International Journal of Industrial and Manufacturing Systems Engineering
2025, Vol. 10, No. 3, pp. 44–52
https://doi.org/10.11648/j.ijimse.20251003.11

Research Article

Mobile Robot Control Using Deep Reinforcement Learning and Autoencoder in Dynamic Environment

Choe Ryong Bom, Pak Mu Rim, Ro Kang Song*, Jo Kwang Bin, Yun Ji Yon

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

Abstract

In recent years, with the rapid development of artificial intelligence, many innovative changes have been made in the field of intelligent mobile robot development. In the field of control and navigation of mobile robots, learning-based methods have many advantages over traditional ones. The study of mobile robot control methods using deep reinforcement learning is a remarkable area in the development of mobile robots that must operate in dynamic environments. In the previous studies, the proposed robot control algorithms using deep reinforcement learning are mostly based on the given target point and obstacle information, the robot path planning is performed, and the corresponding control is based on the obtained path. The DDPG-based method is a typical example. However, in dynamic environments, DRL based robot path planning requires a state of target point and obstacles information, which leads to a large amount of computation, resulting in extremely long convergence time and even non-convergent cases. In this paper, we propose a new method for mobile robot control in dynamic environment that solves the dimensional problem by extracting the features of the configuration of obstacles using autoencoder and learning the DDPG algorithm based on the obtained features. Simulation results show that the proposed algorithm can effectively solve the mobile robot control problem in dynamic environment.

Keywords

DDPG Algorithm, Autoencoder, Mobile Robot, Path Planning, Dynamic Environment

1. Introduction

In the control and navigation of intelligent robots, path planning is a very important task.

A path is the orbit that the robot must travel. In other words, it is a geometric spatial curve that the robot must move from the initial position to the target position to achieve a given goal under the certain constraints. The obtained path should not only be optimal in terms of energy consumption but also convenient to implement the path-following control. [1].

Various methods for robot path planning have been investigated. Robot path planning methods include graph theory based methods such as BFS, DFS, Dijkstra, A* algorithm, vector field based methods such as artificial potential field method, univector field method and mathematical optimization based methods such as genetic algorithms, fuzzy theory, PSO (particle swarm optimization) or ACO (ant colony optimization). [2-12].

In recent years, with the rapid development in robotic technology and artificial intelligence, reinforcement learning

*Corresponding author: GS.RO@star-co.net.kp (Ro Kang Song)

Received: 16 October 2025; **Accepted:** 31 October 2025; **Published:** 11 December 2025

based path planning methods have been widely introduced. The RL based path planning method is widely used in path planning because it does not require environmental information and obtains the optimal path during interaction with the environment. In particular, in path planning of intelligent mobile robots in unknown environments, RL based path planning methods are superior to traditional path planning methods. [13, 14].

To solve the curse of dimensionality problem caused by increase of dimensionality of state vectors in reinforcement learning, deep reinforcement learning algorithms which includes deep learning and reinforcement learning have been studied and are actively introduced in the field of robot path planning. [15].

A typical path planning method using deep reinforcement learning is the DDPG based method. In general, the path planning method using DDPG algorithm is not suitable for robot path planning in dynamic environment because it considers only the position of the robot in the state, considering the target point and the position of the obstacles given. A method for robot path planning in continuous action space by combining the DDPG algorithm and the univector field method has been proposed. [16].

Previously proposed DRL based path planning methods are not applicable in dynamically changing environments due to the long learning time, which requires re-training in different environments. To solve this problem, we must reflect the obstacle and target information at the input of the DDPG algorithm. However, this increases the computational cost exponentially due to the increasing order of the state vector, which leads to a longer learning time and even a non-convergent case.

To solve this problem, we propose the following mobile robot control algorithm.

First, we configure the obstacle array and extract feature vector while reducing dimensionality of obstacle array using autoencoder.

Next, we configure state vector including feature vector extracted from autoencoder and position of robot and target, and train DDPG algorithm using this state vector.

The rest of this paper is as follows.

In section 2, analysis of previous literature on DRL based path planning method and autoencoder.

In section 3, a new method of mobile robot control using DDPG and autoencoder is proposed.

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

Section 5 gives conclusion.

2. Related Work

2.1. DDPG Algorithm

The DDPG algorithm is one of the typical DRL algorithm which 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. Actor network learn to find the action $a = \mu_{\varphi}(s)$ to take in that state using policy gradient according to the state of the agent, 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 1.

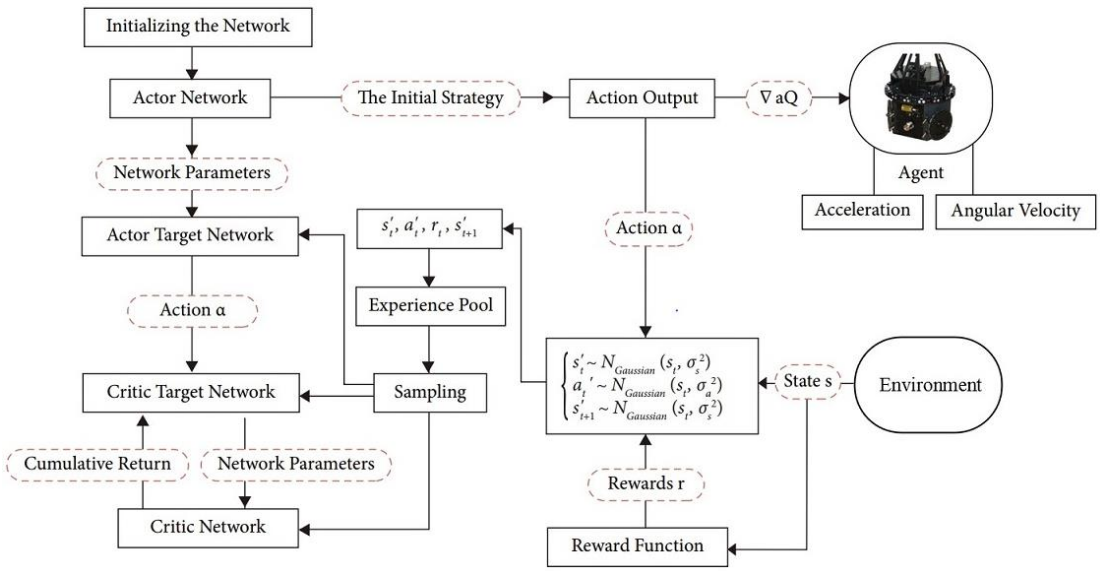

Figure 1. *Configuration of DDPG Algorithm.*

Summarizing the DDPG algorithm,

1. Initialize the experience repository D
2. 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

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

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

2) Perform the action a and move to the next state s'. At this time, receive a reward r and save transition information in experience store D.
3) Randomly select K transition information in D.
4) Get target value of critic network.

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

5) Calculate loss of critic network.

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

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

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

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

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

8) Update parameters of target critic and target actor networks.

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

$$\phi' = \tau\phi + (1-\tau)\phi'$$

2.2. DDPG Based Robot Path Planning

In DDPG based path planning algorithm, the agent is robot, state includes current position of robot, obstacles and target. In DDPG based path planning algorithm, the action space consists of a continuous number of values, such as the acceleration of the robot or the orientation angle to move.

The reward the robot receives consists of positive rewards received when the robot reaches the target, negative rewards received when the robot bumps into obstacles or leaves from the map, and rewards added to obtain the shortest path.

In path planning using the DDPG algorithm, the action space is composed of a continuous number of values, such as the acceleration of the robot or the orientation angle to move.

The reward the robot receives consists of positive rewards received when the goal is reached, negative rewards received when hitting an obstacle or leaving the map, and rewards added to obtain the shortest path.

DDPG based robot path planning algorithm can generate the robot's movement path in continuous action space from the characteristics of algorithm itself.

However, there is a disadvantage that it is difficult to reflect the position information of the obstacle as the state vector.

That is, to reflect the obstacles information as state vector in a dynamically changing environment, the dimension of the state vector becomes very large and the curse of dimensionality arises.

Hence, if the state information in the environment is extracted as a feature vector in advance using a certain dimensionality reduction algorithm, it can shorten the training time of the DDPG algorithm and achieve more efficient robot path planning.

2.3. Autoencoder

Autoencoders are artificial neural networks capable of learning dense representations of the input data, called latent representations or codings, without any supervision (i.e., the training set is unlabeled). These codings typically have a much lower dimensionality than the input data, making autoencoders useful for dimensionality reduction, especially for visualization purposes. Autoencoders also act as feature detectors, and they can be used for unsupervised pretraining of deep neural networks. Lastly, some autoencoders are generative models: they are capable of randomly generating new data that looks very similar to the training data. For example, we could train an autoencoder on pictures of faces, and it would then be able to generate new faces.

Autoencoders simply learn to copy their inputs to their outputs. This may sound like a trivial task, but as you will see, constraining the network in various ways can make it rather difficult. For example, you can limit the size of the latent representations, or you can add noise to the inputs and train the network to recover the original inputs. These constraints prevent the autoencoder from trivially copying the inputs directly to the outputs, which forces it to learn efficient ways of representing the data. In short, the codings are byproducts of the autoencoder learning the identity function under some constraints.

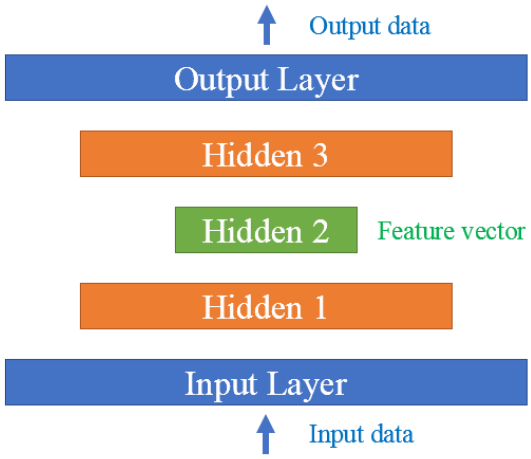

Figure 2. Autoencoder.

Autoencoders with several hidden layers are called stacked autoencoder. The architecture of a stacked autoencoder is typically symmetrical with regard to the central hidden layer (the coding layer).

Stacked autoencoder is efficient as dimensionality reduction algorithm because we can get useful representation of input data in middle hidden layer.

3. Proposed Method

DDPG is an efficient DRL algorithm that can perform the path planning of the robot in a continuous action space. However, for application in dynamically changing environments, as the dimension of the state vector increases, the computation cost increases, even if the algorithm does not converge, as the state contains the obstacle information.

Even without considering the obstacle information, the increase of learning time with the size of the environmental grid is significant. Figure 3 shows the convergence curves of the DDPG algorithm with the environment grid of 10×10 and 20×20.

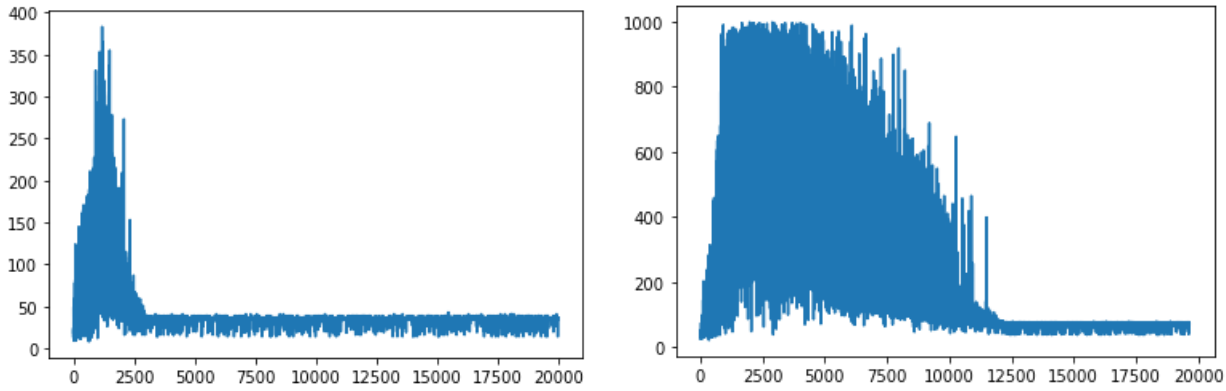

Figure 3. Convergence curve of DDPG algorithm (grid 10×10-left, grid 20×20-right).

In Figure 3 horizontal axis represents the number of episodes and vertical axis represents the time steps in each episode. It can be seen from the Figure 3 that the number of iterations or time steps in each episode involved in the learning process is very large when the environment grid is dense twice, which in this case leads to a great increase in the computation cost.

Table 1. Convergence time comparison according to grid size.

Grid size	Convergence time, s	Total time steps before convergence	Number of episodes before convergence
10×10	0.14	462 815	2 437
20×20	1.38	5 849 234	11 843

When grid size is 40×40, the algorithm does not converge within given limited number of episodes and diverges.

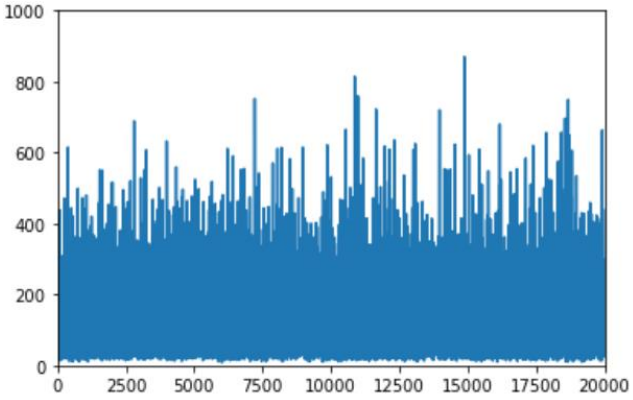

Figure 4. Convergence curve of DDPG algorithm (grid 40×40).

The denser the environment grid, the more accurate the obstacle information can be reflected, and thus the advantage of robot control, but the resulting dimensional problem makes this impossible.

3.1. Feature Vector Extraction of Obstacle Arrangement with Autoencoder

To solve the dimensional problems that arise when the environment grid is dense and to implement path planning in dynamic environments, the dimension of the state to be reflected in the deep reinforcement learning algorithm must be reduced.

Thus, we need to extract the feature vector of the obstacle state using a certain dimensionality reduction algorithm and use it to train the deep reinforcement learning algorithm.

The autoencoder is one of the most efficient dimensionality reduction algorithms.

Using the autoencoder, we can obtain an efficient low-dimensional representation of the obstacle array in its middle layer.

First, we discretize the environment with the 40×40 grid size and limit the number of obstacles below 100. For each discretized position, we set 1 for the position with obstacles and 0 for the other positions.

Then, a 40×40 discretized map is unfolded to form a 1-D vector with 1600 elements.

That is, 1600 elements are 0 or 1.

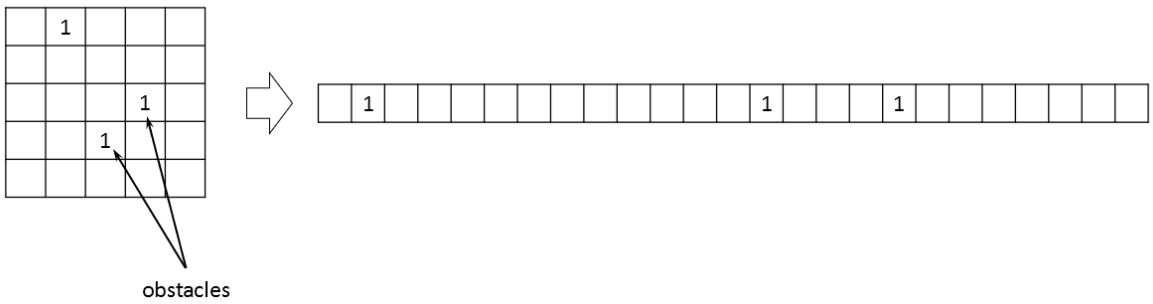

Figure 5. *Collection of the train data for autoencoder.*

We randomly select 100,000 data by varying the position and number of obstacles.

80 percent of them are divided into training data and rest are divided into testing.

The network architecture uses a stacked autoencoder composed of all hidden layers.

The network parameters of the autoencoder are given in Table 2.

Table 2. *Network Parameters of Autoenoder.*

Layer	Type	Number of neurons	Activation
Input		1600	
Layer1	Dense	100	ReLU
Layer2	Dense	10	ReLU
Layer3	Dense	100	ReLU
Output	Dense	1600	-

The training parameters of the autoencoder are given in Table 3.

Table 3. *Training Network Parameters of Autoenoder.*

Parameter	Value
Optimizer	Nadam
Learning rate	0.0001
Loss function	MSE
Number of epoch	100

After the autoencoder is trained, the feature vector is extracted using the encoder.

In other words, the output of layer 3 in the autoencoder has 10 elements as feature vectors of the obstacle state.

Thus, if we use new state vector which is combined the extracted feature vector of autoencoder with the robot's position in the DDPG algorithm, we can solve the mobile robot control problem in dynamic environment more efficiently while guaranteeing the convergence of DDPG algorithm and reducing the computational cost.

3.2. Environment Configuration Using DDPG and Compressed Feature Vector

Now, we apply DDPG algorithm.

In the DDPG based path planning, the agent-robot takes the state as input and learns to output the orientation angle that the robot must move.

The robot performs the action in a continuous environment of a certain size.

The rewards that the robot receives at each step consist of negative rewards that it receives when bumping into obstacles or leaving from the map, negative rewards expressed as the inverse of the distance between robot and target, and reward with varying orientation angle.

$$R=\begin{cases}100, & \text{Reachead at the target}\\ -100, & \text{Bumped into obstacles}\\ -100, & \text{Leaving from the map}\\ -d, & \text{else}\end{cases} \tag{1}$$

Here, d is the distance between current robot position and target.

That is, learning the network to maximize reward means that the robot is trained to reach the target in the shortest time without hitting the obstacle.

Then, the robot state consists of the current position of robot x, y, the target position x_t, y_t, and the feature vector obtained as the output of the autoencoder.

$$s=\begin{bmatrix}x\\ y\\ x_t\\ y_t\\ f\end{bmatrix} \tag{2}$$

Here, f is feature vector obtained from autoencoder.

The state of the robot is the input to the actor network, and the actor network output an angle φ that represents the direction of the robot's motion.

Since the range of the robot's moving direction angle must be $[-\pi, \pi)$ we have restricted the output result of the last layer of the actor network to be the same.

When the robot's moving direction is output, it moves the robot a certain distance d_0 in that direction, and then the robot's position changes as follows:

$$\begin{aligned}x_{k+1}&=x_k+d_0\cos\varphi\\ y_{k+1}&=y_k+d_0\sin\varphi\end{aligned} \tag{3}$$

Here d_0 is the robot's displacement when the robot moved with a normal moving speed during the sampling time, and k represents the step number.

The overall configuration of the proposed method is shown in Figure 6.

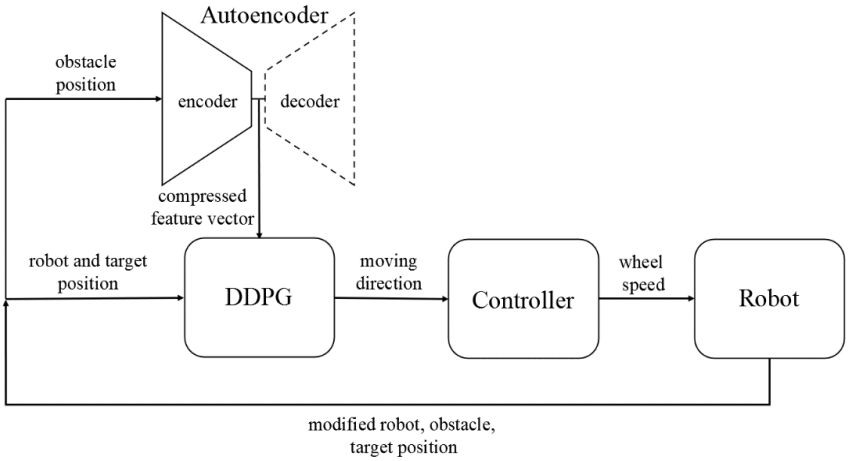

Figure 6. *Configuration of proposed method.*

4. Simulation Result and Analysis

4.1. Configuration of the Environment

Simulation was performed on 8GB RAM, Intel Core-i7-11800H processor with 2.3GHz operation frequency and NVIDIA RTX 3050 GPU on the robot experiment platform *CoppeliaSim*.

Map size is 4×4, and number and position of obstacles, position of the target are randomly set.

Structure of actor and critic network is given in Table 4 and 5.

Table 4. Parameters of the Critic Network.

Layer	Type	Number of parameters	Activation
Input		15	
Layer1	Dense	256	ReLU
Output	Dense	1	Linear

The input of the Critic network consists of the current position of the robot, the position of the target point, the feature vector obtained by the autoencoder, and the orientation angle φ output from the actor network, and the output is a state value Q.

Critic network is regression network to obtain state values, so no activation functions are used in the output layer.

Table 5. Parameters of the Actor Network.

Layer	Type	Number of parameters	Activation
Input		14	
Layer1	Dense	1024	tanh
Layer2	Dense	256	tanh
Output	Dense	6	$\pi \cdot \tanh$

The input of the Actor network consists of the current position of the robot, the position of the target point, and the feature vector obtained by the autoencoder, and the output is an orientation angle φ.

The output of the Actor network is in the range $[-\pi, \pi)$ as the orientation angle the robot must move, so the activation function of the output layer is given by $\pi \cdot \tanh$.

The hyperparameters for applying the DDPG algorithm are given in Table 6.

Table 6. Hyperparameters for DDPG algorithm.

Parameter	Definition	Value
P	Initial number of steps	500
M	Size of minibatch	128
N	Exploration noise	$0.5*0.995^{(\text{num of training})}$
C	Updating frequency of the target network	1
$\lvert D \rvert$	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

4.2. Simulation Result and Analysis

In the path planning simulation, the path planning of a mobile robot is performed using the proposed method when the position, number of obstacles and position of the target are changed.

Figure 7 shows the results of the path planning simulation using the proposed method.

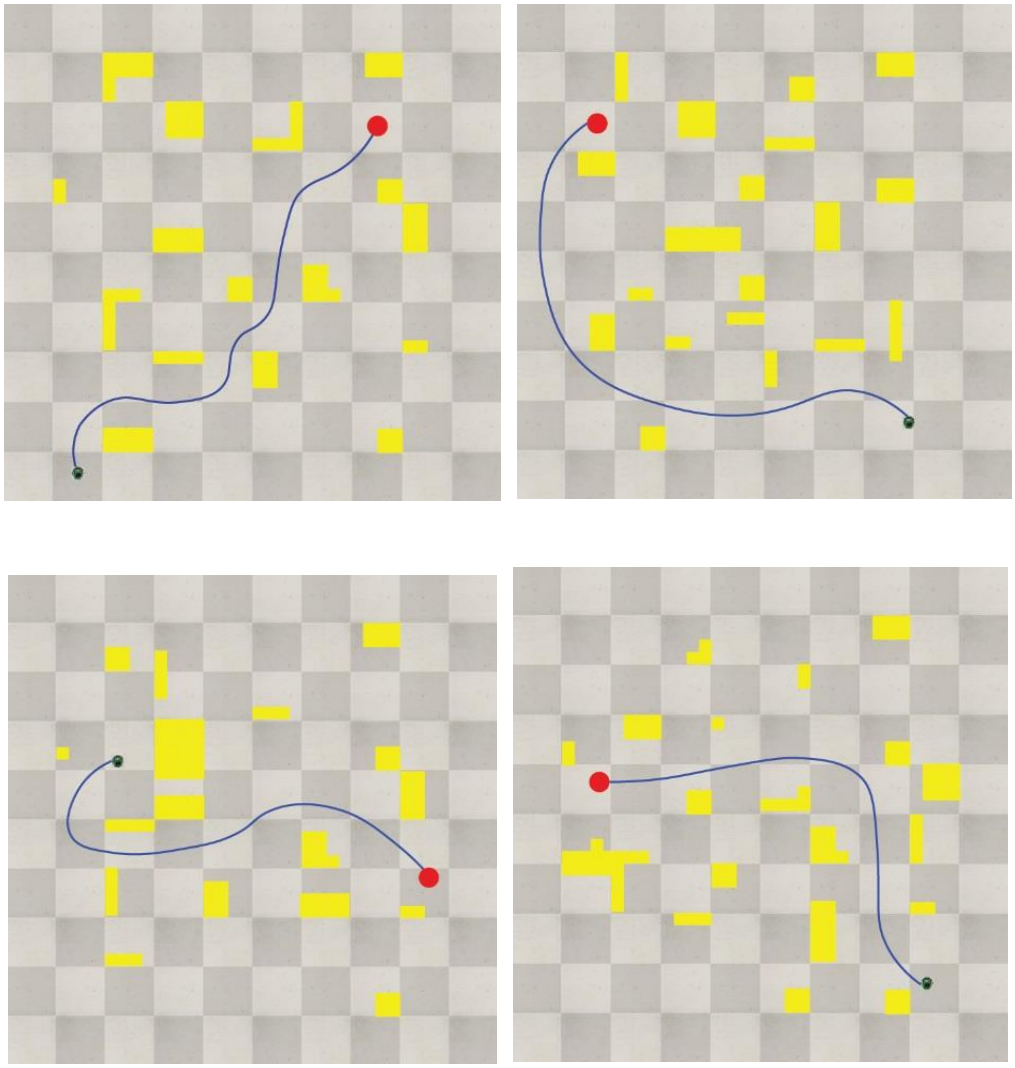

Figure 7. *Path planning result using proposed method.*

As can be seen, the learning was performed by reflecting the position information of the obstacles, thus generating an efficient path under the various obstacles and target positions.

Meanwhile, we measured the inference time of the proposed method to ensure the real-time performance of the path planning.

The inference time of the proposed algorithm is below 0.07 s, and when the maximum moving speed of obstacle is 1 m/s, it is found that the change in obstacle position for the time is 0.07 m, which is not larger than the grid size specified by the DDPG algorithm, so it is fully applicable in dynamically changing environments.

The path planning method using DDPG algorithm not only can overcome local minima and successfully perform the robot path planning in continuous environment, but also is convenient for robot navigation because the robot's action space is continuous.

As the experimental results show, the proposed method is more optimized than the existing algorithm for any obstacle condition and generates a reasonable path from the viewpoint of robot control.

5. Conclusion

In general, we use the DDPG algorithm, a deep reinforcement learning algorithm, for intelligent mobile robot control in continuous action space.

The advantage of DRL based path planning methods is that path planning in unknown environments performs better than existing algorithms.

To do this, the position information of the obstacles must be reflected in the state, and as the order of the state increases, the number of operations increases rapidly, resulting in a long learning time and even in the case of the algorithm not convergent.

Hence, the feature vector of states containing the position information of obstacles is extracted using the dimensionality reduction algorithm, and the DDPG algorithm is trained using it to take advantage of the reinforcement learning algorithm.

We have extracted the features of the obstacle state using the autoencoder and trained the DDPG algorithm using the obtained feature vector, thus providing a method to solve the intelligent mobile robot control problem in dynamic environment.

Simulation results show that our method can fully respond to the dynamic changes of obstacles while ensuring the convergence of the DDPG algorithm and generate a reasonable optimized path from the viewpoint of robot control.

Abbreviations

DDPG	Deep Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
BFS	Breadth First Search
DFS	Depth First Search
PSO	Paarticle Swarm Optimization
ACO	Ant Colony Algorithm
RL	Reinforcement Learning

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.

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. 6329-8 p.

[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. 1398-6 p.

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

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

[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): 2501-9 p. 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): 11-8 p.

[8] Sat C, Dayal RP. Navigational control strategy of humanoid robots using average fuzzy-neuro-genetic hybrid technique. *IRAJ* 2022; 8(1): 22-4 p. 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. 15811-6 p.

[10] Shi YM, Zhang ZY. Research on Path Planning Strategy of Rescue Robot Based on Reinforcement Learning. Journal of Computers 2022; 33(3): 187-8 p. 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. 5671-7 p.

[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): 373-6 p. 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): 177-7 p.

[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): 1515-12 p.

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

[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): 7-11 p. https://doi.org/10.37591/IJART