Deep Reinforcement Learning Based Autonomous Driving with Collision Free for Mobile Robots

Kiwon Yeom
Department of Human Intelligence and Robot Engineering, Sangmyung University, South Korea
Email: pragman@naver.com

Abstract—**The path planning of the mobile robots cannot be separated from the effectiveness of navigation and collision-free motion. In addition, dynamic path planning of unknown environment has always been a challenge for mobile robots since lack of information for the surrounding environment. This paper proposes a Deep Reinforcement Learning (DRL) based collision free path planning architecture for mobile robots. As navigating the environment, the mobile robots can figure out the unknown environment through DRL and the predicted values of control parameters from DRL are used to inputs for the mobile robots in the next time step. In addition, the architecture does not need any supervision. The experimental results of the architecture is compared to well-known approaches and shows that the architecture can be successfully applied to solve the complex navigation problem in the dynamic environments.**

Index Terms—**deep reinforcement learning, autonomous navigation, collision free, path planning, mobile robot.**

I. Introduction

One of the essential and important requisites for the mobile robots is secure navigation in a given environment. The course of the navigation from the starting position to the goal position is called path, and the path planning is to design an optimal path that the mobile robot can successfully travel without collision or failures. For the purpose of this, various path planning methods and control algorithms have been suggested [1], [2] and [3].

In general, the navigation performance of the mobile robot is related to the generated path from the path planner and mainly relies on the information of the surrounding environment which the mobile robot can achieve. However, the mobile robot usually cannot acquire the necessary information of the surrounding environment as well as complete their learning in complicate environments [4] and [5]. Therefore, it is required a new path planning architecture which can make a valid path using sparse and insufficient surrounding information so that the mobile robot can quickly adapt to the environment and autonomously drive to the destination with no collision.

Recently, machine learning and artificial intelligence based methods have been proposed [6], [7], [8] and [9]. In particular, Deep Neural Networks (DNN) has drawn attention to researchers and there are various applications related to mobile robot’s path planning [10], [11], [12], [13], [14].

In addition, to improve the navigation performance of mobile robots, several fusion algorithms such as DNN and PID(Proportional-Integral-Derivatives) were devised [15], [16]. Although the performance of these applications are not outstanding, the proposed methods had been successfully adopted to the mobile robot’s path planning.

Unlike the other supervised learning methods such as regression and back propagation, the DNN based method has obvious advantages in path planning and requires less prior information about the environment as well.

From this observation, this paper proposes Deep Reinforcement Learning based path planning architecture for the mobile robot. In this paper, Reinforcement Learning (RL) takes into account the problem of agents (software agent, autonomous robot itself, etc) learning how to achieve their goals or make decisions. Then, Deep Learning (DL) is used for the agents to decide what actions should be taken for optimizing the objective function such as cost, loss, or etc. In this paper, the proposed algorithm maps the current state from deep neural networks into the action of the deep reinforcement learning. Then, the state information is updated through the enhancement function for the next state.

This paper starts with introducing the kinematic bicycle model of the car-like mobile robot in Section II, and describes the DL architecture in Section III. This paper presents the how the DL maps into RL in Section IV, and shows the experimental results in Section V. Finally, this paper verifies the effectiveness of the proposed architecture by comparing with other path planning algorithms in Section IV and draws the conclusions.

II. Kinematic Bicycle Model for Car-Like Mobile Robots

In this paper, the path planning problem is considered for the car-like mobile robot with four wheels which the front wheels are for steering and the rear wheels are fixed on the chassis as shown in Fig. 1.

Manuscript received October 14, 2021; revised December 13, 2021.
Corresponding author: Kiwon Yeom, +82-41-550-5270

In this paper, a general kinematic bicycle model is taken into account for the mobile robot. This section simply recalls the fundamental formulations from [1] and more details should be referred in [6] and [7].

A four-wheeled vehicle, which consists of the fixed rear wheels and front steering wheels, can be analogous to the general bicycle since the real wheel of the bicycle is fixed to the body and the front wheel is used to handle the direction. The pose of the vehicle is described by the coordinate frame *B* that the origin is located on the center of the rear axle as shown in Fig. 1.

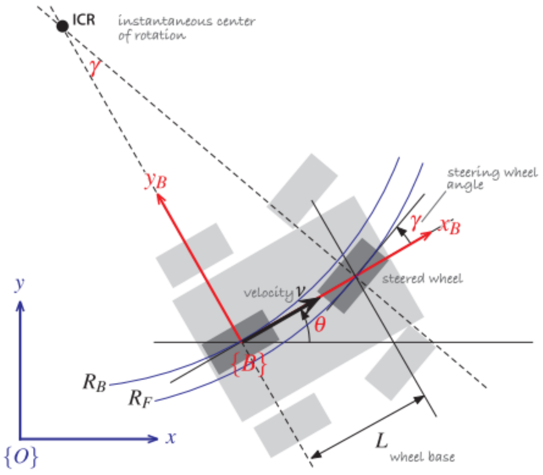

Figure 1. Bicycle model of a car-like mobile robot (figure curtesy of [12]).

The kinematic bicycle model can be called the front wheel steering model since the orientation of the vehicle is decided by the steering angle of the front wheels with respect to the direction. In contrast, the rear wheels are linked together through an axle which is fixed to the vehicle and rotated with no motion of rotation such as roll, pitch and yaw. In addition, for the directional control of the vehicle, the plane which the front wheels contact can rotate with respect to the vertical axis from the ground as the handle is turned.

Since the autonomous mobile robot can be usually described by the continuous-time nonlinear systems, in this paper the control mechanism for the mobile robot is approximated to the simplified control model for the general bicycle as shown in Fig. 2.

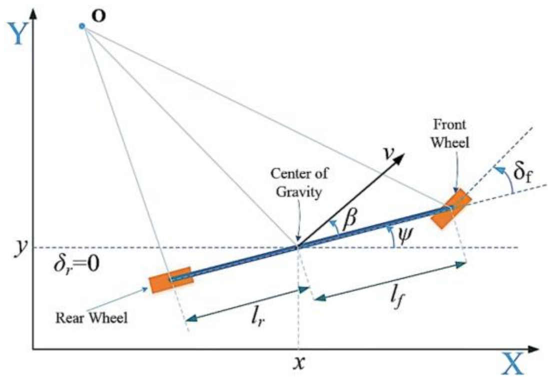

Figure 2. Simplified kinematic bicycle model for the mobile robot (figure curtesy of [13]).

In Fig. 1, the control parameters, which explains the status of the mobile robot, can be described by the generalized coordinates as follows

$$q = (x, y, \theta) \in C \tag{1}$$

where $C \subset R^2 \times S^1$. C represents the configuration space of the mobile robot and S^1 means unit circle with a set of angles $[0, 2\pi)$.

For generating rolling motion of the mobile robot, the robot must rotate the wheels about central shafts (or axles). Each center point of the axle lies along their left or right wheel axis, and is commonly located on the center of individual axle.

In Fig. 1, the dashed lines are represented along the direction which the wheels cannot move. In addition, the lines are intersected at a virtual point which is known as the ICR (Instantaneous Center of Rotation) that is the plane rotation center for the rotation of the mobile robot. Thus, the extension line of the reference point which is lies on the center of the mobile robot with orange color must be intersected at the ICR when the mobile robot makes a left or right turn.

The mobile robot has angular velocity and it is represented with respect to the linear velocity v as follows

$$\dot{\theta} = \frac{v}{R_B} \tag{2}$$

And by simple geometric operation, the turning radius is

$$R_B = L / \tan\gamma \tag{3}$$

where L is the length between the front and rear wheel bases. This means that the turning circle increases as the length is larger. Since the steering angle γ is mechanically restricted, the maximum value of the steering angle can be found as R_B is the minimum value.

The lateral dynamics of bicycle model is as follows

$$\dot{x} = v cos(\psi + \beta) \tag{3}$$

$$\dot{y} = v sin(\psi + \beta) \tag{4}$$

$$\dot{\psi} = \frac{v}{l_r} sin\beta \tag{5}$$

$$\dot{v} = \alpha \tag{6}$$

$$\beta = tan^{-1}\left(\frac{l_r}{l_f + l_r} tan\delta_f\right) \tag{7}$$

where x and y are the coordinates of the center of mass of the mobile robot which are represented in an inertial frame (X, Y). ψ is the heading angle of the mobile robot in an inertial frame and v is the speed of the center of mass of the mobile robot. l_f and l_r describe the distance from the center of the mass of the mobile robot to the front and rear axles, respectively.

β is the angle from the center of mass with respect to the longitudinal axis of the mobile robot. It is also the same as the angle of the current velocity v of the mobile robot.

α is the acceleration of the center of mass for the mobile robot. The direction of the acceleration is the same as the one of velocity.

In this lateral dynamics of bicycle model, the major control inputs are considered as the front and rear steering angles, namely, δ_f and δ_r. Since in this article the rear wheels are fixed to the chassis and assumed that there is no slip from rotation, the mobile robot has only one control parameter. In other words, the rear steering parameter should be zero (namely, $\delta_r = 0$).

It should be considered in terms of the inertia for more accurate dynamics of the mobile robot as shown in Fig. 3.

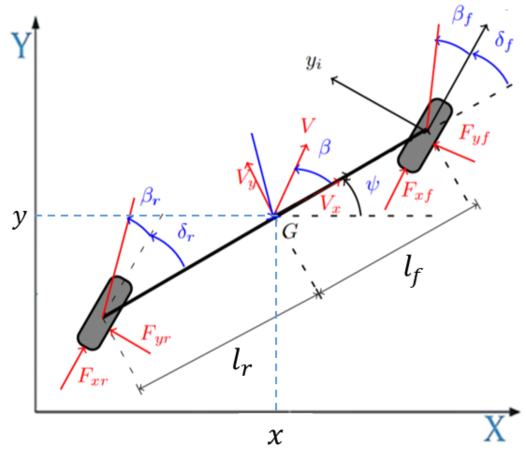

Figure 3. Velocity kinematic bicycle model with lateral forces.

From Eq. 3 to Eq. 5 with respect to the inertia of the dynamic bicycle model can be defined in the same manner as those in the velocity kinematic bicycle model (refer [17], for more specific descriptions).

The differential equations for the mobile robot with inertia are as follows

$$\ddot{\psi} = \frac{1}{J_z}(F_{yf} l_f - F_{yr} l_r) \tag{8}$$

$$\dot{v_x} = \frac{1}{m}(F_{xf} + F_{xr} + \dot{\psi} v_y) \tag{9}$$

$$\dot{v_y} = \frac{1}{m}(F_{yf} + F_{yr} - \dot{\psi} v_x) \tag{10}$$

where $\ddot{\psi}$ describes the ratio of the yaw rotation. m and J_z represent the mass and yaw inertia of the mobile robot, respectively. $F_{y,f}$ and $F_{y,r}$ describe the lateral forces of tires at the front and rear wheels, respectively. In addition, as mentioned before, the corresponding coordinate frame is aligned with the plane of the paired wheels.

For the linear tire model, F_{yi} is defined as follows

$$F_{yi} = -K_{yi}\beta_i \tag{11}$$

where $i \in \{f, r\}$, α_i is the tire slip angle and $C\alpha_i$ is the tire cornering stiffness. However, as mentioned in Section II, the tire parameters are considered in this article.

The tire slip angles are estimated and assuming small angles and they can be approximated as follows

$$\beta_f = \frac{v_y + l_f \dot{\psi}}{v_x} - \delta_f \tag{12}$$

$$\beta_r = \frac{v_y + l_r \dot{\psi}}{v_x} - \delta_r \tag{13}$$

III. Design of Artificial Neurla Network

The proposed DRL based controller consists of four elements such as sensing data, sideslip angles, vehicle controller, and DRL processor as shown in Fig. 4.

DRL processor predicts parameters for vehicle control such as velocity (v) and steering angle (θ), and the generated values are used to handle the mobile robot by the vehicle controller.

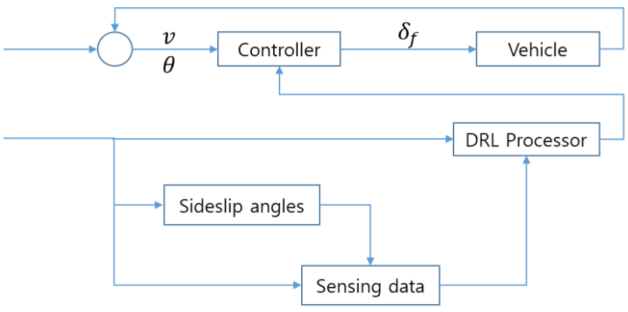

Figure 4. Block diagram of the DRL based mobile robot navigation.

The path planner generates the course of the navigation from the start position to the goal position.

In this paper, ANN comprises multiple layers with multiple neurons, which are divided into an input layer, a hidden layer, and an output layer as shown in Fig. 5.

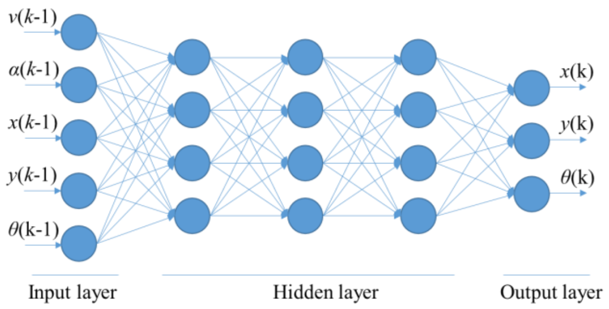

Figure 5. Demonstration of neural networks with four hidden layer

The input vector is should be:

$$x = [x_1, x_2, \dots, x_j, \dots, x_m] \quad j = 1, 2, 3, \dots m \tag{14}$$

And the output vector should be:

$$y = [y_1, y_2, \dots, y_k, \dots, y_n] \quad k = 1, 2, 3, \dots n \tag{15}$$

The neuron input of the hidden layer is as follows

$$l_j^{[i]} = f\left(net_j^{[i]}\right) \tag{16}$$

$$net_j^{[i]} = \sum_{i=1}^{j-1}(w_{ij}^{[i]} \cdot l_j^{[i-1]} + b_j^{[i]}) \tag{17}$$

$$i = 1,2,\dots,p, \quad j = 1,2,\dots,q$$

where i is the number of neurons in layer ith and j is the number of layers, and $w_{ij}^{[i]}$ is weights for the connecting nodes in the $(j-1)$th layer with the nodes of jth layer, and b is the bias for all nodes in the jth layer.

In this model, the transfer function of the input layer to the output layer uses the hyperbolic tangent transfer function (TANSIG), related to a bipolar sigmoid which has an output in the range of -1 to +1. The output layer uses linear transfer function (Purelin). For the learning, this paper uses Traingdx which is a network training function that updates weight and bias values according to gradient descent momentum and an adaptive learning rate. The performance is evaluated using Mean Squared Error (MSE), where the number of epoch is set to 1000 times and the accuracy is set to 0.0001.

In this paper, a deep neural network consists of a fully connected and multilayered that all the nodes (or neurons) in one layer are connected to the neurons in the next layer as shown in Fig. 4. Since fully connected networks is structure agnostic, in this paper, there are no special assumptions needed to be made about the input.

IV. Model of Reinforcement Learning

In this section, the Reinforcement Learning (RL) is introduced for the proposed DRL based controller. Further readings of the reinforcement learning should be referred in [15]. Like the general RL, the proposed DRL has the same RL structure as shown in Fig. 6.

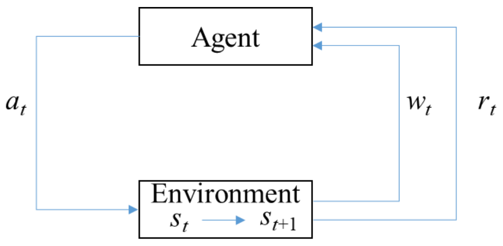

Figure 6. The schematic diagram of the reinforcement learning model.

Agent, which means the mobile robot in this paper, acquires the knowledge or information from a sequence of actions (a_t) such as exploration of surrounding environment. Every time the agent executes an action, the agent get feedback which is the evaluation of the action.

The evaluation is the reward for the agent and it is used as an indicator which describes the result with respect to the action. Therefore, the agent continuously tries to find the most impact action so as to get maximum reward. This process called 'trial and error' and in turn the optimal sequence of actions which the cost (or loss) function becomes the minimum value can be achieved. The other words, the agent can get the highest reward value [18].

Since the reward value indicates the quality of the action, the larger the value is, the better the performed action is. Otherwise it will be degenerated or has a poor impact.

Therefore, the interaction process with the surrounding environment to acquire good rewards by the agent can be regarded as a Markov Decision Processing (MDP) [18].

In this paper, the main objective is for the mobile robot to navigate the unknown environment with collision free. Therefore, the policy of action can be avoiding the obstacles in the surrounding environment.

At each timestamp t, system parameters are given: sensing data s_t, a local goal position g_t at the 2D cartesian coordinate system, the linear velocity v_t and angular velocity ω_t of the mobile robot. The policy which the mobile robot can select to avoid obstacles provides an action command as follows

$$\pi = (a_t, s_t) \in \Pi \tag{18}$$

where Π represents all sets of spatial states. In the policy set, it is required to obtain the strategy π^* called the optimal policy. As long as the random variable set $\{\Pi_1, \Pi_2, \Pi_3, \dots, \Pi_t\}$ satisfies the following equation, the set will have Markov attributes.

$$\begin{aligned} &P_r(\Pi_{t+1} = s' \mid \Pi_t = s_t,\ \Pi_{t-1} = s_{t-1}, \dots,\ \Pi_1 = s_1) \\ &\quad = P_r(\Pi_{t+1} = s' \mid \Pi_t = s_t) \end{aligned} \tag{19}$$

Once the state s is determined, the actions before the state are not correlated to the actions after the state. In addition, the actions are independent of each other.

MDP is defined as the tuple (Π, O, A, T, R, γ) as follows

$$\begin{aligned} &\lambda(s'|s,o,a,t,r,\gamma)) \\ &= P_r\ (\Pi_{t+1} = s' \mid \Pi_t = s,\ O_t = o, A_t = a, \\ &\qquad T_t = t, R_t = r, \gamma) \end{aligned} \tag{20}$$

Once the state s is determined, the actions before the state are not correlated to the actions after the state. In addition, the actions are independent of each other.

MDP is defined as the tuple (Π, O, A, T, R, γ) as follows

$$\begin{aligned} &\lambda(s'|s,o,a,t,r,\gamma)) \\ &= P_r\ (\Pi_{t+1} = s' \mid \Pi_t = s,\ O_t = o, A_t = a, \\ &\qquad T_t = t, R_t = r, \gamma) \end{aligned} \tag{21}$$

where the state set S, the action set A, the reward function R, the state transition function T, and the observation function $O(o|s',a)$. The agent cannot determine the state set S but has to rely on observation function $O(o|s',a)$. After each state transition, the robot receives an immediate reward $R(o|s,a)$ [19].

The state of the environment changes based on the robot's actions $a \in A$, which are in this paper the velocity and steering angle commands(v,θ), and the probability of transition $T(s' |s,a)$. The output of sequential decisions can be considered as a trajectory l from the start (previous) position p_0 to the goal (current) position p_g, where t_g is the travel time to the current position. The objective of the control is to minimize the actual traveling time without any collision with obstacles. Therefore, the objective function can be written as follows

$$\arg\min_{\pi_\Theta} \Psi\, [t_g | a_t = \pi_\theta(o_t), \quad p_t = p_{t-1} + a_t \cdot \Delta t] \tag{22}$$

where Θ is model parameters.

In the MDP problem, a policy $\pi(a|s)$ specifies the probability of mapping state s to action a and the stochastic policy $\pi(a_t|o_t)$ can be evaluated by V^π which is the state value function as follows

$$V^*(s_t, a_t) = \max_\pi \Psi V^\pi(s, a) \tag{23}$$

The reward function in this paper considers task completion (R_g), the collision(R_c), and the progress (R_p) towards the goal position:

$$R(s,a) = R_g(s,a) + R_{fin}(s,a) + R_c(s,a) + R_p(s,a) \tag{24}$$

If the mobile robot reaches its goal, it is rewarded by $R_g(s,a)$ as follows

$$R_g(s,a) = \begin{cases} \zeta & \text{if the goal is achieved} \\ -\zeta & \text{if a collison accures} \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

$R_g(s,a)$ becomes a large positive value if the mobile robot arrives at a location within a 0.5m radius of the final goal.

$$R_{pt}(s,a) = \begin{cases} -\varsigma & \text{if the distance} < \text{steering radius} \\ 0 & \text{otherwise} \end{cases}$$

$R_{pt}(s,a)$ becomes a large negative value if the distance between the robot and the nearest obstacle is less than steering radius since the collision can be occurred. $R_c(s,a)$ is a fixed negative reward for the mobile robot to finish an episode as fast as possible. The fourth reward component is used to speed up the training process as follows

$$R_p(s,a) = \eta \cdot D(s,a) \tag{26}$$

where $D(s,a)$ is the cost function according to the distance between the mobile robot and the subgoal position at time t and η is a scaling factor in the range of 0 to 1.

V. Experimental Environments

In order to verify the performance of the proposed method, this paper constructed a 2D grid map model and the simulation environment in Python with Tensorflow libray.

The simulation program was developed using Python programming language. The computer configuration: 4-core Intel i5 7400 CPU@3.20 GHz, 16G memory, Windows 10 operating system.

As shown in Fig. 7, the chassis of the mobile robot applied a car-like design with 4 wheels. Since the axle of the rear wheels is fixed to the body, the rear wheels can only rotate back and forth without pitch, roll and yaw.

The directional control of the front wheels is based on a simplified Ackerman steering geometry. In addition, the front wheels are forced to maintain parallel alignment at all times by an axle. The axle pivots about its center on a vertical steering rod, which is connected to a steering servo-motor through a rack and pinion system. The servo-motor is controlled by a microcontroller (Raspberry Pi 3 with Quad Core 1.2GHz) using a pulse-width modulation (PWM) signal and is capable of steering angles of $-45 \lessapprox \delta \leq 45°$ about front wheels.

VI. Results and Discussions

To train the neural network and generate a policy to follow a global path without collisions, we sampled the start and goal positions randomly across the free space with short distance at the beginning stage and later increased the distance including several obstacles. After training, we performed experiments to evaluate the policy learned in terms of the success rate, which means that the mobile robot reached the goal position without any collisions and completion time. The performance was compared with dynamic window approach in ROS.

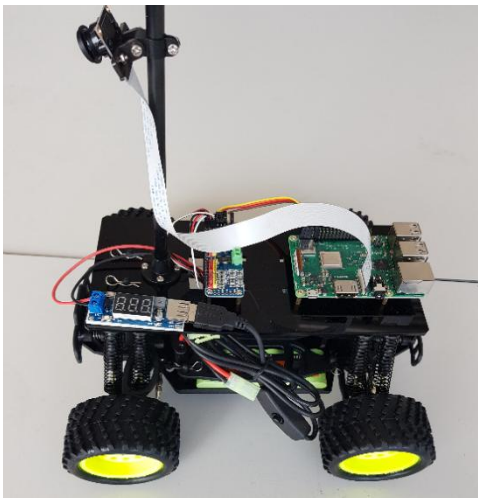

Figure 7. The autonomous car-like mobile robot platform.

A path from (0,0) to (200, 200) is used to validate the proposed DRL algorithm. In this simulation, Dynamic Window Approach (DWA), which is an online collision avoidance strategy for mobile robots, is also used for the purpose of the performance comparison [19].

As shown in Fig. 8, both DWA and DRL algorithms reached the destination without any collisions.

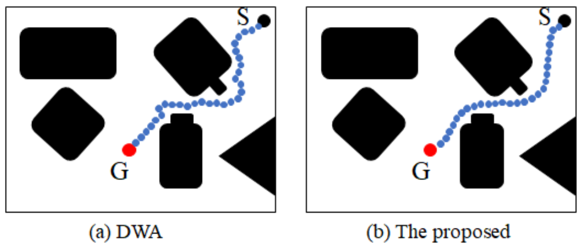

Figure 8. Experimental results of different path planning algorithms.

In addition, all methods effectively avoided obstacles under the different and more complex conditions as shown in Fig. 9.

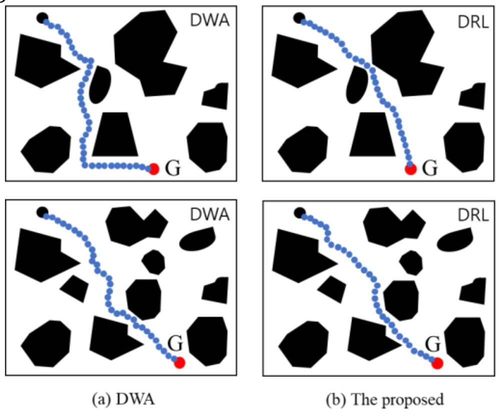

Figure 9. Path tracking in complex environments.

Comparing DWA, it was found that the proposed method efficiently reduced the number of path steps about 21.3% on average after 10 tests.

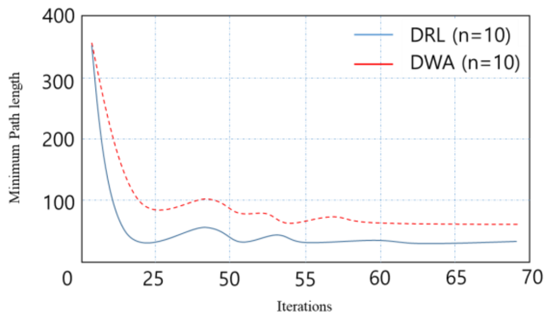

Figure 10. Path convergence curve.

As shown in Fig. 10, the proposed algorithm has improvement over DWA in terms of success rate to reach the goal position.

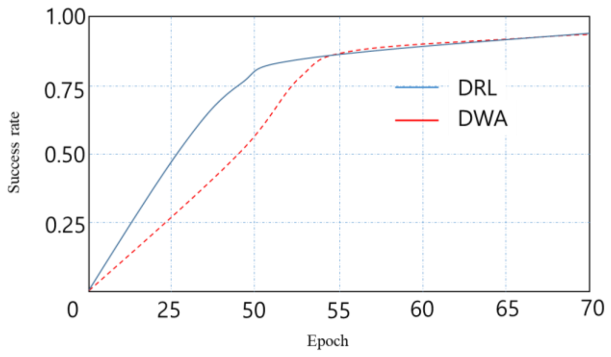

Figure 11. Success rate.

For each environment in Fig. 11, 400 runs were performed with DWA and the proposed algorithm. in each testing environments. The success rates for the scenario 1 are lower than other scenarios. The reason we believe is that enough information was not retrieved due to short learning epoch. However, after tuning the epoch the DWA and the proposed algorithm successfully reached the goal position over 85% for the scenario 2. As shown in Fig. 12, The success rate of the scenario 3 is conspicuously lower than that of other scenarios due to the space complexity. The proposed DRL based algorithm has better performance for all scenarios with respect to average execution time and path length.

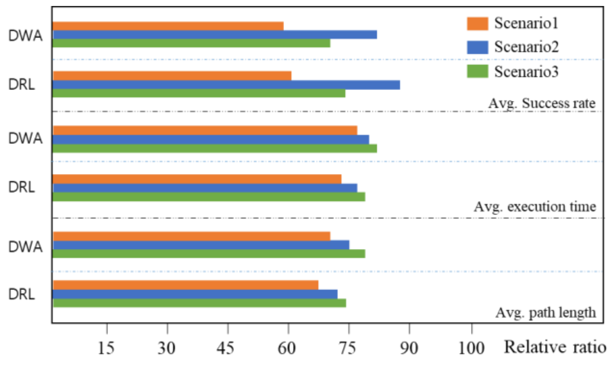

Figure 12. Performance evaluation with average of 400 runs.

The path length of DWA has 10% longer on average than the proposed algorithm. However, the proposed algorithm has more time to complete navigation tasks, which shows the mobile robot controlled much cautiously about unknown environment as shown in Fig. 12.

VII. Conclusions

This article introduced a Deep Reinforcement Learning (DRL) based path planning architecture which was used for trajectory following of a car-like mobile robot. This paper described the bicycle kinematic model for the mobile robot, DRL controller for tracking a car-like mobile robot. The framework that combines the DRL processor and vehicle controller with the deep neural network was described. In order to demonstrate the feasibility of the proposed method in path planning without collisions, the controller was applied to the different environment to reach the goal position.

Experimental results showed that the proposed DRL controller has good performance and make path planning results more practical. Generally, the DRL controller reduced the tracking errors and path length about 20% compared to the DWA controller in this article. For the future work, the study of effectiveness of the DRL controller should be performed in more complex environments with other control strategies.

Conflict of Interest

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

The author had conducted the research and had approved the final version.

REFERENCES

[1] N. H. Amer, H. Zamzuri, K. Hudha, and Z. Kadir. "Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges," *Journal of Advanced Robot and System*, vol. 86, pp. 225-254, 2017.

[2] C. Shen, Y. Shi, and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using Lyapunov-based model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 65, no 7, pp. 5796-5805, 2017.

[3] X. Yu, W. He, H. Li, and J. Sun, "Adaptive fuzzy full-state and OutputFeedback control for uncertain robots with output constraint," *IEEE Transaction on Systems, Man, and Cybernetics*, 2020.

[4] A. V. Duka, "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," *Procedia Technol.*, vol. 12, no. 1, pp. 20-27, Jan. 2014.

[5] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation," in *Proc. CoRL*, 2017.

[6] K. W, Yeom, "Kinematic and dynamic controller design for autonomous driving of car-like mobile robot," *International Journal of Mechanical Engineering and Robotics Research*, 2018.

[7] C. Urmson, J. Anhalt, D. Bagnell, C Baker, and R. Bittner, et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[8] N. H. Amer, H. Zamzuri, K. Hudha, and Z. Kadir, "Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 225-254, 2017.

[9] B. Karlik, A. V. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *Int. J. Artif. Intell. Expert Syst.*, vol. 1, no. 4, pp. 111-122. 2011.

[10] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," In CoRL, 2019.

[11] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," arXiv preprint arXiv:1805.12114, 2018.

[12] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680, 2019.

[13] Z. Lin, Y. J. Zhang, and Y. F. Li, "Path planning for indoor mobile robot based on deep learning," Optik 219 (2020): 165096, 2020.

[14] S. M. Sombolestan, A. Rasooli, and S. Khodaygan, "Optimal path-planning for mobile robots to find a hidden target in an unknown environment based on machine learning," *J Ambient Intell Human Comput*, vol. 10, pp. 1841-1850, 2019.

[15] Zafar, M. Nayab, and J. C. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia Computer Science*, vol. 133, pp. 141-152, 2018.

[16] W. Farag, "Complex trajectory tracking using PID control for autonomous driving," *Int. J. ITS Res.* vol. 18, pp. 356–366, 2020.

[17] W. Liu, C. Chen, G. Knoll, "Gaussian process based model predictive control for overtaking in autonomous driving," *Frontiers in Neurorobotics*, vol. 15, 2021.

[18] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. "Qt-opt: Scalable deep reinforcement learning for vision based robotic manipulation," In CoRL, 2018.

[19] A. Özdemir, V. Sezer, "A hybrid obstacle avoidance method: follow the gap with dynamic window approach," in *Proc. 2017 First IEEE International Conference on Robotic Computing (IRC)*, 2017.

Kiwon Yeom is a professor in the department of human intelligence and robot engineering, Sangmyung University, South Korea**.** He is interested in intelligent robot control and swarm robot control. He is now pursing the intelligent control algorithm for the identical and multiple robots to be used in the disaster situation.