International Journal of Industrial and Manufacturing Systems Engineering
2025, Vol. 10, No. 2, pp. 36-43
https://doi.org/10.11648/j.ijimse.20251002.12

Research Article

Pose Control of Omnidirectional Mobile Robot Using Improved Deep Reinforcement Learning

Kim Kwang Jin, Yun Ji Yon, Ro Kang Song*, Jo Kwang Bin, Pak Mu Rim

Faculty of Mechanical Science and Technology, Kim Chaek University of Technology, Pyongyang, DPR Korea

Abstract

Nowadays, mobile robots are being widely applied in various fields such as indoor carrying and check of products and outdoor exploration. One of the most important problems arising in development of mobile robots is to resolve path planning problem. With active studies of implementation of path planning, lots of algorithms have been developed and especially, the dramatic advance in artificial intelligence (AI) led to advent of algorithms using reinforcement learning (RL). Deep reinforcement learning (DRL) has been developed and it uses neural network to approximate parameters of RL algorithm. DDPG is one of deep reinforcement learning (RL) algorithms and is widely used to solve lots of practical issues as it doesn't need full information of the environment. In other words, path planning with DRL has advantages of possibility for unknown environments in which partial or full information is not given and of direct controllability of the robot. Generally, path planning Up to now, path planning using DRL has considered only position control problem with no consideration of its orientation angle (as the author knows). In this paper, a pose control method using DRL for 3-wheeled omnidirectional mobile robot is proposed. And a method to reduce position error is mentioned. Simulation results show that the proposed method can efficiently solve the control problem of omnidirectional robots.

Keywords

3-wheeled Omnidirectional Mobile Robot, Deep Reinforcement Learning (DRL), DDPG Algorithm, Path Planning, Pose Control

1. Introduction

Recently, mobile robots are being widely used in industry, agriculture, public service and so on. With rapid robot technology and AI, more intelligent autonomous robots have been developed and study to modernize robots to successfully accomplish missions in more complex environments is being intensified.

Robot path planning is really important field in navigation of intelligent autonomous mobile robot. There are graph decomposition method, cell decomposition method [1], univector field method (UVFM), artificial potential field method (APFM) [2, 3], genetic algorithm [4], A* algorithm [5] and the rest in classical path planning method for robots. In addition, some improved methods on classical ones have been proposed [6-8].

There have been many tries to combine classical methods with DRL to reflect not only pure geometric path, but also kinetic model of the robot and as a result, remarkable advance has been achieved [9-11].

*Corresponding author: Skiprotich@mut.ac.ke (Sharon Kiprotich)

Received: 2 September 2025; **Accepted:** 16 September 2025; **Published:** 9 October 2025

In paper [12, 13], they proposed a robot path planning method using Deep Q-Network (DQN) and compared with classical ones.

In paper [14], they proposed a way to increase convergence speed of DQN by means of improving grid map. Another method of using previous experience and knowledge to improve convergence speed of DQN was proposed in paper [15]. These methods guarantee optimum and convergence of path generation. However, as they use discrete action space of robot, they have some inevitable problems in combining kinetic model of robot with path planning.

In paper [16], to overcome the front problem, they proposed a method to combine DDPG algorithm with univector filed method so that action space of the omnidirectional robot can be considered as being continuous. Thus, it has position state of the robot as input, outputs orientation angle as its output and establish simultaneous control of the robot. However, as it considered only robot's position without its orientation angle, so it can't make use of its advantages: possibility of moving on optimal path and free change of orientation. It also can't reflect kinetic characteristics of the control object. It also can't reduce the position error less than a constant threshold.

To settle this problem, new method for simultaneous control of robot has been proposed in this paper.

First, as state vector of DRL algorithm, position of the omnidirectional mobile robot is composed with its orientation angle. Next, kinetic model of the robot was reflected in learning environment and the real-time angular velocities of wheels are determined.

This paper is divided into following sections.

In section 2, analysis of previous literature on kinetic model of 3-wheeled omnidirectional mobile robot and path planning by reinforcement learning are given.

In section 3, a new method of control position and orientation of the robot using DRL and of reduction of pose error of the robot is proposed.

In section 4, simulation of proposed method is performed and the result is analyzed.

Section 5 gives conclusion.

2. Previous Literature

2.1. Kinetic Model of 3-Wheeled Omnidirectional Mobile Robot

Since omnidirectional mobile robots have several advantages such as high mobility, free controllability of its orientation on moving path, they are widely used in various fields.

As can be seen in figure 1, in a 3-wheeled omnidirectional mobile robot, the wheels are set at $120°$ spacing and each wheel is an omnidirectional wheel.

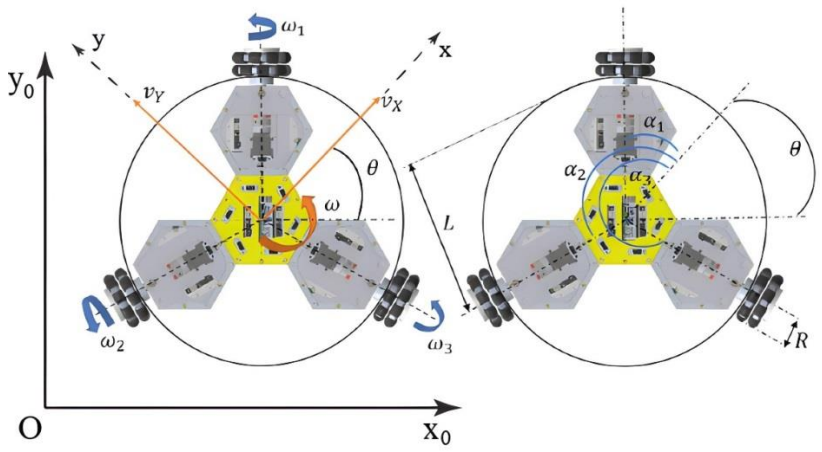

Figure 1. *3-wheeled Omnidirectional Mobile Robot.*

The forward kinematic model of a three-wheeled omnidirectional mobile robot is expressed in matrix form as follows:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \frac{R}{3}\begin{bmatrix} \sin\alpha_1 & \sin\alpha_2 & \sin\alpha_3 \\ -\cos\alpha_1 & -\cos\alpha_2 & -\cos\alpha_3 \\ -1/L & -1/L & -1/L \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = M\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (1)$$

Hence, the inverse kinematics model of the robot can be expressed as

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = M^{-1}\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (2)$$

where, v_x, v_y are linear velocity to x- and y-axis, respectively, ω is angular velocity of the robot, $\omega_1, \omega_2, \omega_3$ are angular velocity of each wheel, L is radius of the robot body, R is

radius of the wheel, θ is orientation of the robot in absolute coordinate system and α_1, α_2, α_3 are angle of placement of each wheel.

From known angular velocities of each wheel from Eq. (1), we can calculate the x- and y-axis linear velocities and angular velocity of the robot body.

2.2. DDPG Algorithm

The DDPG algorithm has the advantage of being able to handle continuous behavior space, unlike other DRL algorithms. The DDPG algorithm is based on the action-critic method and learns two neural networks, actor network and critic network. In the actor network, according to the state of the agent, we learn the action $a=\mu_\varphi(s)$ to take in that state, and the critic network computes the value $Q(s,a)$ according to the action taken in that state. Repeating this process several times, we store the transition information in the experience repository. Then, k data are randomly extracted from the experience repository to train the network. The block diagram of the DDPG algorithm is shown in Figure 2.

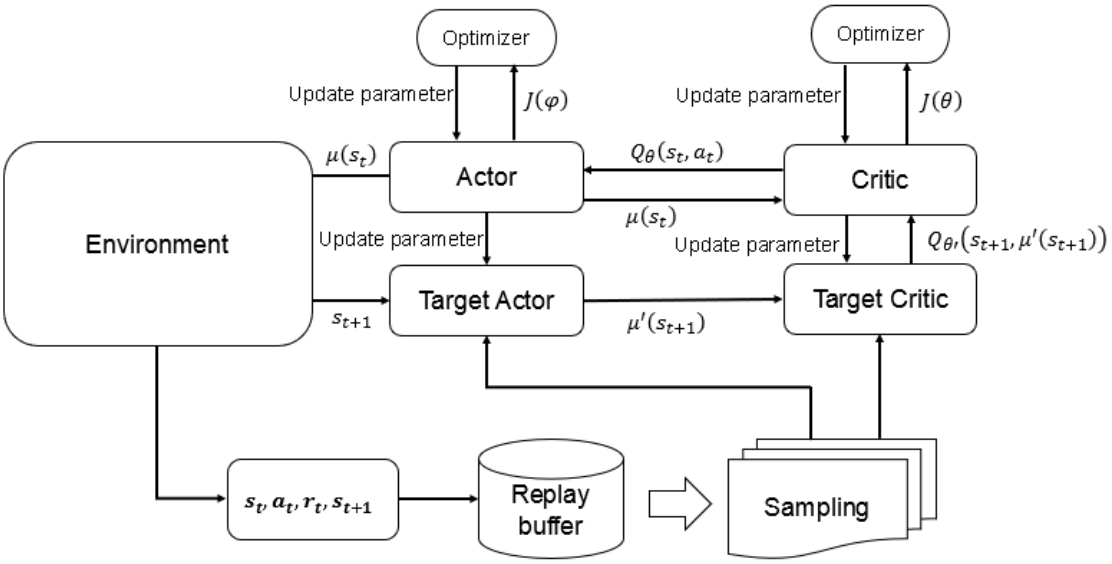

Figure 2. *Configuration of DDPG Algorithm.*

Summarizing the DDPG algorithm,

Initialize the experience repository D.

Randomly initialize the weight parameters of actor network φ, critic network θ, target actor network φ' and target critic network θ'.

Repeat Step 4 by N episodes.

In each step of every episode:

Based on policy $\mu_\phi(s)$, choose action a and add noise to search for new behavior.

$$a=\mu_\phi(s)+N$$

Perform the action a and move to the next state s'. Then, receive a reward r and save transition information in experience store D.

Randomly select K transition information in D.

Get target value of critic network.

$$y_i=r_i+\gamma Q_{\theta'}\left(s_i',\ \mu_{\phi'}(s_i')\right)$$

Calculate loss of critic network.

$$J(\theta)=\frac{1}{K}\sum_i\left(y_i-Q_\theta(s_i,\ a_i)\right)^2$$

Calculate gradient of loss function $\nabla_\theta J(\theta)$ and update critic network using gradient descent method.

$$\theta=\theta-\alpha\nabla_\theta J(\theta)$$

Calculate gradient of loss function $\nabla_\varphi J(\varphi)$ and update actor network using gradient ascent method.

$$\varphi=\varphi+\alpha\nabla_\varphi J(\varphi)$$

Update parameters of target critic and target actor networks.

$$\theta' = \tau\theta + (1-\tau)\theta'$$
$$\varphi' = \tau\varphi + (1-\tau)\varphi'$$

2.3. Path Planning Algorithm Using DDPG Combined with UVFM

In paper [8], a new method to combine DDPG algorithm and UVFM was proposed, so that they make action space continuous and simultaneously move the robot toward the direction which is outputted from the actor network of DDPG algorithm. Here, current position is given as input of the actor network of DDPG algorithm and orientation angle to which robot should move towards is given as output.

$$input = \{x, y\} \quad output = \{\phi\} \tag{3}$$

State update of the robot is performed by moving constant distance towards the output of the actor network.

$$\begin{aligned} x_{k+1} &= x_k + d \cdot \cos\phi \\ y_{k+1} &= y_k + d \cdot \sin\phi \end{aligned} \tag{4}$$

Here, x_{k+1} and y_{k+1} is position coordinates of the robot at $k+1$ step, ϕ is output of actor network of DDPG: orientation angle of the robot and d is length of movement of robot in one step (during sampling time).

The disadvantage of this approach is that it cannot take advantage of omnidirectional mobile robots that can freely change the orientation angle of the robot on the moving path because only the robot position is considered and orientation angle is ignored. Also, because this method considers orientation angle as its action space, it causes serious problems such as the jump of the angular velocity of wheels in the real-time control of the robot.

3. Proposed Method

3.1. Configuration of Environment Using DDPG

In our case, the agent for DDPG algorithm is 3-wheeled omnidirectional mobile robot.

In proposed method, to take advantage of the omnidirectional mobile robot, not only the position coordinates (x, y), but also the orientation angle θ of the robot is considered as its state like Eq. (5).

$$s = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5}$$

Next, the action space of the robot (output of actor network) is described as a vector which consists of angular velocities of each wheel of the robot. In this case, the angular velocities are the continuous value within a specified range.

Thus, we mapped a vector consisting of the angular velocities of the wheels to each state of the environment (map).

$$a = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{6}$$

After the angular velocities of the wheels are outputted from the actor network of the DDPG algorithm, x-axis velocity v_x, y-axis velocity v_y and the angular velocity of the robot body are calculated with its kinetic model and state of the robot can be updated by Eq. (7) from calculation of displacement and rotation angle during dt.

$$\begin{cases} x_{k+1} = x_k + v_x \cdot dt \cdot \cos(\theta_k) + v_y \cdot dt \cdot \cos(\theta_k + \pi/2) \\ y_{k+1} = y_k + v_x \cdot dt \cdot \sin(\theta_k) + v_y \cdot dt \cdot \sin(\theta_k + \pi/2) \\ \theta_{k+1} = \theta_k + \omega \cdot dt \end{cases} \tag{7}$$

Then, the DDPG algorithm should be trained to output angular velocities of each wheel of the robot from given state (position, orientation angle).

3.2. Improvement of DDPG Algorithm

The rewards that the robot receives at each stage are set to reflect the state transition such as collision with obstacles, map departure and target arrival.

Actions that make robot collide with obstacles or leave the map are completely unnecessary for the robot's mission, so very large negative reward is given to these actions.

Since we consider the motion of the robot in a continuous space, it is impossible to reach the target point exactly. Hence in [8], a certain region around the target point is defined as the target region, if the robot enters this region, it is estimated that the robot reached to the target and the episode is terminated. This method has the disadvantage that the robot position error cannot overcome certain threshold: radius of this region (d_0).

A small target region radius may lead to a long convergence time or even no convergence of the algorithm, whereas a large target region radius may result in a big position error instead of a fast convergence speed.

To overcome this drawback, we did not end the episode immediately when the robot enters the target region, but rather, the closer to the target, the more positive reward was received and episode was terminated at the moment that the distance to the target becomes larger again. In this way the final position error of the robot could be much reduced.

To ensure that the robot finally get certain target orientation angle, a constant negative reward proportional to the error

with the current orientation angle is given.

There may be many undesirable orientation changes during robot movement, which greatly affect the robot's motion time, control, and optimality of the path. Hence, a constant negative compensation was added to reduce the variation of orientation of the robot.

Other transitions receive negative reward proportional to the distance to the target point so that the robot could successfully reach to the target point.

Overall, the reward that the robot receives is expressed as Eq. (8).

$$R=\begin{cases} R_1 & \text{Collide with obstacles} \\ R_2 & \text{Depart from the map} \\ R_3 & \text{Enter in target region} \\ R_4 & \text{Reward proportional to orientation error} \\ R_5 & \text{Reward proportional to variation of orientation} \\ R_6 & \text{Otherwise} \end{cases} \tag{8}$$

Here, R_3, R_4, R_5, R_6 are calculated with Eq. (9).

$$\begin{aligned} R_3 &= \begin{cases} \min\left\{k_3 \cdot \frac{1}{d_i},\ 1000\right\}, & d \le d_0 \\ 0, & d > d_0 \end{cases} \\ R_4 &= k_4 \cdot \left|\theta_t - \theta_i\right| \\ R_5 &= k_5 \cdot \left|\theta_i - \theta_{i-1}\right| \\ R_6 &= k_6 \cdot d_i \end{aligned} \tag{9}$$

Where,

d_i - Distance between robot and target at step i

d_0 - Critical distance to estimate arrival to target region

θ_i - Orientation of the robot at step i

θ_t - Target orientation angle

k_3, k_4, k_5, k_6 - Coefficients

In reinforcement learning, networks are trained to maximize reward, so that the robot can reach the target state (position, orientation angle) faster in shorter time without any collision with obstacles and departure from the map and with less variation of orientation angle.

4. Simulation Result and Analysis

The training environment of reinforcement learning was built on simulation application- *"CoppeliaSim"*, and the DDPG algorithm was implemented using the deep learning library *"pytorch"*.

The simulations were performed on 8GB RAM, Intel Core-i7-1165G7 processor and NVIDIA RTX 3050 GPU platform.

4.1. Configuration of the Environment

On the whole map size of 10×10, the initial state of the robot is $(2, 2, 0°)$, the target state is $(8, 7.2, 90°)$ and 4 obstacles are placed randomly. The diameters of the obstacles are in range of $(1.5,\ 2.5)$.

4.2. Determination of Simulation Parameter

Structure of the actor and critic network are shown in following table.

Table 1. *Parameters of the Actor Network.*

Layer	Type	Number of parameters	Activation
Input		3	
Layer1	Dense	32	tanh
Layer2	Dense	256	tanh
Layer3	Dense	32	tanh
Output	Dense	6	$5\pi\cdot\tanh$

Because the output of the actor network is the angular velocity of the wheel, it must be in range of $[-5\pi,\ +5\pi]$, so activation function of the output layer is set as $5\pi\cdot\tanh$.

Table 2. *Parameters of the Critic Network.*

Layer	Type	Number of parameters	Activation
Input		6	
Layer1	Dense	32	tanh
Layer2	Dense	256	tanh
Layer3	Dense	32	tanh
Output	Dense	1	*Linear*

Parameter values for applying DDPG algorithm is given as following tables.

Table 3. Hyperparameters for DDPG Algorithm.

Parameter	Definition	Value
P	Initial number of steps	500
M	Size of minibatch	64
N	Exploration noise	$0.5 \times 0.995^{(\text{number of training})}$
C	Updating frequency of the target network	1
$\|D\|$	Size of experience pool	10000
α	Learning rate	actor-10^{-4}, critic-10^{-3}
γ	Discount factor	0.9
τ	Soft replacement factor	0.001

Parameter values for reward configuration are given in table 4.

Table 4. Parameters for Reward Configuration.

Parameter	Definition	Value
R_1	Negative reward when collide with obstacles	-1000
R_2	Negative reward when depart from the map	-1000
k_3	Coefficient	10
k_4	Coefficient	5
k_5	Coefficient	20
k_6	Coefficient	3
d_0	Radius of target region	0.5

4.3. Simulation Result and Analysis

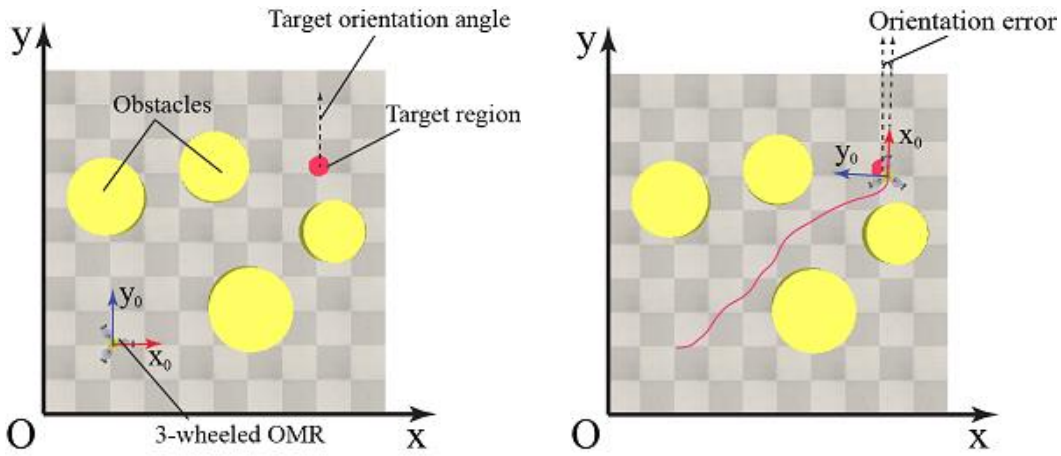

Figure 3. Pose Control Result of Robot.

Figure 3 shows the control result considering the orientation angle of the 3-wheeled omnidirectional mobile robot using DDPG algorithm. In the figure, yellow cylinders represent obstacles and red circle is the target region. The red and blue segments intersecting vertically with each other represent the local coordinates fixed to the robot and the robot path is marked with pink.

As can be seen from the figure, the robot successfully overcame the obstacle, reached the target region, and was moving almost similar to the desired direction at the final moment.

The result of comparing the position error at the target point after path planning in the same environment using the previous algorithm proposed algorithm are presented in Table 5.

Table 5. *Position Error from Target point.*

No	Previous algorithm	Proposed algorithm
1	0.413808	0.041575
2	0.42216	0.476927
3	0.424337	0.369148
4	0.411378	0.131149
5	0.469893	0.131009
6	0.455647	0.056521
7	0.443464	0.000584
8	0.454903	0.151529
9	0.463397	0.343016
10	0.486252	0.103636
Average	0.444524	0.180509

As we can see in the table 5, the new configuration of the state and reward in the proposed algorithm is enough to solve the problem of the position error from the target point not exceeding the threshold.

Similarly, the result of the calculation of the orientation angle error for different target values is given in Table 6.

Table 6. *Orientation Angle Error.*

No	Target value ( )	Reached value ( )	Error ( )
1	30	34.06459	4.064586
2	60	58.69824	1.301763
3	90	96.6034	6.603395
4	120	125.7604	5.760367
5	150	150.6944	0.694423
6	180	178.3868	1.61317
7	210	216.9049	6.904861
8	240	239.9423	0.057667
9	270	268.2965	1.703503
10	300	305.5362	5.536204
Average	-	-	3.423994

As can be seen from Table 6, it can be seen that the goal of the orientation angle control is achieved by the average $3.423994°$, which is the result of the control by the omnidirectional mobile robot using the proposed algorithm for different target orientation angles.

5. Conclusion

The robot control studied so far has been considered only the robot position and the orientation angle have been ignored. Also, because orientation angle is taken as action space of the robot, so it causes serious problems in real robot's control.

In this paper, we took advantage of omnidirectional mobile robots by considering both position and orientation angle simultaneously in the control of the robot, and proposed a method to control the robot in real time by combining the kinetic model of the robot with the DDPG algorithm. It also improved the convergence speed and accuracy of learning by new configuration of reward function. Simulation results show that the proposed method can achieve a better control of the robot.

Abbreviations

AI	Artificial Intelligence
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DDPG	Deep Deterministic Policy Gradient
UVFM	Univector Field Method
APFM	Artificial Potential Field Method
DQN	Deep Q-Network
OMR	Omnidirectional Mobile Robot

Acknowledgments

We would like to thank Pak Ju Song, Wang Chol Jin and Sin Ju Hyok for their contributions to the study.

This study was supported by Kim Chaek University of Technology.

Author Contributions

Kim Kwang Jin: Conceptualization, Writing
Yun Ji Yon: Formal Analysis, Validation
Ro Kang Song: Methodology, Supervison
Jo Kwang Bin: Resources
Pak Mu Rim: Formal Analysis, Validation

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Ashleigh S, Silvia F. A Cell Decomposition Approach to Cooperative Path Planning and Collision Avoidance via Disjunctive Programming. 49th IEEE Conference on Decision and Control; 2010 Dec 15-17; Atlanta, USA; 2011. 6329-8p.

[2] Christoph Oberndorfer. Research on new Artificial Intelligence based Path Planning Algorithms with Focus on Autonomous Driving [PhM Thesis]. Munich: University of Applied Sciences Munich; 2017.

[3] Koren Y, Borenstein J. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. Proceedings of the IEEE Conference on Robotics and Automation; 1991 Apr 7-12; California, USA; 1991. 1398-6p.

[4] Arora T, Gigras Y, Arora V. Robotic Path Planning using Genetic Algorithm in Dynamic Environment. *IJCA* 2014; 89(11): 8-5p.

[5] Mahadevi S, Shylaja KR, Ravinandan ME. Memory Based A-Star Algorithm for Path Planning of a Mobile Robot. *IJSR* 2014; 3(6): 1351-5p.

[6] Yu ZN, Duan P, Meng LL, et al. Multi-objective path planning for mobile robot with an improved artificial bee colony algorithm. *MBE* 2022; 20(2): 2501-9p.
https://doi.org/10.3934/mbe.2023117

[7] Ren Y, Liu JY. Automatic Obstacle Avoidance Path Planning Method for Unmanned Ground Vehicle Based on Improved Bee Colony Algorithm. *JJMIE* 2022; 16(1): 11-8p.

[8] Sat C, Dayal RP. Navigational control strategy of humanoid robots using average fuzzy-neuro-genetic hybrid technique. *IRAJ* 2022; 8(1): 22-4p.
https://doi.org/10.15406/iratj.2022.08.00239

[9] Jeevan R, Srihari PV, Satya JP, et al. Real Time Path Planning of Robot using Deep Reinforcement Learning. Preprints of the 21st IFAC World Congress (Virtual); July 12-17, 2020; Berlin, Germany; 2020. 15811-6p.

[10] Shi YM, Zhang ZY. Research on Path Planning Strategy of Rescue Robot Based on Reinforcement Learning. Journal of Computers 2022; 33(3): 187-8p.
https://doi.org/10.53106/199115992022063303015

[11] Lucia L, Daniel D, Gianluca C, et al. Robot Navigation in Crowded Environments Using Deep Reinforcement Learning. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(Virtual); October 25-29, 2020, Las Vegas, NV, USA; 2020. 5671-7p.

[12] Phalgun C, Rolf D, Thomas H. Robotic Path Planning by Q Learning and a Performance Comparison with Classical Path Finding Algorithms. *IJMERR* 2022; 11(6): 373-6p.
https://doi.org/10.18178/ijmerr.11.6.373-378

[13] Yang Y, Li JT, Peng LL. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. *CAAI Trans. Intell. Technol* 2020; 5(3): 177-7p.
https://doi.org/10.1049/trit.2020.0024

[14] Zhu AY, Dai TH, Xu GY, et al. Deep Reinforcement Learning for Real-Time Assembly Planning in Robot-Based Prefabricated Construction. *IEEE Trans. Auto. Sci. Technol* 2023; 20(3): 1515-12p.

[15] Chen Jiong. Construction of an Intelligent Robot Path Recognition System Supported by Deep Learning Network algorithms. *IJACSA* 2023; 14(10): 172-10p.

[16] Yun JY, Ro KS, Pak JS, et al. Path Planning using DDPG Algorithm and Univector Field Method for Intelligent Mobile Robot. *IJARAT* 2024; 2(2): 7-11p.
https://doi.org/10.37591/IJART