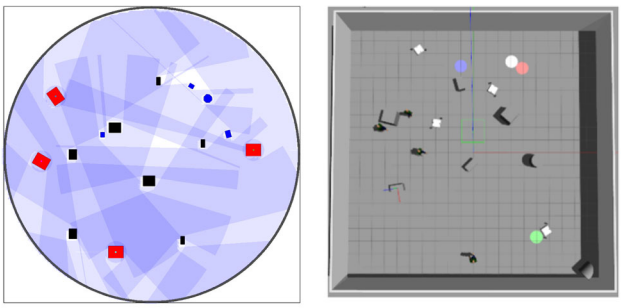
The limit condition of 2000 steps suppresses the tendency to reach the target position by hovering around the target position for a long time, or avoiding the obstacle by moving through an inefficient path. Without such limits, the robot will learn a very safe obstacle avoidance, such as slowing down or stopping, only gaining small R_g for a long time.

3.2 Network

To apply SAC algorithm on collision avoidance, an actor network to derive a policy and a critic network to derive the Q-function for computing the cost of the policy were constructed.

Referring to the network, MRCA [18], the actor network shown in Fig. 4a consisted of two convolution layers, three fully connected layers, two nonlinear activation ReLU functions, and one linear activation function. Among the state in 3, the lidar data 512×3 corresponding to 3 sample times was input and passed through two convolution layers to derive a temporal change.

This information, combined with the relative distance (x, y) from the robot to the target position, and the velocity of the robot (v, ω) passed through two fully connected layers. Finally, the mean velocity v^t_{means} and log standard deviation v^t_{logstd} form a Gaussian distribution of actions. The final action a_t was sampled.



(a) Stage **(b)** Gazebo

Fig. 5 Environments for the Stage and Gazebo simulators for training

The Critic network shown in Fig. 4b is similar to that of the Actor network, but the action obtained from the Actor network was added and the final result obtained is the Q-value.

3.3 Training

3.3.1 Training environment

Two simulators were used for training simulation. The first simulator, i.e., the Stage simulator [19] in Fig. 5a considers only the kinematic factors, so behaviors denoting the dynamic characteristics such as inertia and friction cannot be trained accurately, resulting in significant differences between the real-world and training environments. However, the simulation time can be accelerated due to its lightweight simulator, allowing the advantage of fast learning.

The second simulator is the Gazebo simulator [20] in Fig. 5b. It takes into account the dynamic factors, such as inertia and friction as well as the kinematic factors. The difference between the real-world and training environments also exists for the Gazebo simulator, but the difference can be reduced. It has a disadvantage of longer simulation time. In consideration of these advantages and disadvantages, after

fast training in the Stage simulator, the actor and critic networks were trained again in the Gazebo simulator to reduce the difference between the real-world and the learning environment.

The training environment in the Stage simulator was configured for a circular space with a diameter of 30 m, placing randomly 7 static objects and 4 dynamic objects that do not have any avoidance policy and move in a straight line at random speeds between 0.5 and 1 m/s. In Fig. 5a, the red squares were the robots, blues were dynamic objects, blacks were static objects. We also deployed 4 robots to train the policy simultaneously, and this multi-robot training has the advantage that each robot treats others as dynamic obstacles to learn how to avoid obstacles via various movements. Moreover, since 4 robots gathered the necessary information for training together, it can increase the learning speed. The training environment in the Gazebo simulator was configured for a square space of 20 m by 20 m, placing randomly objects just like the stage environment, and 4 robots were trained with the policy simultaneously. In Fig. 5b, the white boxes were the robots having the goals in red, blue, yellow, and white circles individually. The walking people were dynamic objects, and the gray boxes and cylinders were static objects.

3.3.2 Training algorithm

After trying to train with different algorithms, we adopted SAC to train the weights of deep neural networks.

SAC is an off-policy learning algorithm that can use all the information collected during the training process for deep neural network learning. So, sample-efficient is high. The objective function of SAC is given by

$$J(\theta) = \sum_{t=0}^{T} E_{\pi}[r(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t)]. \tag{11}$$

The entropy function H and the hyper-parameter α, which determines the relative importance of the compensation entropy, have been added to the existing objective function. In the training with SAC, the agent acts more randomly and this should result in effective learning. Since the Gaussian distribution generated is relatively flat and if the optimal policy does not work well in the real environment due to the difference between the training and the real environments, there is a possibility to solve the problem with the next best policy.

In fact, we compared the results after training with the on-policy algorithm, Proximal Policy Optimization (PPO), and it was confirmed that SAC achieves a faster learning speed due to high sample efficiency.

The off-policy algorithm, SAC, requires replay of memory to store the generated state, reward, and action values

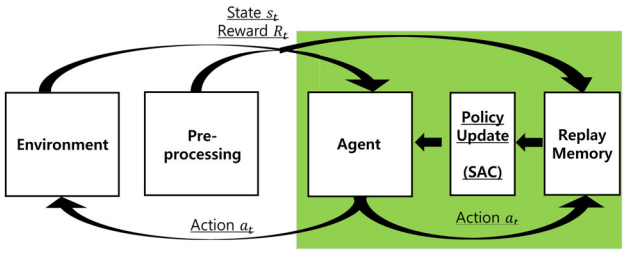


Fig. 6 Training framework for mobile robot collision avoidance learning

Table 2 Hyper-parameters for training in SAC

Parameters	Value
Target smoothing factor τ	0.005
Discount factor γ	0.99
Learning rate α	0.0003
Batch size	1024
Replay memory size	500,000
Target update interval	1

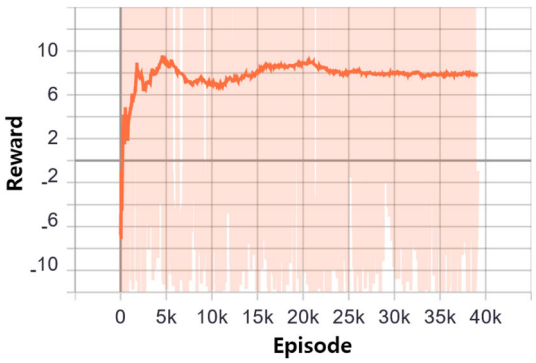
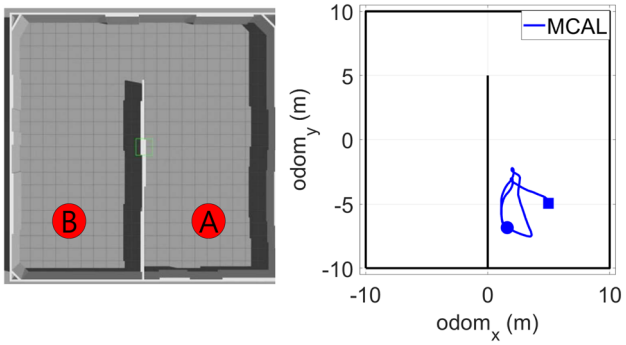


Fig. 7 Reward for the stage simulation training

in every step of the training process, and the related training framework shown in Fig. 6 can improve policy through the information stored in this repository. Pre-processing in Fig. 6 refers to the process of obtaining a global pose from scan data, odometry data, and map data. It transforms information obtained from the environment to match the input of the network.

3.3.3 Train result

The policy was trained using the Stage simulator with the hyper-parameters in Table 2. The reward graph in Fig. 7 confirmed that the policy converged from the 20,000 episodes. Furthermore, the policy with 3000 more episodes was trained using the Gazebo simulator, to reduce the difference between the training environment and real-world environment.



(a) Gazebo environment **(b)** Robot odometry

Fig. 8 Problem with MCAL (square marker: start point, circle marker: end point)

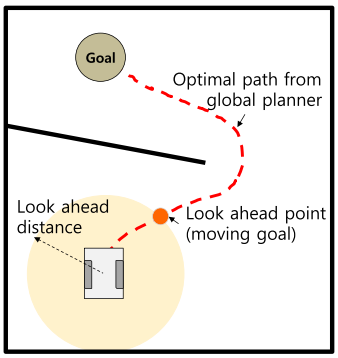


Fig. 9 Look-ahead point and optimal path

3.4 Problem of RL-based dynamic obstacle avoidance

It was confirmed that the robot was trained according to the above process successfully and to move without colliding with any static and dynamic obstacles in open spaces. In particular, it was impressive to respond differently to dynamic and static obstacles. As the dynamic obstacles come near, the forward velocity decreases significantly, the rotational velocity increases significantly, and as the movement of dynamic obstacle slows down, the changes in velocities then decrease.

However, certain problems were found when the robot moved from point **A** to point **B** in a virtual environment, as shown in Fig. 8a.

The movement path of the robot is shown in Fig. 8b, identifying the problem that the robot avoids the wall in the middle, but fails to reach the target point and continues to hover around the wall. This problem was caused by conflicting objectives of the robot to minimize the distance from the target point and to maintain a certain distance from the obstacle while moving according to the proposed method. Thus, according to the action suggested by the trained policy of MCAL, the robot moves in an inefficient path, resulting in the problem of not finding the driving direction under certain circumstances. This occurs in certain situations, such as in the presence of obstacles close to the front of the robot, which can be attributed to a lack of lidar data and a lack of training information in various environments.

These problems can be improved with a lot of training in a variety of environments. Also, previous RL-based robot navigation studies solved this problem through effective exploration. [21,22]. However, it is impossible to learn an optimal path comparable to traditional algorithms for solving optimization problems. The problem-solving through exploration is highly likely to fail to derive the optimal path if the observation value is generated equally for the complex and wide environment. Therefore, we tried to solve these problems by integrating RL-based mobile robot navigation with traditional method.

4 RL-based dynamic obstacle avoidance with optimal path

To solve the problem of MCAL mentioned above, we included classical optimal path planning to obtain an improved MCAL method, so that the robot follows the optimal path which the global path planner generates based on the map. As shown in Fig. 9, we adopted the concept of look head distance from the path-following method PurePursuit [23]. The look-ahead point is a moving point maintaining a certain distance from the robot and replaces the target point in the MCAL input.

In this section, we will propose Mobile robot Collision Avoidance Learning with Path (MCAL_P) that follows the look-ahead point through the trained policy and reaches the target point with collision avoidance.

The overall procedure including path planner and look-ahead point is shown in Fig. 10

- The *robot* obtains the position and speed data (/Odometry) and the lidar data (/Scan) via interactions with the external environment, *world*.
- Fpr *Localization*, map information (/Map) and lidar data (/Scan) were compared, and the robot's position (/Global Pose) was derived based on the map.
- *Path Planning* such as A* [24] algorithm gives global optimal path from the robot's location to the target point (/Path) from the input of map information (/Map), target position (/Goal), and the robot's location (/Global pose).
- *Find Look Ahead Point* finds a look-ahead point (/Look Ahead Point), a point away from the robot by the look-ahead distance in the path, based on the input of the optimal path (/Path) and the position of the robot based on the map (/Global pose).

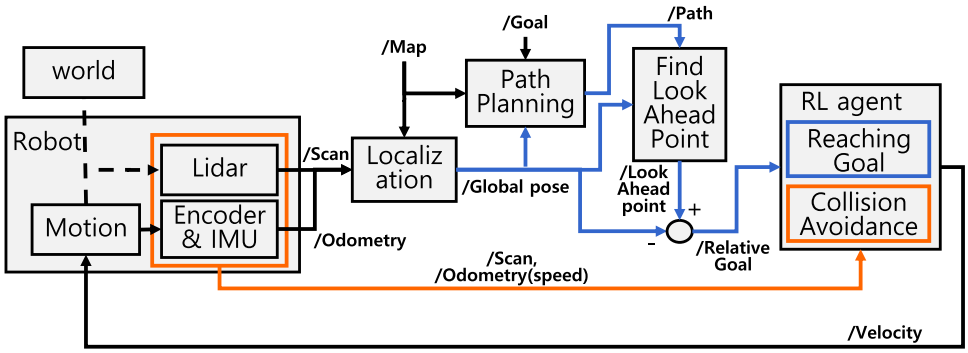


Fig. 10 The MCAL_P framework

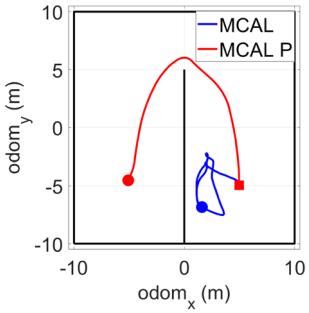


Fig. 11 Robot odometry MCAL vs. MCAL_P (square marker: start point, circle marker: end point)

– The relative difference between the look-ahead point on the path (/Look Ahead Point) and the position of the robot based on the map (/Global pose) is the distance to the look-ahead point (/Relative Goal), which becomes the input of the *RL agent*.
– The *RL agent* receives the robot to the look-ahead point (/Relative Goal), 3 steps of lidar data (/Scan), the speed of the robot (/Odometry.speed) and outputs the forward and rotational velocity (/Velocity) for driving to the look-ahead point without collision through the trained deep neural network of reinforcement learning.
– The *Robot* moves by controlling the forward and rotational velocity obtained through the *RL agent*.

Without new training, the RL agent described in Sect. 3 was reused.

4.1 Improvement of the MCAL problem

To check whether the problem of MCAL introduced in Sect. 3 was solved using MCAL_P, the test in Sect. 3.4 was conducted in the same manner.

Figure 11 compares the movement paths of the mobile robot using MCAL and MCAL_P. As intended, the MCAL_P drove effectively in environments where MCAL failed by obtaining a favorable direction for driving through the optimal path.

Table 3 Common parameters for DWA and TEB

Parameters	Value
Footprint	1.15×0.8 m^2
Obstacle range	6.0 m
Raytrace range	10.0 m
Inflation radius	0.2 m
Local costmap size	6×6 m^2
Maximum forward velocity	0.55 m/s
Minimum forward velocity	0.0 m/s
Maximum rotational velocity	0.6 rad/s
Minimum rotational velocity	0.6 rad/s

5 Simulation and real-world experiments

5.1 Real-world experiments in standardized environments

The proposed method was implemented based on Navigation stack [25] in Robot Operating System (ROS) [26]. Furthermore, we also implemented DWA and TEB, the two most commonly used conventional non-communicating obstacle avoidance driving methods. We compared the obstacle avoidance success, travel time, driving path, computation time, and velocities of the proposed method through driving experiments in the same environment as those of TEB and DWA.

The common parameters required for DWA and TEB are shown in Table 3, the other parameters required for DWA are listed in Table 4, and the other parameters required for TEB are put in Table 5. For other parameters, the default values provided by the Navigation stack in the ROS were used. The parameters of each algorithm were set to achieve same performance as much as possible in straight driving without obstacles. However, it was difficult to make them completely identical due to structural differences.

Table 4 Parameters for DWA

Parameters	Value
Time to forward-simulate trajectories	5.0 s
Number of samples in the x velocity space	8
Number of samples in the y velocity space	0
Number of samples in the θ velocity space	8
Weight for staying close to the path	32.0
Weight for reaching its local goal	20.0
Weight for avoiding obstacles	0.02

Table 5 Parameters for TEB

Parameters	Value
Desired temporal resolution of the trajectory	0.3 s
Hysteresis for automatic resizing	0.1
Minimum number of samples	3
Minimum obstacle distance	0.6 m
Inflation distance	0.6 m
Safety margin to penalty functions	0.1
Weight for maximum forward velocity	2.0
Weight for maximum rotational velocity	1.0
Weight for maximum forward acceleration	1.0
Weight for maximum rotational acceleration	1.0
Weight for non-holonomic kinematics	1000.0
Weight for forward direction	1.0
Weight for execution time	1
Weight for distancing from obstacles	50

5.1.1 Experimental environment and methods

The real-world experiments were conducted by configuring 6 real-world environments as shown in Fig. 12. In each environment, the mobile robot aims to avoid obstacles in the path while driving, with 10 round trips from position A to position B, with MCAL_P, DWA, and TEB, respectively. MCAL_P has not trained in these 6 environments and this experiment would check the generalization of the policy trained in the simulator.

Map A Empty environment without objects (Fig. 12a)
Map B Environment where 3 static objects are placed at 4 m intervals (Fig. 12b)
Map C Environment where 3 static objects are placed at 2 m intervals (Fig. 12c)
Map D Environment where two dynamic objects reciprocate in a direction perpendicular to the robot's moving direction (Fig. 12d)
Map E Environment where two dynamic objects reciprocate in a direction parallel to the robot's moving direction (Fig. 12e)

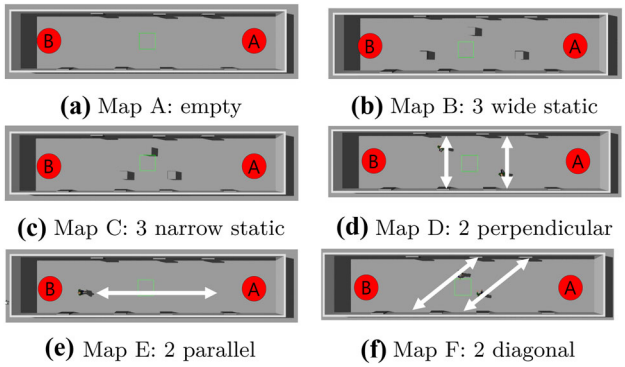


Fig. 12 Environments for performance test(https://youtu.be/xxzoh1XbAl0?t=25)

Map F Environment where two dynamic objects reciprocate in a direction diagonal to the robot's moving direction (Fig. 12f)

5.1.2 Experimental results

Table 6 shows the probability of obstacle avoidance and the results of the average travel time for 10 runs. The experiment results were analyzed separately for the success rate of obstacle avoidance, travel time, computation time, trajectory, and driving tendency.

Success rate Table 6 shows that MCAL_P, DWA, and TEB performed well in the Map B and Map C environments that are related to static obstacle avoidance, and relatively well in the Map D and Map F environments which are related to dynamic obstacle avoidance. However, in Map F, where dynamic objects move in parallel directions of the robot's progression, the DWA failed to evade, while the TEB succeeded in avoiding only two out of ten times. In all environments, the MCAL_P showed a significantly higher probability of dynamic obstacle avoidance compared to the DWA and TEB.

Travel time Overall, the MCAL_P showed faster driving in all environments than the DWA and TEB. This is because DWA and TEB are always subject to acceleration and deceleration by speed profile. The MCAL_P does not have a speed profile and only accelerates and decelerates in certain situations by the trained policy, which seems to result in faster driving. However, in narrow road environments such as Map C, it was found that the robot was driving slower than DWA and TEB.

Computation time The time taken to derive the forward and rotational velocity, which are the outputs of each method were compared using the same PC Intel i9-9900K with 32 GB ram and Geforce GTX 1080Ti. To derive the required velocity for the situation, MCAL_P took the fastest time,

Table 6 Success rate and travel time for MCAL_P performance test

Map	MCAL_P		DWA		TEB	
	Success	Time (s)	Success	Time (s)	Success	Time (s)
A	**10/10**	**28.1**	10/10	29.0	10/10	32.3
B	**10/10**	**32.3**	10/10	35.7	10/10	36.9
C	9/10	40.1	**9/10**	**33.0**	9/10	39.9
D	**9/10**	**34.8**	7/10	39.3	9/10	38.7
E	**10/10**	**29.7**	0/10	–	2/10	36.4
F	**10/10**	**36.1**	9/10	39.6	9/10	40.8

Bold values indicate the best performance in each map

0.00745 s, the TEB 0.015 s, and the DWA the slowest time, 0.028 s.

Trajectory Figure 13 shows the trajectories of the robot in each map. In the Map A environment without an obstacle (Fig. 13a), MCAL_P traveled with a relatively large y-axis movement compared to DWA and TEB, indicating that it is very difficult for MCAL_P to drive straight. This will be discussed further in the next section. When avoiding static obstacles in Map B (Fig. 13b) and Map C (Fig. 13c) environments, the MCAL_P took a relatively large path to avoid obstacles. Such avoidance of the obstacles which suddenly started moving was expected during the training process of MCAL_P. In Map D environment with dynamic obstacles, it can be seen that the path of MCAL_P in Fig. 13d has a larger movement along the y-axis compared to DWA and TEB.

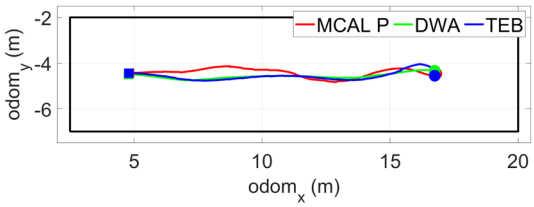
Such characteristics of these driving trajectories were observed probably because the MCAL_P avoided obstacles by various methods such as stopping and turning to avoid obstacles depending on the situation. On the other hand, the DWA and TEB tended to stop first to avoid obstacles and waited for them to pass before driving. This avoidance tendency can also be confirmed through the trajectories in the Map E environment. In Fig. 13e, the DWA and TEB collided with an obstacle at point (− 4.5, 11) and moved toward the target point after the obstacle moved away. The trajectories of DWA and TEB in Map E in Fig. 13e are almost identical to those of Map A. This means that DWA and TEB were not able to adopt different behaviors from the obstacle-free environment Map A for the obstacle coming forward and avoid the collision by stopping and waiting for the obstacle to pass away. However, the MCAL_P rapidly moved in the y-axis direction at point (− 4.5, 11), which was close to the obstacle, and by changing direction, avoided the collision with the approaching obstacle from the front. Due to the difference in these avoidance methods, the DWA and TEB cannot avoid dynamic obstacles with various movements, but the MCAL_P can avoid dynamic obstacles with various movements in various ways.

Driving tendency Figures 14 and 15 show the graphs of the forward and the rotational velocity for six environments. When avoiding static obstacles in Figs. 14 and 15b, c, the DWA and TEB avoided the obstacles by accelerating and decelerating the speed without stopping; however, to avoid the dynamic obstacles in Figs. 14 and 15d–f, the DWA and TEB avoided obstacles simply by stopping. Due to this avoidance method, the dynamic obstacles moving ahead of the robot cannot be avoided. Unlike the DWA and TEB, the MCAL_P avoided obstacles by moving, changing directions, or accelerating/decelerating speed depending upon the situation without distinguishing whether the obstacle is static or dynamic. The MCAL_P can avoid obstacles with varying motion, but in narrow sections such as Map C. The forward velocity graphs in Fig. 14c show a large variation and the tendency to steer after stopping to avoid obstacles. As a result, the driving performance is degraded by performing the avoidance through a somewhat inefficient path when avoiding static obstacles. Interestingly, at the start of the driving in Fig. 14, the MCAL_P and other algorithms have different acceleration. The TEB and DWA have acceleration limitations but the actual acceleration on the start was much smaller than the limit. While moving, the DWA and TEB can accelerate and decelerate faster than staring and this can be seen by the sharp drop in Fig. 14c–e. We consider that these factors are characteristic of the algorithm.

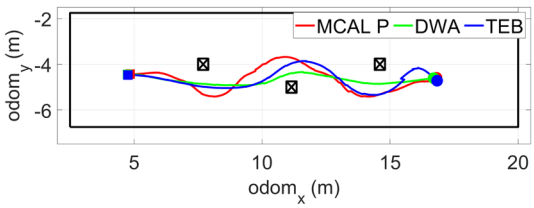
In addition, the rotational velocity graph in Fig. 15 shows that the MCAL_P rotates left and right repeatedly and moves with great shaking. Such movements occur even in situations where there are no obstacles, such as in Fig. 13a, and is responsible for the robot's failure to move along a straight line. Since moving both along a curved path and a straight one is not performed well, the robot cannot travel along the optimal path. Therefore, the robot always draws an inefficient path when driving. This is the disadvantage of MCAL_P.

5.2 Real-world experiment in real working environment

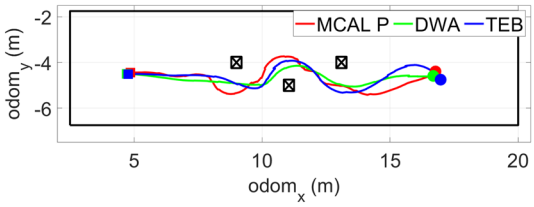
To identify the possibility of obstacle avoidance driving in spaces such as real collaborative factories and parks, we constructed an environment in which a large number of people moved and the robot proceeded via driving using the MCAL_P methodology.



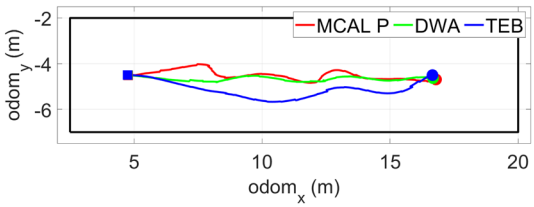
(a) Map A: empty



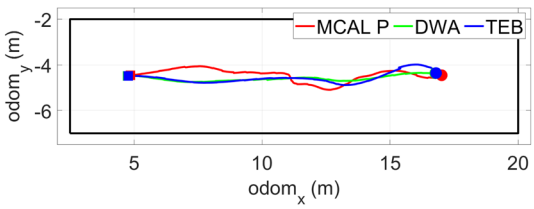
(b) Map B: 3 wide static



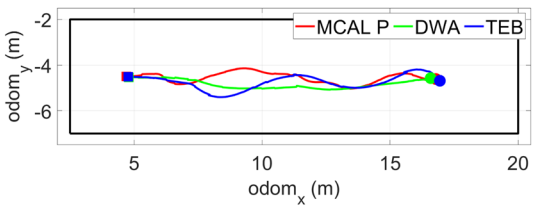
(c) Map C: 3 narrow static



(d) Map D: 2 perpendicular

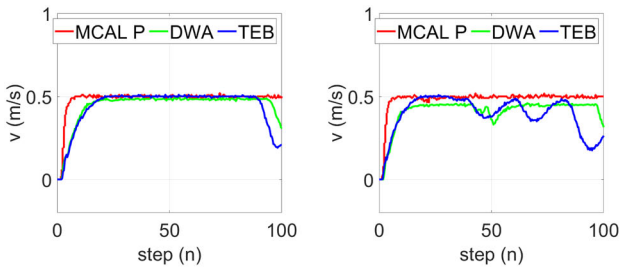


(e) Map E: 2 parallel



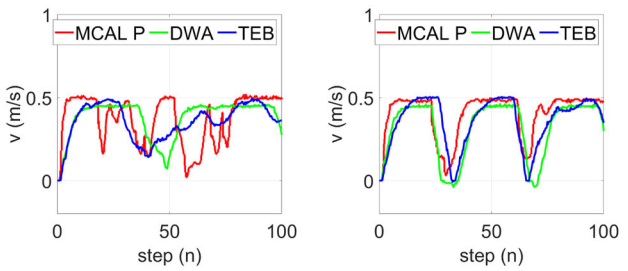
(f) Map F: 2 diagonal

Fig. 13 Robot odometry for performance test (square marker: start point, circle marker: end point)



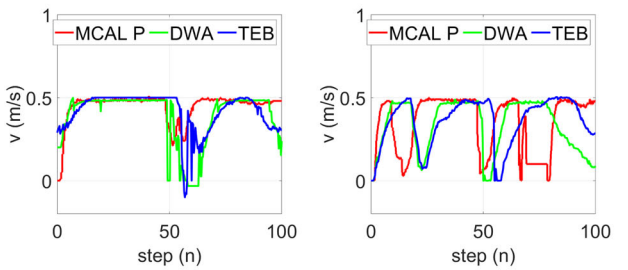
(a) Map A: empty

(b) Map B: 3 wide static



(c) Map C: 3 narrow static

(d) Map D: 2 perpendicular



(e) Map E: 2 parallel

(f) Map F: 2 diagonal

Fig. 14 Forward velocity of the robot

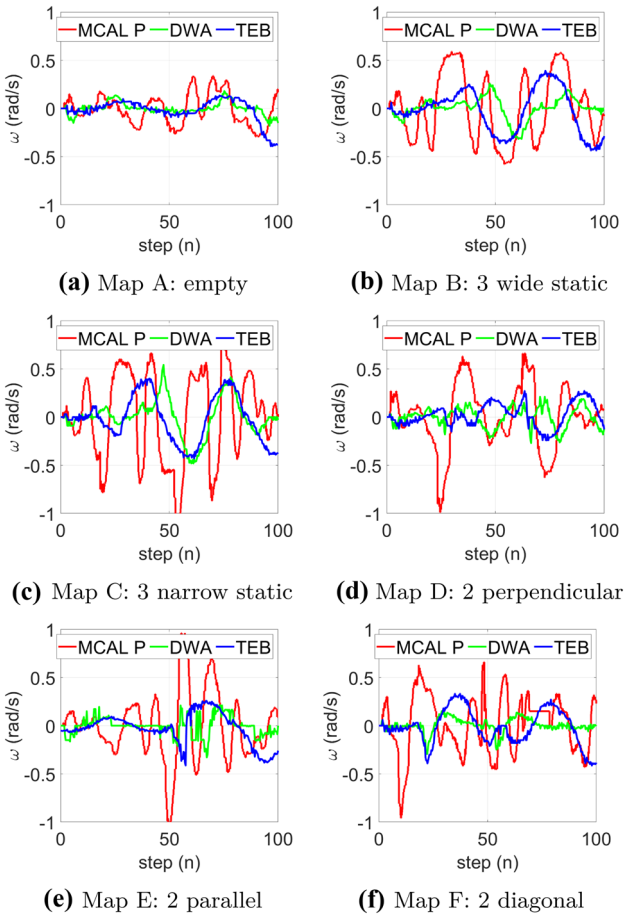
The robot started from position A in Fig. 16 and reached the final target point D via positions B and C. In this environment, there were 4 static obstacles and 7 dynamic obstacles (see Fig. 17).

It could be seen that the MCAL_P avoided with excellent performance even when a large number of people congregated and moved. In this experiment, there were 20 times when robots had to avoid various obstacles, and all 20 times succeeded in avoiding them. The robot avoided fast approaching people in various ways, such as making a sudden stop or changing direction.

Thus, it is expected that the robot will be able to perform excellent obstacle-avoidance driving even in collaborative environments such as actual factories and crowded environments in parks and airports.

5.3 Multi-robot driving simulation

In this work, MCAL, MCAL_P were trained in the form of multi-robots. Accordingly, a simulation experiment was conducted to confirm that each robot was well avoided and



(a) Map A: empty **(b)** Map B: 3 wide static

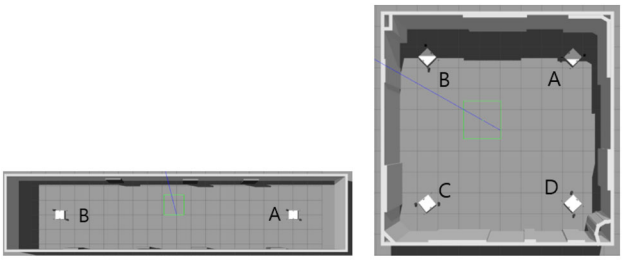
(c) Map C: 3 narrow static **(d)** Map D: 2 perpendicular

(e) Map E: 2 parallel **(f)** Map F: 2 diagonal

Fig. 15 Rotational velocity of the robot

Fig. 16 Map for real working environment

Fig. 17 Experiment for real working environment (https://youtu.be/xxzoh1XbAl0?t=91)



(a) 2 robots **(b)** 4 robots

Fig. 18 Virtual environment for multi robot test (https://youtu.be/xxzoh1XbAl0?t=148)

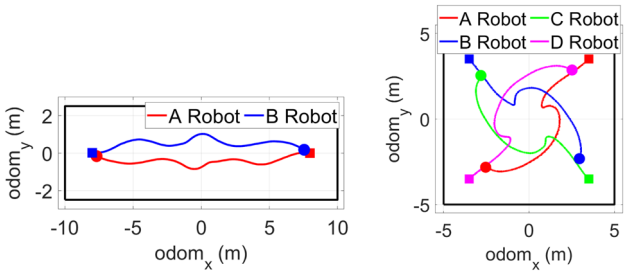
traveled to the target point even in an environment where a number of robots traveled together, such as a logistics center.

5.3.1 Experimental environment and methods

A 10 × 5 m environment in which two robots were run (Fig. 18a), and a 10 × 10 m environment in which four robots were run (Fig. 18b) were constructed for multi-robot navigation and driving. In Fig. 18a, robots A and B, where position of one was the target of the other, conducted 10 obstacle avoidance runs. Similarly, in Fig. 18b, an experiment was conducted in which robot A moved to the position of robot C as the target position, robot B went to the position of D as the target position, robot C drove to the position of robot A as the target position, and robot D moved to the position of robot B as the target position 10 times.

5.3.2 Experiment result

In both the experiments in which the two robots avoided each other and in which the four robots avoided one another, the robots succeeded in all the 10 attempts and showed excellent obstacle avoidance performance. Although the robots did not communicate with each other, they had the same policies. Therefore, they took the same actions to avoid, and unlike



(a) 2 robots **(b)** 4 robots

Fig. 19 Robots odometry for multi-robot test (square marker: start point, circle marker: end point)

people and objects defined as dynamic obstacles in previous experiments, each robot easily avoided each other. The paths for the same policy can be found in Fig. 19. It can be seen that the robots derived similar driving paths under similar conditions and showed a tendency to avoid by turning to the left of the others.

6 Analysis and discussion

When driving to avoid obstacles through MCAL, the problem was that the driving could not be continued for a long time if the driving direction was not found. To improve this, the proposed MCAL_P provides the robot with a favorable direction for driving through an optimal path. The MCAL_P was able to effectively avoid dynamic obstacles in various ways, such as acceleration, directional change, and stopping unlike the DWA and TEB, which avoided dynamic obstacles only through stop actions, thus confirming the best dynamic obstacle avoidance performance. This observation seems to be because the objective functions of DWA and TEB were calculated without considering the movement direction and speed of the obstacle, and thus, only favorable for static obstacle avoidance. On the other hand, the MCAL_P can grasp temporal movement of obstacles around the robot by setting the lidar data of the current step and the previous two steps as a state. The dynamic obstacle avoidance performance of MCAL_P is excellent because the obstacle avoidance policy was trained considering the moving direction and speed of the dynamic object. In addition, the MCAL_P also showed excellent avoidance performance in a complex environment where a large number of large people move, and showed smooth avoidance behavior even in multi-robot environments.

However, the MCAL_P has certain disadvantages also. It tends not to decelerate and accelerate except in the situations of avoiding obstacles, which can be a problem when carrying work. Despite reducing inefficient driving using the optimal path, the disadvantage of making inefficient driving, such as frequent deceleration and stop movements, remains when avoiding narrow-positioned static obstacles. This is attributed to the problem in training focused on dynamic obstacle avoidance, largely avoiding static obstacles considering the possibility of movement, and the problem caused by the robot's poorly controlled and out-of-target movements.

Another drawback is the problem of not being able to draw a straight path, which is believed to be due to reinforcement learning. During training, the robot drives directly and receives the evaluation of the drive as a reward, and learns the policy for obstacle avoidance and driving in a way that maximizes its rewards. Therefore, if driving along a straight line is not executed during training, the robot cannot determine whether a straight line is a good way of driving because it would not have any information on straight line driving. However, in the reinforcement learning of continuous action, the probability that the rotational velocity of 0 rad/s will be continuously derived for all steps is close to 0, so information on straight-line driving is not obtained.

7 Conclusion

The current study proposed a method in which a mobile robot is able to avoid both static and dynamic obstacles and can drive to the target point based on reinforcement learning. Assuming decentralized control without communication between agents, only the information on distance obtained through lidar was used to derive the avoidance policy for new and various obstacles. In this regard, for effective dynamic obstacle avoidance, the method of predicting and avoiding the movements of dynamic obstacles was adopted based on the information of the lidar distance of the present and the past two steps. Instead of separately predicting the motions of the obstacles, planning, and following a path, reinforcement learning was used to take into account the avoidance of various motions of an irregular dynamic environment from the observed state and output the action directly. To reduce the difference between the real driving environment and the training environment, we developed a training environment where dynamic characteristics were considered and implemented using soft actor critic as the reinforcement learning algorithm to learn policies. Furthermore, due to the high sample efficiency of SAC, the policy could be obtained in short time. As the first idea, the reinforcement learning-based obstacle avoidance driving method proposed in this study involves forward and rotational velocity, which are continuous actions for obstacle avoidance and driving to target position by inputting the lidar data over time, the speed of the robot, and the distance to the target position. However, it showed a problem that driving in a specific environment was impossible. In order to solve this problem, a path planner was integrated, and the input of the trained policy was modified with the lidar data, the speed of the robot, and the distance to

the look-ahead point. The improved method, while following the optimal path globally, performed various methods of avoidance through acceleration/deceleration, stopping, and direction change according to the movement of the dynamic obstacle and the distance between the robot and the obstacle locally.

Through various simulations and real-world experiments, it was confirmed that the obstacle avoidance performance was excellent in the actual work environment. The dynamic object avoidance performance of MCAL_P was much better than the DWA and TEB algorithms, and the time of operation to derive the algorithm's resulting values, linear and angular velocity was also shorter. In real-world environmental experiments, we confirm that robots using MCAL_P can also avoid objects that appear suddenly at fast response rates. We also found that in multi-agent environments, more effective and secure avoidance proceeds because robots using MCAL_P avoid each other with the same policy.

However, there are certain disadvantages also that need to be improved, e.g., it cannot drive in a straight line, draws inefficient paths while avoiding static obstacles, and does not perform acceleration/deceleration while driving other than obstacle avoidance.

Videos of the experiment are available at https://youtu.be/xxzoh1XbAl0.

Acknowledgements This research was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0008473, HRD Program for Industrial Innovation).

References

1. Abe Y, Matsuo Y (2001) Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In: IEEE international conference on intelligent robots and systems. https://doi.org/10.1109/iros.2001.977147
2. Martinez-Gomez L, Fraichard T (2009) Collision avoidance in dynamic environments: an ICS-based solution and its comparative evaluation. In: Proceedings—IEEE international conference on robotics and automation. https://doi.org/10.1109/ROBOT.2009.5152536
3. Tan Q, Fan T, Pan J, Manocha D (2019) DeepMNavigate: deep reinforced multi-robot navigation unifying local & global collision avoidance. arXiv:1910.09441
4. Xue X, Li Z, Zhang D, Yan Y (2019) A deep reinforcement learning method for mobile robot collision avoidance based on double DQN. In: IEEE international symposium on industrial electronics. https://doi.org/10.1109/ISIE.2019.8781522
5. Chen C, Liu Y, Kreiss S, Alahi A (2019) Crowd-robot interaction: crowd-aware robot navigation with attention-based deep reinforcement learning. In: Proceedings—IEEE international conference on robotics and automation. https://doi.org/10.1109/ICRA.2019.8794134, arXiv:1809.08835
6. Fox D, Burgard W, Thrun S (1997) The dynamic window approach to collision avoidance. IEEE Robotics Autom Mag 10(1109/100):580977
7. Rosmann C, Hoffmann F, Bertram T, (2015) Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control. In: European control conference. ECC 2015. https://doi.org/10.1109/ECC.2015.7331052
8. Fiorini P, Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. Int J Robotics Res. https://doi.org/10.1177/027836499801700706
9. Van Berg JD, Lin M, Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings—IEEE international conference on robotics and automation. https://doi.org/10.1109/ROBOT.2008.4543489
10. Snape J, Berg JVD, Guy SJ, Manocha D (2011) The hybrid reciprocal velocity obstacle. IEEE Trans Robotics. https://doi.org/10.1109/TRO.2011.2120810
11. Van Den Berg J, Guy SJ, Lin M, Manocha D (2011) Reciprocal n-body collision avoidance. In: Springer tracts in advanced robotics. https://doi.org/10.1007/978-3-642-19457-3_1
12. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. Nature. https://doi.org/10.1038/nature16961
13. Kahn G, Abbeel P, Levine S (2020) BADGR: An autonomous self-supervised learning-based navigation system. arXiv:2002.05700
14. Chen YF, Everett M, Liu M, How JP (2017) Socially aware motion planning with deep reinforcement learning. In: IEEE international conference on intelligent robots and systems, https://doi.org/10.1109/IROS.2017.8202312, arXiv:1703.08862
15. Long P, Fanl T, Liao X, Liu W, Zhang H, Pan J (2018) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: Proceedings—IEEE international conference on robotics and automation. https://doi.org/10.1109/ICRA.2018.8461113, arXiv:1709.10082
16. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: 35th international conference on machine learning, ICML. arXiv:1801.01290
17. Burgard W, Stachniss C, Bennewitz M, Arras K (2018) Introduction to mobile robotics—Bayes filter, particle filter and Monte Carlo localization (uni freiburg. edn). Lectures
18. Fan T, Long P, Liu W, Pan J (2020) Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int J Robotics Res 39(7):856–892. https://doi.org/10.1177/0278364920916531
19. Gerkey B, Vaughan R, Howard A (2003) The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of international conference on advanced robotics (ICAR 2003)
20. Koenig N, Howard A (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS). https://doi.org/10.1109/iros.2004.1389727
21. Botteghi M, Khaled M, Sirmaçek B, Poel M (2020) Entropy-based exploration for mobile robot navigation: a learning-based approach. In: Planning and robotics workshop, PlanRob
22. Feng S, Sebastian B, Ben-Tzvi P (2021) A collision avoidance method based on deep reinforcement learning. Robotics 10(2). https://doi.org/10.3390/robotics10020073, https://www.mdpi.com/2218-6581/10/2/73
23. Morales J, Martínez JL, Martínez MA (2009) Mandow A (2009) Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2d laser scanner. EURASIP J Adv Sig Process 1:935237. https://doi.org/10.1155/2009/935237
24. SM L (2006) Search for feasible plans. Planning algorithms. Cambridge University Press.

25. Marder-Eppstein E, Berger E, Foote T, Gerkey B, Konolige K (2010) The office marathon: Robust navigation in an indoor office environment. In: Proceedings—IEEE international conference on robotics and automation. https://doi.org/10.1109/ROBOT.2010.5509725

26. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: ICRA workshop on open source software