Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 4115767, 21 pages
https://doi.org/10.1155/2022/4115767

Research Article

Mobile Robot Application with Hierarchical Start Position DQN

Emre Erkan [1] **and Muhammet Ali Arserim** [2]

[1]*Department of Electronic Communication, Batman University, Batman 72500, Turkey*
[2]*Electrical and Electronics Engineering Department, Dicle University, Diyarbakir 21280, Turkey*

Correspondence should be addressed to Emre Erkan; emre.erkan@batman.edu.tr

Received 27 April 2022; Revised 26 July 2022; Accepted 3 August 2022; Published 5 September 2022

Academic Editor: Abdul Rehman Javed

Advances in deep learning significantly affect reinforcement learning, which results in the emergence of Deep RL (DRL). DRL does not need a data set and has the potential beyond the performance of human experts, resulting in significant developments in the field of artificial intelligence. However, because a DRL agent has to interact with the environment a lot while it is trained, it is difficult to be trained directly in the real environment due to the long training time, high cost, and possible material damage. Therefore, most or all of the training of DRL agents for real-world applications is conducted in virtual environments. This study focused on the difficulty in a mobile robot to reach its target by making a path plan in a real-world environment. The Minimalistic Gridworld virtual environment has been used for training the DRL agent, and to our knowledge, we have implemented the first real-world implementation for this environment. A DRL algorithm with higher performance than the classical Deep Q-network algorithm was created with the expanded environment. A mobile robot was designed for use in a real-world application. To match the virtual environment with the real environment, algorithms that can detect the position of the mobile robot and the target, as well as the rotation of the mobile robot, were created. As a result, a DRL-based mobile robot was developed that uses only the top view of the environment and can reach its target regardless of its initial position and rotation.

1. Introduction

The path planning is one of the important topics of the robotics since navigating a robot to a desired destination without collision is a significant task [1]. The significance of such a task is estimated to increase even further [2, 3] due to evolving interaction between robots and human beings. This means the human beings will have more places shared with robots giving rise to the importance of path planning. In literature, conventional path planning techniques such as A-star, ant colony optimization, and Artificial Potential Field (APF) can be encountered [4] which are not efficient to address the issues in real-time within complex environments [5]. With respect to achieving a more robust solution, deep learning (DL) has been successfully implemented for path planning [6]. However, DL needs labeled data in order to learn the environment which is a time-consuming task [7]. To overcome the latter problem, the researchers have recently adopted the deep reinforcement learning (DRL) technique as one of the efficient solutions.

Reinforcement learning (RL) tries to maximize the numerical reward signals by focusing on actions that must be taken depending on the specific cases [8]. Instead of guiding the RL agent to perform a specific task, it is asked to learn which actions are more rewarding [9]. The RL agent, interacting with the environment sufficiently, learns which action contributes more to the cumulative reward. In another word, it learns the best state-action pair. However, exploring the environment becomes costly in the case of increasing numbers of states and actions due to the curse of dimensionality [10]. Therefore, it may become impossible to solve some of the real-world problems with high dimensions of action and state spaces by using traditional reinforcement learning [11].

The remarkable progress of the DL has also significantly affected the RL and resulted in the DRL method, which is a combination of both methods [12]. Significant achievements have been reported for arcade games [13, 14], board games [15, 16], autonomous driving [17], robots interacting with humans [18], drones [19], unmanned surface vehicles [20],

robot navigation problems [21], and robotic surgery [22], which all are examples of complex systems with multiple states.

Several recent studies on path planning of mobile robots by using the DRL method are briefly provided below:

(i) Zheng et al. [1] proposed a method that optimizes the path planning process of a mobile robot for an unmapped environment. In the proposed method, a memory module based on the special structure of long short-term memory (LSTM) was introduced. Thanks to this method, the long-term memory capacity of the robot was increased, and the number of trials and calculation time required for training was shortened.

(ii) Gong et al. [5] suggested optimizing the deep deterministic policy gradient (DDPG) network with LSTM to solve the path planning problem of the mobile robot in environments with static obstacles. In this method, a mixed noise along with a more reasonable reward function was used for quick training. The proposed algorithm and DDPG-based algorithms [23, 24] were experimentally compared and the advantages of the proposed algorithm were demonstrated in a complex environment in terms of exploration efficiency, optimum path and time.

(iii) In Zhou et al. [7], an improved DQN algorithm was proposed for the path planning problem of patrolling robots. In this study, the reward penalty functions were improved, and the sparse reward problem was resolved by optimizing the state-action space by adding new reward value points. The advantages of the proposed algorithm over the classical DQN algorithm were experimentally demonstrated.

(iv) In Wang et al. [25], an improved DQN method was proposed by focusing on the problem of poor exploration and sparse reward in mobile robot path planning. The reward function was improved by combining the artificial potential field method with the reward function to optimize the state-action space. The performances of the proposed method and classical DQN were compared.

(v) In Xing et al. [26], the area division Deep Q-Network (AD-DQN) method was proposed. A mobile wireless powertrain robot was able to determine the optimal path with the proposed method in terms of charging a large number of IoT devices.

(vi) Huang et al. [27] proposed a method that determines two reward thresholds for solving the anomalous reward problem encountered in the path planning process of a mobile robot in an unknown dynamic environment. The improvement in value-based DRL algorithms was experimentally demonstrated with the proposed method.

(vii) Yu et al. [28] proposed a mobile robot path planning method based on neural networks and hierarchical reinforcement learning. The performance of the proposed method was evaluated for an environment with static obstacles.

(viii) Wang et al. [29] proposed a DDQN-based method with prioritized experience replay (PER) for mobile robot path planning. Simulation experiments were shown that this method had a better convergence rate and success rate than classical DQN in unknown environments.

(ix) Zhang et al. [30] focused on similar sample redundancy problems that negatively affect the training of the DQN algorithm for environments with static obstacles. An algorithm was proposed in this study in order to optimize the training samples using similarity scanning matrix such that more useful training examples can be used.

(x) Ruan et al. [31] proposed a D3QN-based LN-D3QN algorithm for mobile robots that avoid the obstacles. The proposed algorithm accelerated the neural network training by normalizing the layers of the neural network. In addition, it was observed that the learning time was reduced by using prior experience repetition. Experiments showed that the robot trained with the LN-D3QN algorithm can adapt to unknown environments faster than that of the D3QN algorithm.

(xi) Kim et al. [32] proposed a DQN method combined with a Gated Recurrent Unit (GRU) to solve the path planning problem of a mobile robot.

On the contrary to the above listed works, in this study, a DRL agent was trained in a discrete virtual environment with sparse rewards and focused on the real-world targeting problem of a mobile robot using these DRL network parameters. The contributions of this study can be listed as follows:

(i) An improved DQN algorithm is proposed for large environments with sparse rewards in order to increase the convergence rate and success rate according to the classical DQN algorithm.

(ii) Network parameters of the agent trained in the Minigrid virtual environment were used for the first time in the real environment.

(iii) A pixel-based image processing algorithm is proposed to detect the rotation of the mobile robot.

(iv) The model trained in the virtual environment is used to provide a more efficient operation in the real environment.

(v) The proposed design allows the mobile robot to be used in swarm robotics and multi-agent systems with the advantage of low cost and long-range controllability from a central computer.

The rest of the article is organized as follows. Section 2 introduces the virtual environment used in DRL agent training. In Section 3, the DQN algorithm with hierarchical

starting position is presented as an efficient method, in terms of training time and successfulness rate, compared to the conventional DQN algorithm. Section 4 describes the design and control of the mobile robot used in the real environment. Section 5 explains how to pair a discrete virtual environment with a real-world environment and the mode of operation of the DRL-based mobile robot which can achieve its target regardless of its starting position and rotation. Section 6 explains the implementation stage and presents the performance analyses. Finally, the paper is concluded in Section 7.

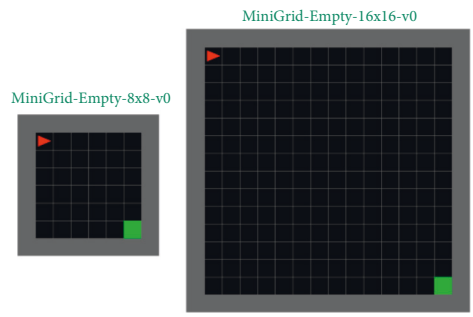

Figure 1: Environments to be used in the application.

2. Structure of Virtual Environment

OpenAI Gym is a toolkit designed for RL research. Creating a common interface, OpenAI Gym contains many different tasks and environments [33]. This study uses one of the OpenAI Gym platforms, the Minimalistic Gridworld Environment (MiniGrid). The MiniGrid Environment platform has numerous environments with increasing difficulty levels [34].

The studies regarding natural language processing [35–37], reinforcement learning for environments with sparse reward signals [38–45] and hierarchical-based reinforcement learning in natural language processing [46] can be found for MiniGrid environment. But the studies were limited to the virtual environment. This study focuses on real environment application by using the agent trained in virtual environment.

The environments shown in Figure 1 are used in the application. Both environments represent an empty room, and the agent's goal is to reach the green target plot in an environment that provides sparse reward. The only difference between the two environments is the number of plots constituting the environment. Therefore, this section where virtual environment analysis is performed explains the structure of MiniGrid-Empty-8 × 8-v0 environment.

2.1. States. The MiniGrid-Empty-8 × 8-v0 environment consists of 8 × 8 plots. The state of the virtual environment can be obtained in three different ways. As shown in Figure 2, these include;

(i) RGB image consisting of a combination of 32 pixels × 32 pixels plots by default,

(ii) String structure in which each plot is encoded with two characters,

(iii) 3-dimensional matrix where each plot is encoded with numerical values.

In the application, the state of the virtual environment is obtained as a 3-dimensional matrix. As the number of parameters is much less compared to the RGB image, and the parameters consist of numerical values, it is easy to make them suitable for artificial neural network (ANN) use. As the number of data entered in the ANN increases, the network becomes more complex and the outer wall whose state never changes is removed to make the network simpler. The state parameters of the environment of a 3-dimensional matrix structure are flattened as in Figure 3 so that they can be used in ANNs.

As shown in Figure 3, each plot has three parameters (object, color, direction). These parameters are added consecutively as in the figure, resulting in an array of 108 parameters showing the state of the environment. The array created for the MiniGrid-Empty-16 × 16-v0 environment has 588 parameters.

2.2. Actions. Practically, the agent can make 3 different movements. A numerical value is defined for each movement made by the agent. These are

(i) Turn 90° left ⟶ 0.

(ii) Turn 90° right ⟶ 1.

(iii) Move forward 1 unit ⟶ 2.

2.3. Reward Function. In the application, the total reward value that the agent can receive in a level ranges from 0 to 1. In order for the agent to complete the episode, the agent must either reach its target or take the maximum number of steps. The maximum number of steps for the application was determined as 50. If the agent reaches at its target, the reward function is as shown in the following equation:

$$R = 1 - 0,9 * \frac{\text{Number of steps}}{\text{Maximum number of steps}}. \quad (1)$$

According to the reward function, as the number of steps increases, the reward earned decreases, thus the agent should find the shortest path to maximize its reward. If the agent takes the maximum number of steps, the episode ends and the agent gets 0 points because it fails.

3. DQN and HSP-DQN

With the advancement in DL, the Markov Decision Process can be solved with deep neural networks [47]. As one of such methods, DQN is successful in some areas at the human level [13]. DQN is a DRL algorithm combining supervised learning and RL techniques [13]. The structure of DQN, which is frequently used in fields such as games [13], robotics [48], and natural language processing [49], is shown in Figure 4.

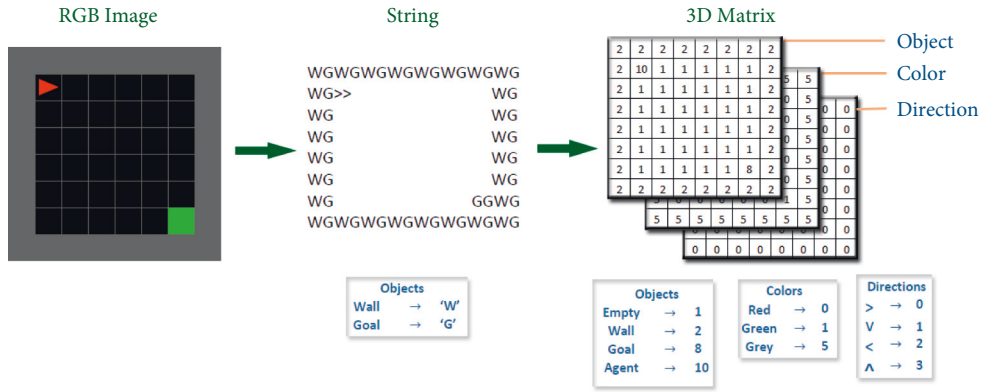

Figure 2: Virtual environment status obtained in three different ways.

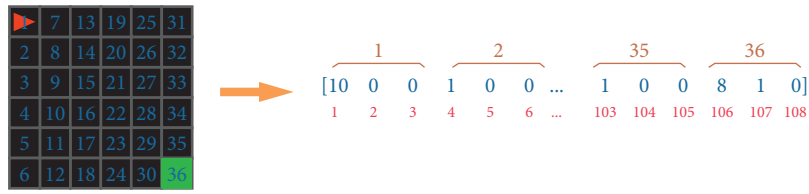

Figure 3: Flattening virtual environment state parameters.

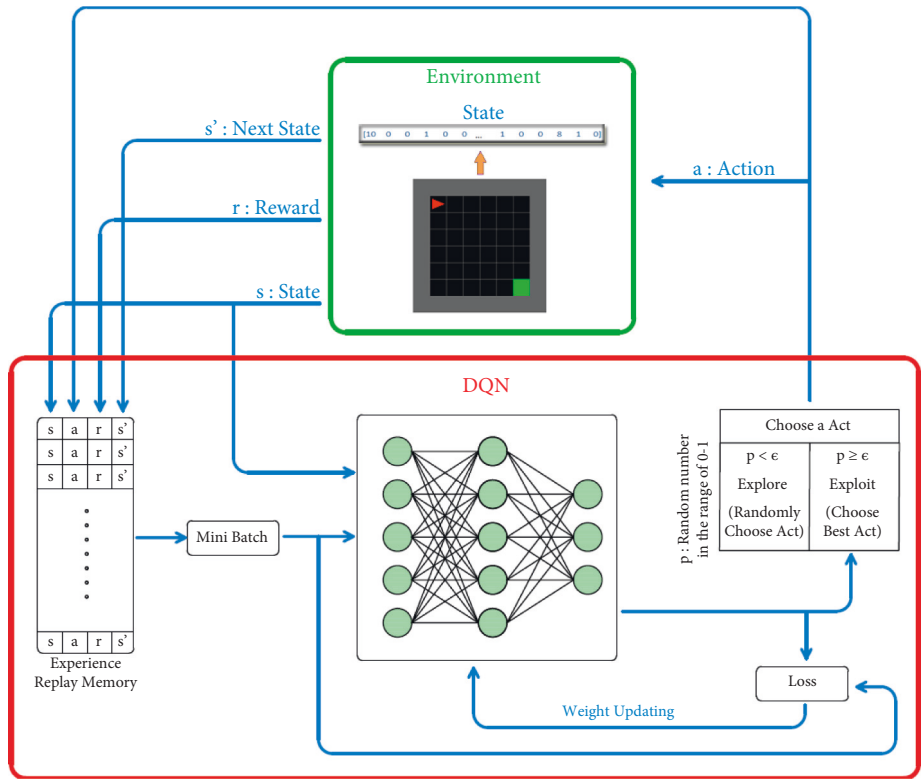

Figure 4: Structure of DQN.

In the DQN algorithm, the agent makes random movements at the beginning of the training and records its experiences about the environment in the experience replay memory. How much reward (r) the action (a) brings in a state (s) and the new state (s') after the action are recorded in this recording field. When the experiences obtained in this way reach a sufficient number, the experience equal to the size of the mini-batch is randomly selected from the experience replay memory and used for the training of the Q-network. So, the network is not affected by local minimums. Also, in the DQN algorithm, the agent determines its action using Epsilon-Greedy (ϵ). It ϵ can take values between 0 and 1. ϵ A value close to 1 indicates that the agent focuses on exploration, while a value close to 0 indicates that

it focuses on making the best action determined by its experience. For this reason, at the initial stage of training, ϵ the agent is allowed to explore the environment by choosing a value close to 1. As the episodes progress, ϵ its value is proportionally reduced, allowing the agent to choose the best action using its experience. The DQN algorithm is very practical for discrete and small-sized environments due to its described structure.

In the application, MiniGrid-Empty-8 × 8-v0 and MiniGrid-Empty-16 × 16-v0 environments are used as virtual environments. The agent was trained separately for the specified environments with the DQN algorithm and the HSP-DQN (Hierarchical Initial Position DQN) algorithm proposed by us, and the algorithm performances were examined. The network architecture in Figure 5 is used for both algorithms.

In the real environment application, it is desired that the mobile robot reaches its target, regardless of its starting position and direction. Therefore, in the training with DQN, the position and the direction of the agent are chosen randomly in each new episode.

However, in the training with HSP-DQN, a region where the agent has a high probability of reaching the target with random movements is determined, and the training is started in this region. The agent is started in a random position and direction, provided that it stays in the determined region in each new episode. After the network is sufficiently trained for this region, a new training region is determined and ϵ value is increased, allowing for the agent to re-explore. There are two regions at this stage; the region where the training was performed for the first time and the region where the training was already performed. Both regions are used during training. In each new episode, one of the two episodes is chosen alternately and the agent is started in a random position and direction. In this way, the previous information of the network can be kept up to date and new regions can be explored. At this stage, after the network is sufficiently trained, the region trained for the first time is included in the previously trained region. ϵ value is increased again and a new exploration region is determined. Thus, the network is trained gradually. These stages continue until the network learns the whole environment. Figure 6 shows the application of the HSP-DQN algorithm in the MiniGrid-Empty-8 × 8-v0 environment.

Figure 7 shows DQN and HSP-DQN reward graphs for MiniGrid-Empty-8 × 8-v0 environment. As the HSP-DQN algorithm focuses on exploration in each new region definition, sections where it fails or gets low scores are observed. However, the graphics show similarity after 1500 s.

Table 1 shows performance data of DQN and HSP-DQN. In both algorithms, successful results were obtained at the end of 3000 sections. Although the total number of steps of HSP-DQN is slightly less than that of DQN, it does not provide an obvious advantage.

As the MiniGrid-Empty-8 × 8-v0 environment area is not large, the DQN algorithm has been successful in learning the environment.

Algorithm performances are also compared for the MiniGrid-Empty-16 × 16-v0 environment with a larger area. Figure 8 shows the application of the HSP-DQN algorithm in the MiniGrid-Empty-16 × 16-v0 environment.

Figure 9 shows DQN and HSP-DQN reward graphs for MiniGrid-Empty-16 × 16-v0 environment. For the DQN algorithm, the number of successful episodes in the first 2000 episodes is quite low. At the end of 8000 episodes, it is observed that it often fails. In the HSP-DQN algorithm, successful and unsuccessful episodes are observed up to 6000 episodes, but in the following episodes, it is seen that the failures became sparse and the reward interval tended to narrow, that is, the agent explores short paths.

Table 2 shows performance data of DQN and HSP-DQN. Accordingly, the successful episode rate of HSP-DQN is quite high compared to DQN. The total number of steps of DQN at the end of 8000 episodes is approximately 84% more than HSP-DQN. In other words, the training period of the DQN, which has a low success rate, is also quite long compared to the HSP-DQN. Also, at the end of 8000 episodes, the Epsilon-Greedy value decreases to 0.018 in DQN, while this value is 0.178 for HSP-DQN. So, the HSP-DQN is likely to explore shorter paths.

4. Robot Design

4.1. General Structure of the Robot. Maneuverability is an important feature for mobile robots [50]. Omni-wheeled robots have high maneuverability due to their ability to move directly from one position to another without being redirected [51].

As shown in Figure 10, a mobile robot with high maneuverability was designed to use the agent trained in the virtual environment in the real environment application. The robot consists of five layers. The 1st and 2nd layers are the visual layers and are used for the determination of the robot's position and rotation. Their intended purposes will be explained in the next sections.

The 3rd layer has two serial connected Li-ion batteries with a current capacity of 2500 mAH and a voltage value of 3.7 V, which provide for the energy needs of the robot.

In the 4th layer, there is the robot control circuit that executes commands from the computer via the RF controller, as shown in Figure 11.

RF control circuit and robot control circuit are microcontroller-based circuits. Both circuits use Atmel's ATmega328P microcontroller. The robot control circuit is designed so that it can control DC motors with encoder, control the PWM speed, and give audible and visual warnings. For the communication between the controller and the robot, the SX1278 transceiver module that operates in the 433 MHz frequency band and communicates with Long Range (LoRa) modulation is used.

The 5th layer is the layer containing four DC motors with N20 encoder and four 8-cylinder omni wheels with a 50 mm diameter, which are controlled by these motors.

4.2. Movements and Calibration of Mobile Robot. In this study, the agent was trained in the empty virtual environment. There are three movements the agent can make for this environment (go forward 1 unit, 90° turn right, 90° turn left). Empty environment is a discrete environment.

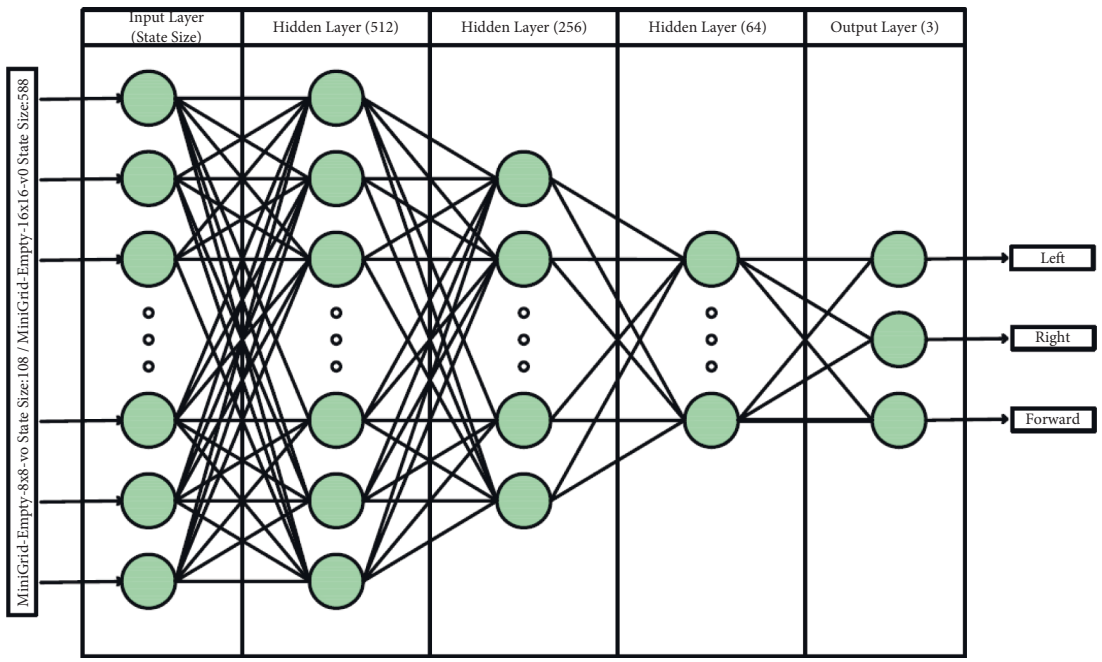

Figure 5: Q-network architecture.

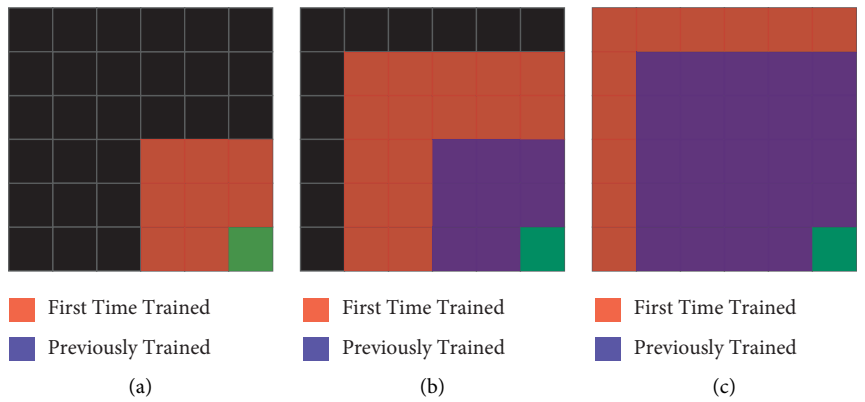

Figure 6: Application of HSP-DQN algorithm to minigrid-empty-8 × 8-v0 environment. (a) Episodes 0–500. (b) Episodes 500–1000. (c) Episodes 1000–3000.

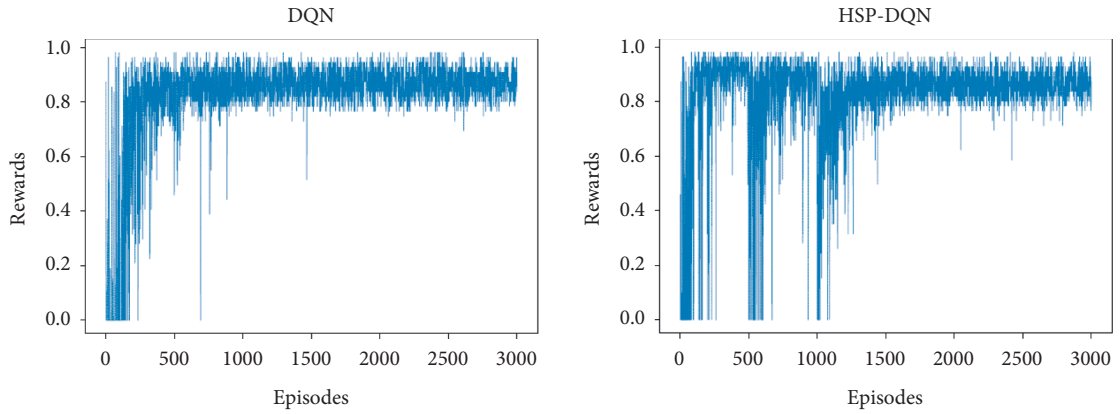

Figure 7: DQN and HSP-DQN reward graphs (minigrid-empty-8 × 8-v0).

Table 1: Comparison of DQN and HSP-DQN (MiniGrid-Empty-8 × 8-v0).

Minigrid-Empty-8 × 8-v0	DQN	Episodes	1–500	500–1000	1000–2000	2000–3000
		Epsilon-greedy	$1 > \varepsilon > 0.779$	$0.779 > \varepsilon > 0.606$	$0.606 > \varepsilon > 0.368$	$0.368 > \varepsilon > 0.223$
		Total number of steps	10646	4127	7373	7305
		Number of successful episodes	387	499	1000	1000
		Success rate (%)	77.40	99.80	100	100
		Total number of steps			29451	
		Average number of steps per episode			9.82	
	HSP-DQN	Episodes	1–500	500–1000	1000–2000	2000–3000
		Epsilon-greedy	$1 > \varepsilon > 0.779$	$0.800 > \varepsilon > 0.623$	$0.800 > \varepsilon > 0.485$	$0.485 > \varepsilon > 0.294$
		Total number of steps	6083	5037	9847	7659
		Number of successful episodes	441	483	993	1000
		Success rate (%)	88.20	96.60	99.30	100
		Total number of steps			28626	
		Average number of steps per episode			9.54	

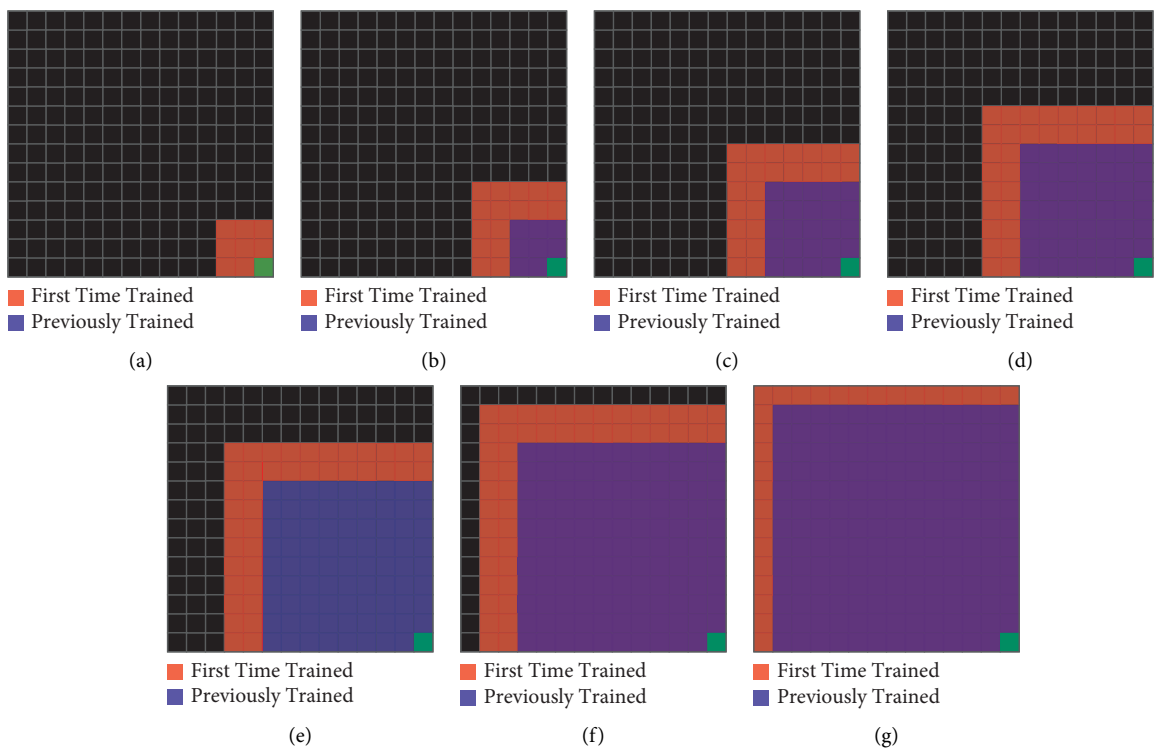

Figure 8: Application of HSP-DQN algorithm in the Minigrid-Empty-16 × 16-v0 environment. (a) Episodes: 0–500. (b) Episodes: 500–1000. (c) Episodes: 1000–2000. (d) Episodes: 2000–3000. (e) Episodes: 3000–4000. (f) Episodes: 4000–5000. (g) Episodes: 5000–8000.

Nevertheless, because the real environment is a continuous environment, the mobile robot can be in numerous different angular and positional positions. In order for the mobile robot to use the information it receives from the discrete environment, the real environment should be likened to a discrete environment. In other words, the mobile robot should be able to move in a way that can bring to a desired position. The mobile robot was designed to make 6 different movements, as shown in Figure 12, to go to the desired location. It can make turns with an accuracy of approximately 3° and linear movements with an accuracy of about 1 cm in the range of 0°–360°.

Although omni-wheeled robots have high maneuverability, the floor under them can negatively affect their movements [52]. Although the designed mobile robot has motors with encoder, encoder signals cannot guarantee the position of the robot. Due to the floor under the robot and the standing positions of the omni wheels, undesirable situations such as idle rotation of the wheels during take-off may be encountered. To reduce the mentioned problems relatively, the mobile robot calibration program shown in Figure 13 was prepared. By using this program, the number of signals from the encoder for 1 unit step of the robot (determined as 30 cm for this study), and the number of

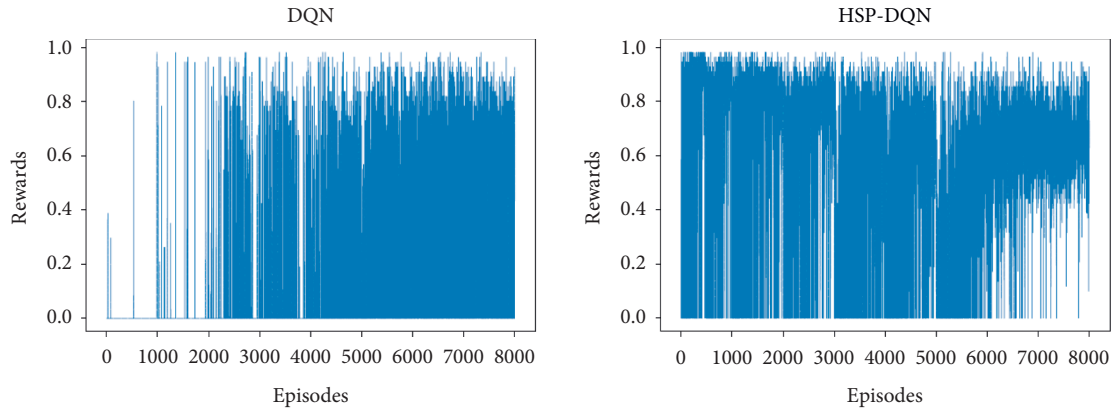

Figure 9: DQN and HSP-DQN reward graphs (Minigrid-Empty-16×16-v0).

signals from the encoder for 90^o rotation can be found. Multiple movements can be sent and it can be waited for the specified time or not waited in transition between movements depending on the selected mode (discrete, continuous). Angular and linear movements at different values can be determined and the speed of motors can be controlled with PWM.

To control the abovementioned features of the mobile robot, a data package is designed as in Figure 14. This data packet is prepared according to the action to be taken by the robot and sent with RF signals, and the robot can perform the necessary actions by decoding these signals.

Table 3 shows the experimentally obtained encoder signal values to be used in the study.

For example, the data packet required for the mobile robot to move 1 unit forward and turn 90° left is shown in Figure 15.

5. Real Environment Application with HSP-DQN

This section shows that the HSP-DQN network trained in a discrete empty environment can also be used in a real environment. Here, the HSP-DQN network was trained in a discrete environment and its application was performed by likening the real environment of the application to a discrete structure.

Figure 16 shows the real environment, which will be likened to a discrete structure. The environment data are obtained by the IP camera that takes the top view of the environment. However, the mobile robot used in the real environment receives the action commands from the computer via RF signals.

Simulation of real environment to discrete structure:

(i) Detection of mobile robot and target with convolutional neural networks,

(ii) Plotting the real environment as in the virtual environment,

(iii) Correction of the position and angle of the mobile robot to fit in the discrete environment,

(iv) The real environment is matched with the virtual environment in stages.

At the end of these stages, the mobile robot is able to use the HSP-DQN network trained in the virtual environment. Explanations about the stages are given in this section.

5.1. Detection of Objects with Convolutional Neural Networks. Convolutional Neural Networks (CNN) form the basis of image classification by DL [53]. The effectiveness of CNNs was proven in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition held in 2012 [54]. After this success, studies with CNN gained acceleration [55–59].

Today, CNN models developed for object detection have reached an enormous position due to their success in classification. However, model sizes are also increasing. This limits its use in real-world applications where latency and resource usage are important, such as autonomous robots. Therefore, model efficiency is becoming increasingly important in real-life applications [60].

In our study, EfficientDet D0 CNN model was used for object detection. This model, which uses the EffiCientNet model as the backbone network, has a good performance in image classification by scaling the network width, depth, and input resolution together. As the number of parameters of the model is optimized, it is also advantageous in terms of using system resources [60–62].

The CNN model was trained to detect the agent and target objects shown in Figure 17. 117 and 32 images were used for training and testing, respectively, in different resolutions and light conditions.

The total-loss graph of the model trained on 30,000 epochs is shown in Figure 18. The graphs show that the model has been trained to a large extent after 6,000 epochs.

Using the CNN model, the presence of the Agent and target in the environment, the position and the center information of the objects in the image are determined. In addition, the pixel distance of the environment can be converted by taking the agent image as a reference.

5.2. Plotting of Real Environment. The environment where the HSP-DQN network is trained is a discrete environment. However, in order for the mobile robot to use the trained HSP-DQN network, the real environment should be likened

Table 2: Comparison of DQN and HSP-DQN (MiniGrid-Empty-16×16-v0).

MiniGrid-Empty-16×16-v0	DQN	Episodes	1–500	500–1000	1000–2000	2000–3000	3000–4000	4000–5000	5000–6000	6000–7000	7000–8000
		Epsilon-greedy	$1 > \varepsilon > 0.779$	$0.779 > \varepsilon > 0.606$	$0.606 > \varepsilon > 0.368$	$0.368 > \varepsilon > 0.223$	$0.223 > \varepsilon > 0.135$	$0.135 > \varepsilon > 0.082$	$0.082 > \varepsilon > 0.050$	$0.050 > \varepsilon > 0.030$	$0.030 > \varepsilon > 0.018$
		Total number of steps	24973	24912	49430	46311	45486	42255	37297	32136	28881
		Number of successful episodes	2	2	16	119	144	261	424	569	708
		Success rate (%)	0.40	0.40	1.60	11.90	14.40	26.10	42.40	56.90	70.80
	Total number of steps				331681						
	Average number of steps per episode				41.46						
	HSP-DQN	Episodes	1–500	500–1000	1000–2000	2000–3000	3000–4000	4000–5000	5000–6000	6000–7000	7000–8000
		Epsilon-greedy	$1 > \varepsilon > 0.779$	$0.800 > \varepsilon > 0.623$	$0{,}800 > \varepsilon > 0{,}485$	$0{,}800 > \varepsilon > 0{,}485$	$0{,}800 > \varepsilon > 0{,}485$	$0{,}800 > \varepsilon > 0{,}485$	$0{,}800 > \varepsilon > 0{,}485$	$0.485 > \varepsilon > 0.294$	$0.294 > \varepsilon > 0.178$
		Total number of steps	8806	7157	17491	19176	26923	26424	34068	20562	19607
		Number of successful episodes	388	442	835	842	709	766	577	970	994
		Success rate (%)	77.60	88.40	83.50	84.20	70.90	76.60	57.70	97.00	99.40
	Total number of steps				180214						
	Average number of steps per episode				22.53						

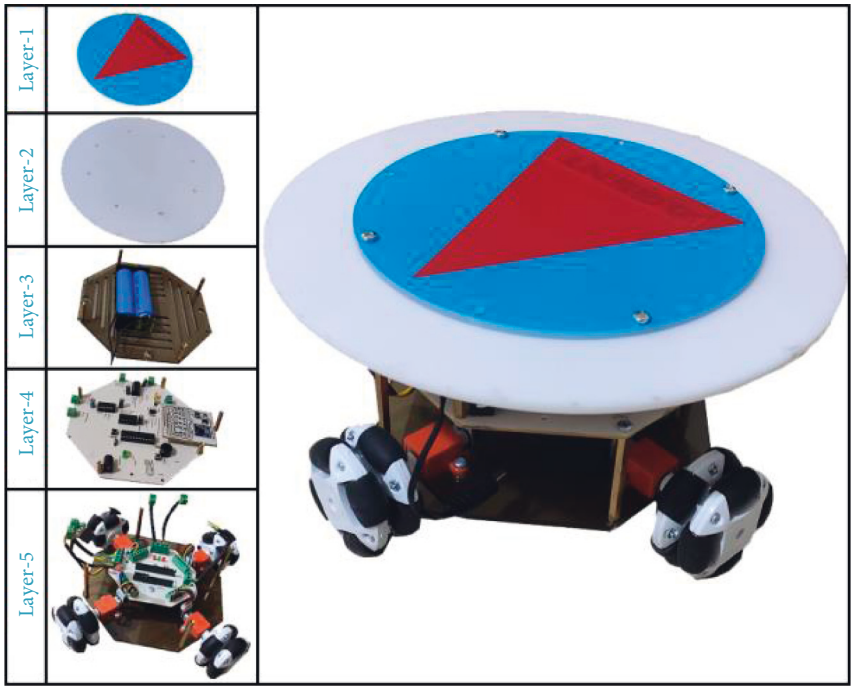

Figure 10: Omni-wheeled mobile robot.

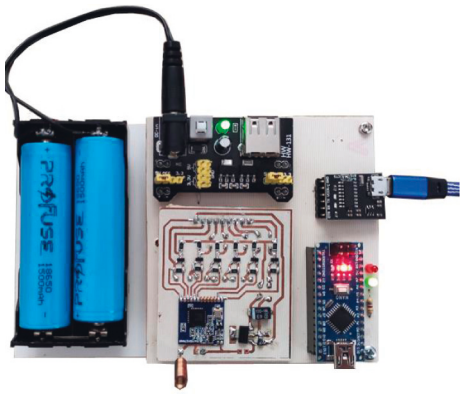

Figure 11: RF controller.

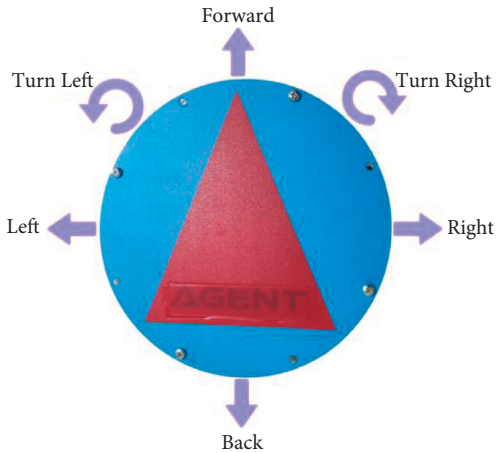

Figure 12: Movements of mobile robot.

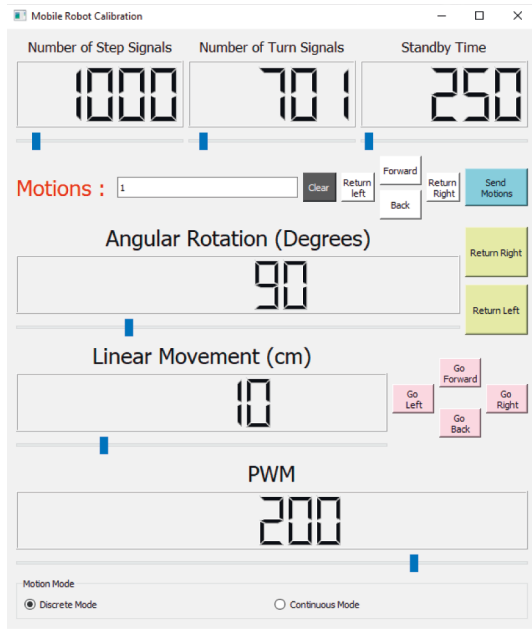

Figure 13: Mobile robot calibration program.

to the discrete environment. The steps of the simulation process are described below;

Stage-1: Real environment data are obtained using only the top view of the environment. Using the CNN from this image, the positions of the agent and the target are determined as in Figure 19.

Stage-2: Centers of the target $(Goal_x, Goal_y)$ and the agent $(\text{Agent}_x, \text{Agent}_y)$ are calculated for use in the

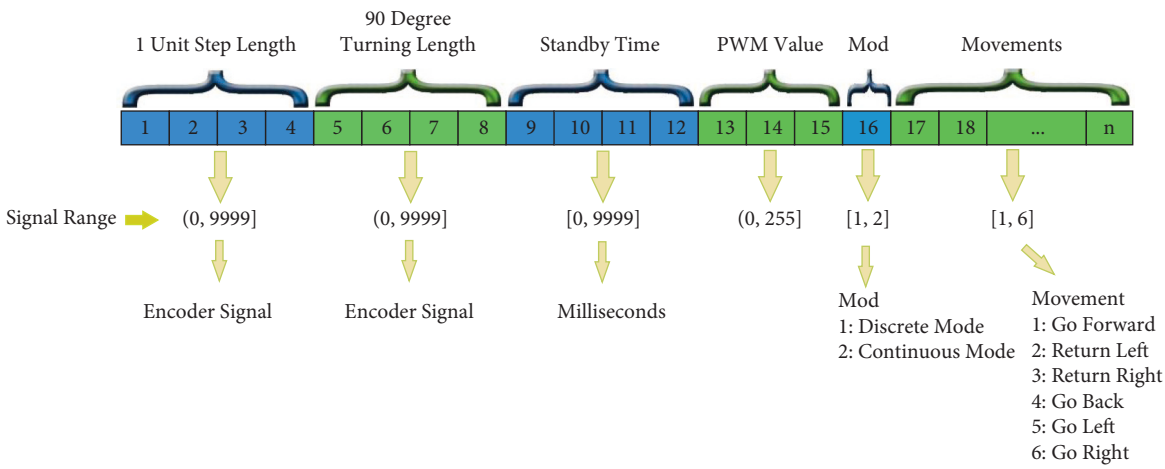

Figure 14: RF data packet.

Table 3: Experimentally obtained encoder signal values.

Action	Encoder signal
1 cm linear motion	22
30 cm linear motion (one unit)	2070
3° turning motion	10
90° turning movement	900

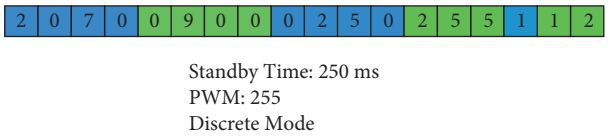

Figure 15: Sample data package.

next stages. The area of each plot in the real environment is determined as $30 \times 30\ \text{cm}^2$. In other words, the mobile robot should travel 30 cm to pass from the center of one plot to the center of the other plot. The step distance set as 30 cm (Step_d) is calculated by taking the agent image as a reference. The diameter of the circular agent image is 16 cm and the length of one side (d) of the frame that detects the agent is approximately equal to the diameter of the agent image. So 1 cm is approximately equal to $d/16$ pixels. The step distance of the mobile robot in pixels is shown in the following equation:

$$\text{Step}_d = 30\,\text{cm} \longrightarrow \text{Step}_d = \frac{30}{16} * \text{dpixel}. \tag{2}$$

Stage-3: The centers of the plots in the real environment including width (Image_w) and length (Image_h) of the environment image are determined using equation (3). As the location of the agent is fixed when the real environment is plotted, a list of the plot centers ($G_{centers}$) is created by taking the center of the target as a reference, and the real environment is plotted.

$$G_{centers} = \left\{ a, b \mid a, b \in \text{N}, \left(0 < \text{Goal}_x \pm \text{a} * \text{Step}_d < \text{Image}_w, 0 < \text{Goal}_y \pm \text{b} * \text{Step}_d < \text{Image}_h\right). \right\} \tag{3}$$

Stage-4: The location of the agent may not match the center of any plot because the center of the target is used as the reference in the plotting process. For this reason, the mobile robot should be positioned according to the nearest plot. Using equation (4), the distances of the mobile robot to the plot centers ((Center_d)) are listed.

$$\text{Center}_d = \left\{ (k, l) \in \mid G_{centers} \sqrt{\left(\text{Agent}_x - x_k\right)^2 - \left(\text{Agent}_y - y_l\right)^2}, \right\} \tag{4}$$

The element of the $G_{centers}$ list Center_d as the minimum value in the list shows the plot center closest to the mobile robot. This plot also shows the initial position of the mobile robot.

The plotted real environment is shown in Figure 20. After the real environment is plotted, the location where the center of the robot should be at the beginning is determined.

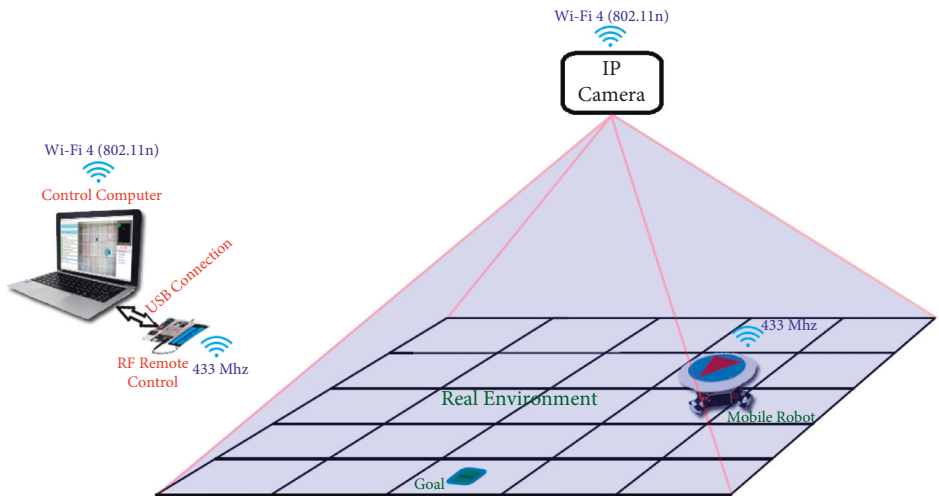

Figure 16: Real environment structure.

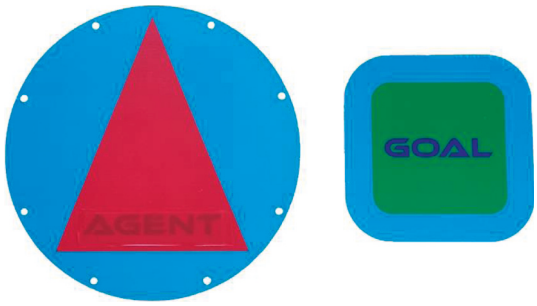

Figure 17: Agent and target objects.

5.3. Determination of the Rotation of the Mobile Robot. CNNs have become the main method in the field of image recognition thanks to their strong feature extraction ability. Most CNNs do not deal with the angle of the image and just focus on image recognition. However, angle data are also crucial in robot applications [63].

In the present study, the agent was trained with HSP-DQN in a discrete virtual environment. Only four directions exist for the virtual environment. However, in the real environment, the mobile robot can take an unlimited angular position. In order for the mobile robot to use the information in the virtual environment, it should adjust its direction to resemble the most suitable direction in the virtual environment. For this reason, a structure that can detect the angle of the mobile robot is required. The image used by CNN for the detection of the mobile robot is also used in the algorithm created for the detection of the mobile robot's angle. The angle and direction of the isosceles triangle on the mobile robot are also the same as the angle and direction of the mobile robot. The purpose of the algorithm is to determine the direction and angle of the mobile robot by determining the direction and angle of the triangle in the image.

Figure 21 shows the stages of the algorithm.

Stage-1: A top image of the mobile robot and the environment where the target is located is taken.

Stage-2: The coordinates of the agent are determined by CNN and this part is taken from the image. Just below the agent image, a white layer is used, which visually covers the part under the robot. Thus, while the algorithm is running, it is not affected by the colors on the ground and the mobile robot can use the algorithm on any color ground.

Stage-3: Bilateral Filter is applied to reduce noise and sharpen edges in the image [64].

Stage-4: As the mobile robot image is in the shape of a circle, the frame determined by CNN is roughly like a square, the center of the frame always remains inside the triangle shape. *R*, *G*, and *B* values are listed separately by retrieving the RGB values of the pixel in the center of the image and its eight neighbors. The values in each list are ordered in ascending order. The reference RGB values of the triangle image are obtained by retrieving the 5th elements of the lists created for *R*, *G*, and *B*. This process aims to establish a structure that can work in different light environments. A two-color image is obtained by assigning red color value to pixels that are close to the reference RGB values and white color value to other pixels.

Stage-5: The pixels where the red color is first detected are determined by scanning the image separately from the right, left, bottom, and top. Thus, four points are obtained.

Stage-6: The closest two points within the four detected points show the same corner. Only one of the two points can be used. To make a selection, the distances of two points to other points are calculated separately and the point that is farther from other points is selected as the 3rd corner point because being further away implies that the point is further away.

Stage-7: The distances between the corner points are calculated. As the twin sides of the triangle are longer than the base side, the side with the two closest vertices shows the base of the triangle and the other vertex shows the vertex of the triangle. By drawing a line to the midpoint of the vertex and the base, a Pythagorean triangle is formed as in Figure 21 Stage-7, and the angle

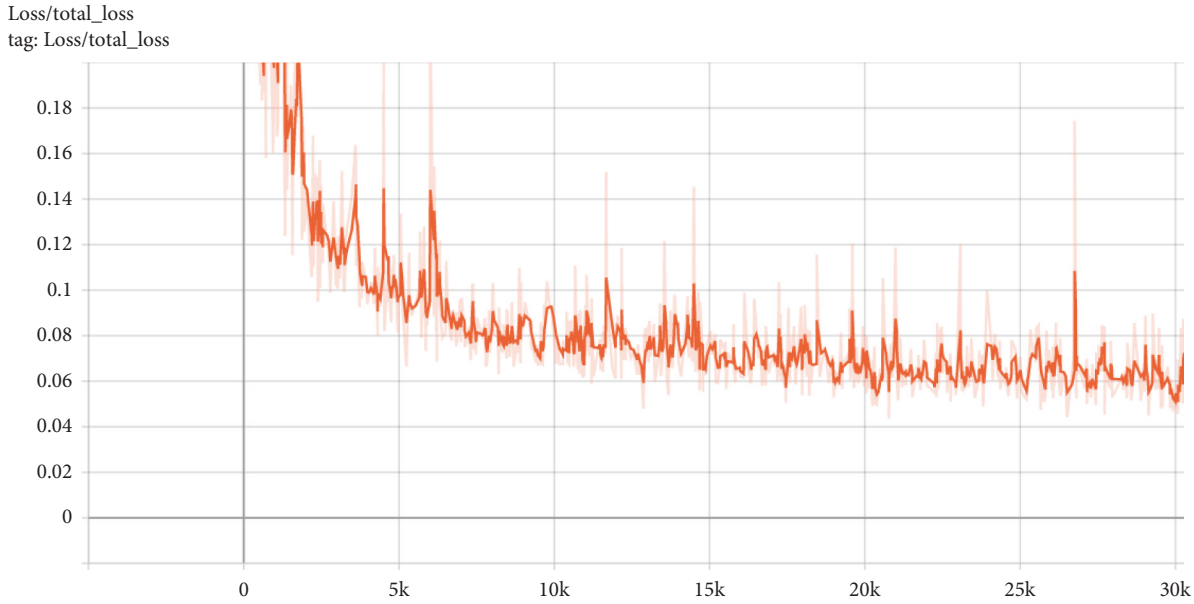

Figure 18: Total-loss graph.

Figure 19: Detection of agent and target with CNN.

of the triangle, hence the angle of the mobile robot, is determined by Arctangent.

If the angle found is 0°, the position of the mobile robot in real environment and virtual environments can be matched. If the angle found is different from 0°, the direction of the mobile robot can be in one of the four regions as shown in Figure 22. The triangle direction determines which region the mobile robot is in. The direction to which the mobile robot will turn is determined by comparing which direction it is angularly close to.

Using the algorithm, it is determined in which region the mobile robot is, how much the robot should turn angularly to the right or left to match the most suitable direction in the virtual environment, and in which position the mobile robot will be after the rotation is corrected.

The mobile robot autonomously adjusts its angle using the values determined in this episode and then positions it to the initial position determined in the previous episode with linear movements (right, left, forward, backward). After the mobile robot corrects its angle and position, both the mobile robot and the target are located in the center of the plot they are in, as shown in Figure 23. The mobile robot is also positioned in one of the four directions it can take, as in the virtual environment.

As the mobile robot and the target are located only in the centers of the plots and there are four directions that the mobile robot can take, as shown in Figure 24, the real environment and the discrete virtual environment can be matched.

5.4. Matching the Real Environment with the Virtual Environment and Determining the Path Plan. To use the HSP-DQN network trained in the virtual environment in the real environment, it is necessary to match the position of the mobile robot with the position of the virtual agent, and the position of the real target with the position of the virtual target. For the virtual environment, the agent can be in any position and direction at the beginning, and the target is located in the lower right corner of the virtual environment as in Figure 25.

However, in the real environment, the mobile robot can be in any direction and position around the target. In other words, many situations exist where the real and virtual environments cannot match. To solve this problem, the real environment is divided into four regions, as in Figure 26, according to the position of the mobile robot to the target. The virtual environment is rotated so that it can match the real environment.

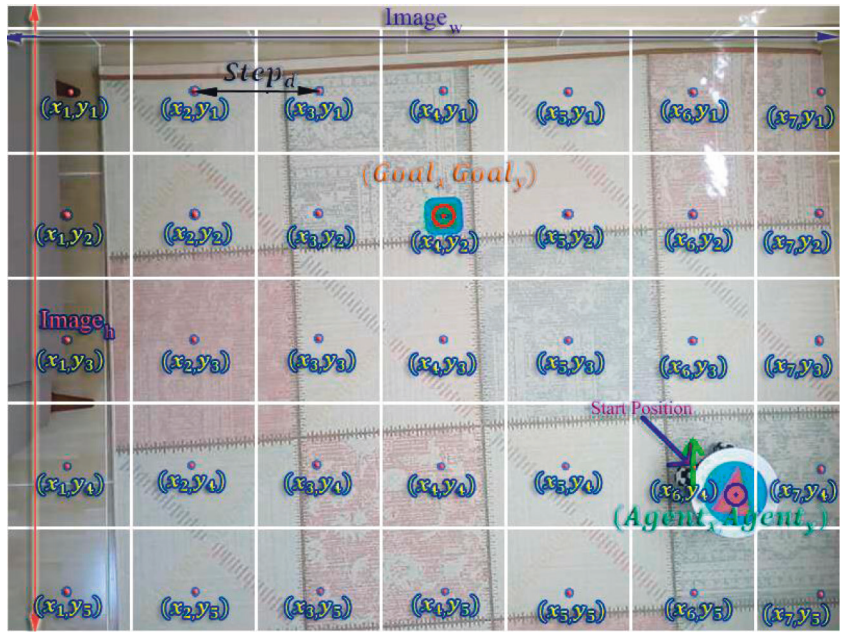

Figure 20: Plotting of the real environment.

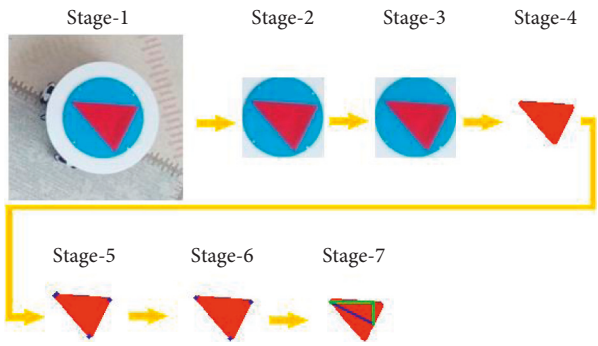

Figure 21: Stages of angle determination algorithm.

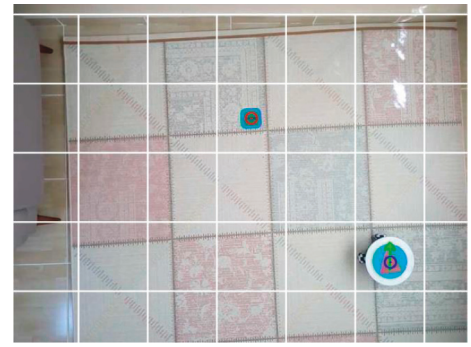

Figure 23: Positioning the mobile robot to the initial position.

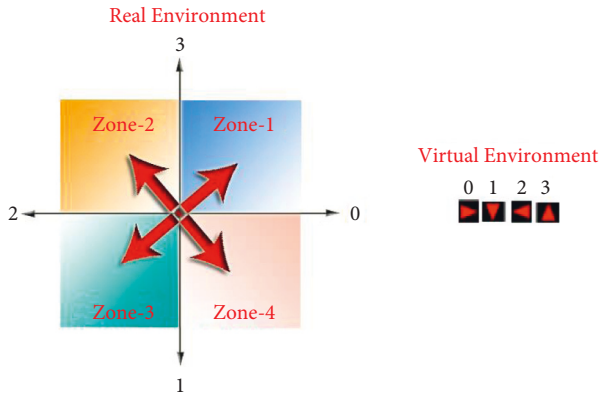

Figure 22: Directions in real environment vs. virtual environment.

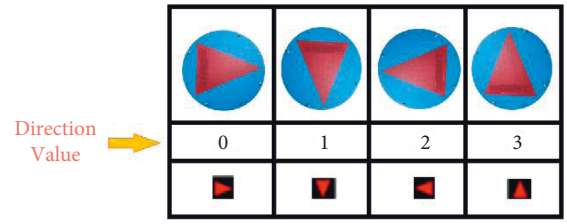

Figure 24: Directions for mobile robot and virtual agent.

To match the rotated virtual environment with the real environment, the initial direction and the position of the virtual agent are converted according to the region as in Figure 27.

As a result of these conversions, all states in the real environment can be matched with the virtual environment and the trained HSP-DQN network can be used for the real environment, as in Figure 28.

The path plan is determined in the virtual environment before the mobile robot takes action. With the trained HSP-DQN, an action is generated for the current state of the agent and applied in the virtual environment, enabling the agent to move to its new position. The HSP-DQN continues to generate action until the agent reaches its target, and these

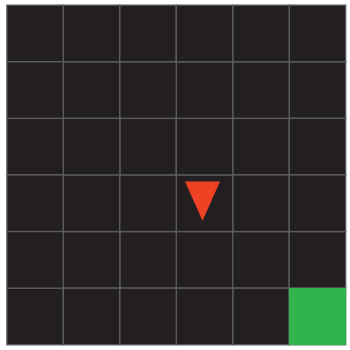

Figure 25: Virtual environment.

actions are listed for the mobile robot to use. In this way, the path plan of the mobile robot is obtained.

After the path plan is drawn, the actions to be taken are sent to the mobile robot in order. After each action, the angle and the position of the mobile robot are checked and deviations are corrected. Whether the mobile robot completes its current action or not is determined by the background difference method. The mobile robot is ensured to reach the target by sending the next action it needs to do after each completed action.

6. Discussion and Future Work

In the previous sections of the paper, the proposed algorithms have been discussed. In this section, the performance of the system will be examined. Although DQN algorithm is successful in MiniGrid-Empty-8x8-v0 environment, it cannot achieve similar success in MiniGrid-Empty-16 × 16-v0 environment because the agent can only receive rewards when it arrives at the destination in MiniGrid-Empty environments. If the environment has a large area such as MiniGrid-Empty-16 × 16-v0 and the rewards are sparse, it is difficult for the DQN algorithm to generate enough meaningful data to train the network with random movements, and the convergence speed becomes very slow. In the proposed method, the initial position of the agent is started in the region where the probability of reaching the reward is high and this increases the probability of the network to reach meaningful data. Then, as the network learns the paths in the environment the area where the agent can start is gradually expanded. Thus, for large areas with sparse rewards the convergence problem of the network is addressed. Also, performance analyses are performed for MiniGrid-Empty-16 × 16, which is a larger environment than MiniGrid-Empty-8x8.

In the experimental study, a computer with Windows 10 Pro 64 Bit operating system was used. The hardware specifications are provided in Table 4.

Neural networks used in DQN and HSP-DQN algorithms have the same features. The information about the structure of the neural network is shown in Table 5 and the hyper parameters of the algorithms are shown in Table 6.

In the DQN algorithm, the number of different states that the agent can start is 780. While the neural network is being trained, one of the 780 states is randomly selected as the initial state in each new section. In the proposed algorithm, as the sections progress, the area where the training can begin is gradually enlarged. Thus, the number of different states that the agent can initiate increases, as shown in Table 7.

In both algorithms, the performances of the models were examined after training 8000 sections under the above-mentioned conditions. The time spent by the algorithms for training the neural network is shown in Table 8. The training of the proposed algorithm was completed in 45.78% less time than the DQN algorithm. Although the agent moved more in the environment in the DQN algorithm, the time spent in the sections increased as it reached fewer targets compared to the proposed algorithm.

In order to compare the success of the trained models, all conditions were tested in the environment with 780 different initial conditions. The related results are presented in Table 9. From the table it can be seen that success rate of the proposed algorithm is 41.28% higher than the DQN algorithm. Success of the algorithms is also graphically shown in Figure 29(a).

Moreover, the lengths of the paths detected by the algorithms were also examined. The models trained with DQN and the proposed algorithm were started with the same initial conditions and the obtained path lengths were compared as shown in Figure 29(b). The proposed algorithm finds a shorter path compared to the DQN algorithm in 70.1% of different initial conditions.

As a result of the tests, it was seen that the success rate and convergence speed of the proposed algorithm are better than the DQN algorithm for large environments with sparse rewards. In addition, the effects of some DRL-based algorithms developed with our study on success rates are shown in Table 10.

In the real environment application of the study, only the top view of the environment was used as the system input. Since it was desired that the application run in real time, this image should be used efficiently. The object detection capability, speed, and model size of the model used for object detection were very important. Therefore, the EfficientDet-D0 model was used for object detection by considering these criteria. The performance analyses of the EfficientDet model presented in the original article are shown in Figure 30. The model is considered to be suitable for mobile robot applications with limited resources in terms of model size, speed, and object detection.

Performance of the EfficientDet model in our application was also tested. The model was trained as specified in Section 5.1. The trained model detected objects with the image taken from the IP camera mounted at a height of 180 cm from the ground. The captured image had a resolution of 800 ∗ 600 pixels and covered an area of $2.1 \times 1.6\ m^2$. In the performed tests, the model could successfully detect mobile robot and target objects, provided that the ambient lighting is appropriate. Also, it was determined that the object recognition process was completed in the time interval of 820–920 ms.

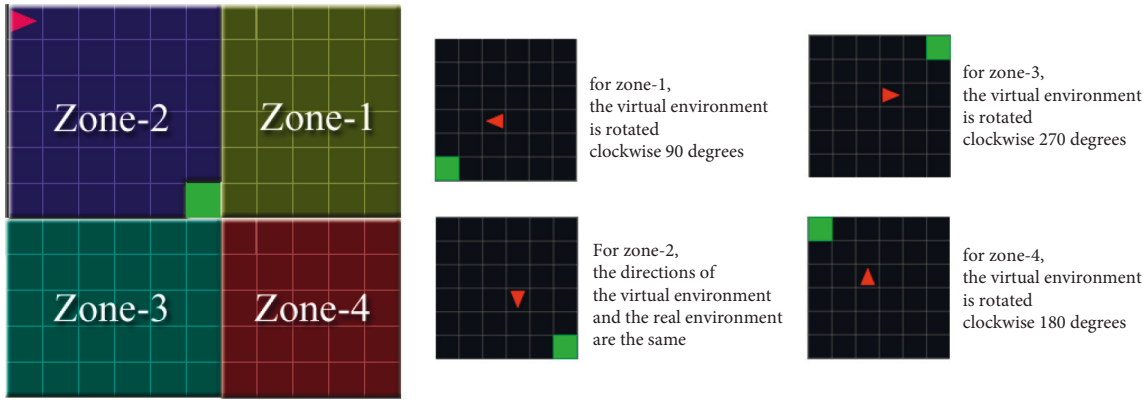

Figure 26: Rotation of the virtual environment to obtain regions in the real environment.

matrix index
(x_1,y_1): Location of mobile robot
(x_2,y_2): Location of real goal

ZONES	Real Directions		Virtual Directions	Real Start Position	Virtual Start Position
ZONE-1 → $x_1 - x_2 < 0$ and $y_1 - y_2 \geq 0$	0	→	3	(x,y) → (6+x,6-y)	
	1	→	0		
	2	→	1		
	3	→	2		
ZONE-2 → $x_1 - x_2 \geq 0$ and $y_1 - y_2 \geq 0$	0	→	0	(x,y) → (6-x,6-y)	
	1	→	1		
	2	→	2		
	3	→	3		
ZONE-3 → $x_1 - x_2 \geq 0$ and $y_1 - y_2 < 0$	0	→	1	(x,y) → (6-x,6+y)	
	1	→	2		
	2	→	3		
	3	→	0		
ZONE-4 → $x_1 - x_2 < 0$ and $y_1 - y_2 < 0$	0	→	2	(x,y) → (6+x,6+y)	
	1	→	3		
	2	→	0		
	3	→	1		

Figure 27: Conversion of virtual environment according to real environment.

As mentioned earlier, there are 780 different initial conditions in the MiniGrid-Empty-16 × 16 virtual environment. With the employment of the structure described in Section 5.4, the target is ensured to be in any position in the environment, rather than just in a certain position. In other words, the target can be positioned in 196 different positions at the beginning. Thus, the success of the trained network is ensured for $780 \times 196 = 152.880$ different initial conditions. In here, the recommended regional matching process takes less than 1 ms to complete.

The real environment processes are performed in different steps. Table 11 provides the processing times for those steps. The designed system is capable of detecting the state of the environment within 1–1.5 seconds in order to allow the robot to perform a specific task.

A central computer is used for the designed system in order to perform the semantic data extraction, determination of the mobile robot motion, and notification of the determined motion. The mobile robot acts as an actuator by using the commands from the central computer. As the mobile robot does not require any computational load, a low-cost robot was designed with the part listed in Table 12.

LoRa-based modulation is used for the wireless communication between the computer and the mobile robot. The LoRa has advantages such as wide coverage, low power consumption, low cost, and no license requirement [72]. However, it has a data transfer rate of 290 kbps which is slower than many wireless communication methods [73]. Despite the latter disadvantage, LoRa still is an ideal communication method for the application reported in this work since only small-sized data signals are transmitted for the movement of the robot.

In the real environment experiments, the DRL-based mobile robot was launched with different initial conditions for the four regions that are specified in Section 5.4. As can be observed from the following link, the mobile robot was

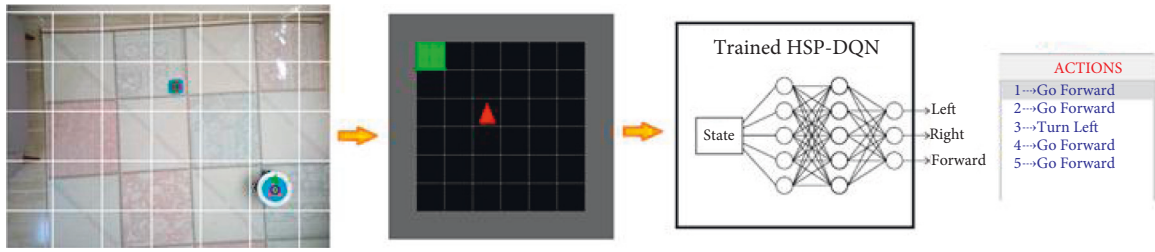

Figure 28: Use of HSP-DQN in real environment.

Table 4: Specifications of the computer used in the experimental study.

Hardware	Specifications
CPU	Intel™ Core™ i7-9750H CPU @ 2.60 GHz 2.60 GHz
GPU	NVIDIA GeForce RTX 2070 8 GB
RAM	16 GB 2667 MHz

Table 5: Features of the neural network used in DQN and HSP-DQN.

Layers	Number of neurons	Activation function	Parameters
Input layer	512	ReLU	301568
Hidden Layer-1	256	ReLU	131328
Hidden Layer-2	64	ReLU	16448
Output layer	3	Linear	195
Total parameters			449.539
Optimizer: RMSprop algorithm			
Loss: Mean squared error			
Learning rate: 0.00025			
Rho: 0.95			
Epsilon: 0.01			

Table 6: Hyper parameters of DQN and HSP-DQN algorithms.

Hyper parameters	Value
Total episodes	8000
Max step	50
Memory	2000
Epsilon greedy	1.0
Minimum epsilon greedy	0,001
Epsilon greedy decay	0,9995
Mini batch	64

able to successfully reach the target in all experiments (Please see: https://youtu.be/bEAmJF6lD4g). The following were also observed in the experiments:

(i) The omni-wheeled structure of the mobile robot has advantages in terms of maneuvering. However, it is prone to disturbances from the floor on which the robot is operating. Although this situation does not prevent the robot from reaching its target, it negatively affects the time of the robot to reach the target.

Table 7: Gradual enlargement of the training area.

Education levels	Episodes	Area	Number of different states the agent can start
1st level	1–500	3 × 3	32
2nd level	501–1000	5 × 5	96
3rd level	1001–2000	7 × 7	192
4th level	2001–3000	9 × 9	320
5th level	3001–4000	11 × 11	480
6th level	4001–5000	13 × 13	672
7th level	5001–8000	14 × 14	780

Table 8: Training times of DQN and HSP-DQN.

Algorithm	Episodes	Total step	Training time (hours)
DQN	8000	331681	8,3 hours
HSP-DQN	8000	180214	4,5 hours

Table 9: Success rates of DQN and HSP-DQN algorithms.

Algorithm	Succeeded	Failed	Success rate (%)
DQN	442	338	56,67
HSP-DQN	764	16	97,95

(ii) Thanks to the adaptive structure of the system, the working area of the system can be changed by changing the height of the camera from the ground. However, excessive magnification of the area covered by the camera or improper lighting of the environment may adversely affect the detection of objects in the environment.

This work forms a good basis for the future works as it demonstrates the possibility of controlling many robots within a long range by using a single center. The designed robot has the potential to be used in swarm robotics and multi-agent systems, as it is cost-effective and long-range controllable; thus, this work presents a good contribution to one of the important research topics known as multi-robot applications [74–76]. The latter is important for accelerating global policy education, as the use of large numbers of robots can provide greater data diversity [77]. In the future, real-world applications of DRL algorithms, which will include

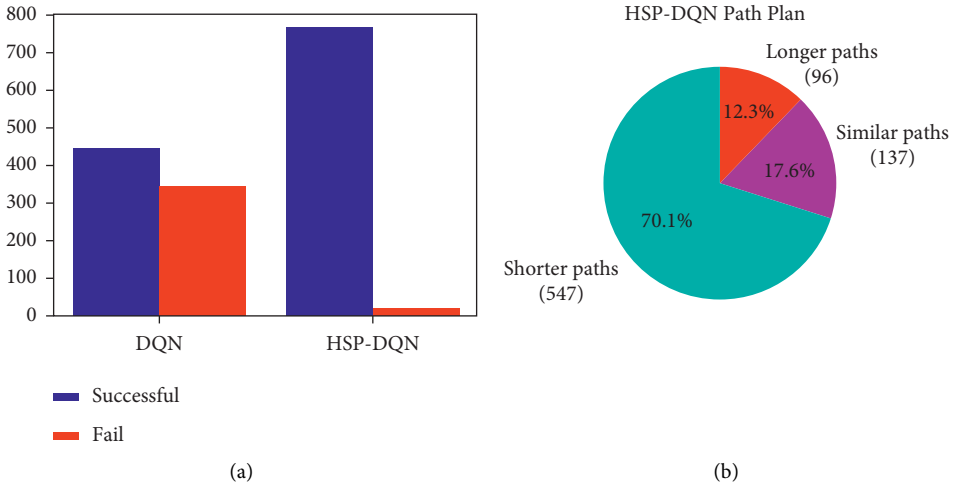

Figure 29: Virtual environment (minigrid-empty-16×16-v0) performances of HSP-DQN, and DQN.

Table 10: The effect of improved DRL-based algorithms on the success rate.

Reference	Authors	Compared algorithm (success rate %)	Proposed algorithm (success rate %)	Simulation	Real system
[65]	Pfeiffer et al.	IL or CPO (37)	IL + CPO (90)	Y	Y
[66]	Hsu et al.	A3C (17.5)	LSTM + DRL (49.5)	Y	Y
[67]	Long et al.	NH-ORCA (78)	Parallel PPO (96.5)	Y	N
[68]	Wang et al.	DDPG (42.67)	Fast-RDPG (97.42)	Y	N
[69]	Lin et al.	PPO (23)	Improved PPO (88)	Y	Y
[70]	Wang et al.	POfD (58.5)	LwH (96.5)	Y	N
[71]	Leiva and Ruiz-del-Solar	DDPG (69)	PCL-LSTM (88)	Y	Y
[29]	Wang et al.	DQN (63.4)	DDQN with PER (81.1)	Y	N
[32]	Kim et al.	DQN (46)	DQN + GRU + action skipping (87)	Y	N
[30]	Zhang et al.	DQN (75)	Improved DQN (90)	Y	N
[7]	Zhou et al.	DDPG (84.25)	LDDPG + D (94.25)	Y	N
[5]	Gong et al.	DDPG (83.5)	MN-LSTM-DDPG (90.5)	Y	N
[26]	Xing et al.	DQN (75)	Area division DQN (100)	Y	Y
Our study	Erkan and Arserim	DQN (56.67)	HSP-DQN (97.95)	Y	Y

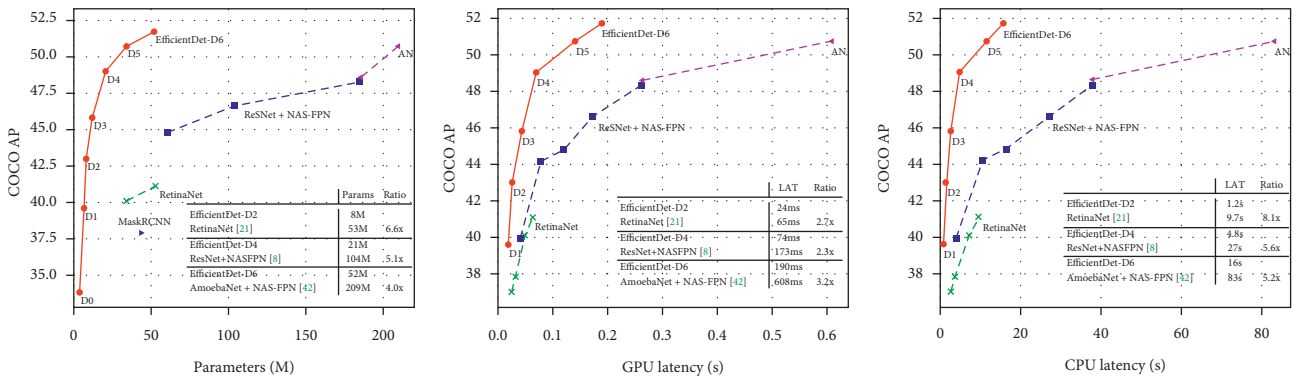

Figure 30: Performance of the efficientdet model on the COCO dataset [60]. (a) Model size. (b) GPU latency. (c) CPU latency.

Table 11: Time of each operation.

Operation	Min. tim (ms)	Max. time (ms)
Object detection	820	920
Parceling the environment	15	16
Ro determining the angle of the robot	40	60
Area determination	0	1
Path planning	100	300
Sending the motion signal from the computer to the controller and from the controller to the robot	100	150
Decoding incoming motion signal by robot	10	20
	1085 ms < total time < 1467 ms	

Table 12: Cost of the mobile robot.

Parts	Cost ($)
Robot body + omni wheels (4 pieces)	20
N20 micro encoder gear motor 6V 105 rpm (4 pieces)	24
Electronic components	28
Total	72

sequential tasks to enable the agent to reach its target optimally in environments with static and dynamic obstacles or control more than one mobile robot, can be performed as a future work.

7. Conclusion

In this study, the HSP-DQN algorithm, which had a higher convergence rate and success rate than the classical DQN algorithm, was proposed for the environments with sparse rewards. With the proposed algorithm, a real environment application was performed for the first time by using the network parameters of an agent trained in the minimalistic grid world virtual environment. A low-cost mobile robot that can be controlled within a long range by a central computer was designed for this purpose. Also, a system which can efficiently match the virtual environment with a discrete structure and the real environment was designed.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Zheng, S. Mao, Z. Wu, P. Kong, and H. Qiang, "Improved path planning for indoor patrol robot based on deep reinforcement learning," *Symmetry*, vol. 14, no. 1, p. 132, 2022.

[2] P. K. R. Maddikunta, Q. V. Pham, P. B et al., "Industry 5.0: a survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, Article ID 100257, 2022.

[3] W. Zehra, A. R. Javed, Z. Jalil, H. U. Khan, and T. R. Gadekallu, "Cross corpus multi-lingual speech emotion recognition using ensemble learning," *Complex Intell. Syst.*vol. 74, no. 4, pp. 1845–1854, 2021.

[4] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: a review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[5] H. Gong, P. Wang, C. Ni, and N. Cheng, "Efficient path planning for mobile robot based on deep deterministic policy gradient," *Sensors*, vol. 22, no. 9, p. 3579, 2022.

[6] L. Zhang, Y. Zhang, and Y. Li, "Path planning for indoor Mobile robot based on deep learning," *Optik*, vol. 219, Article ID 165096, 2020.

[7] Q. Zhou, L. Lyu, and H. Liu, "Deep reinforcement learning with long-time memory capability for robot mapless navigation," in *Proceedings of the 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 1215–1220, Hangzhou, China, May 2022.

[8] A. G. Sutton, S. Richard, and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Massachusetts, United States, 1998.

[9] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the International Conference on Machine Learning*, pp. 1329–1338, Jun. 2016.

[10] T. P. Lillicrap, J Jonathan, P. Alexander et al., "Continuous control with deep reinforcement learning," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–14, Sep. 2015.

[11] H. Li, R. Cai, N. Liu, X. Lin, and Y. Wang, "Deep reinforcement learning: algorithm, applications, and ultra-low-power implementation," *Nano Communication Networks*, vol. 16, pp. 81–90, 2018.

[12] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, Article ID 106685, 2021.

[13] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Article ID 5187540, 2015.

[14] V. Mnih, K. Koray, S. David et al., "Playing Atari with Deep Reinforcement Learning," 2013, https://arxiv.org/abs/1312.5602.

[15] D. Silver, H. Thomas, S. Julian et al., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," 2017, https://arxiv.org/abs/1712.01815.

[16] D. Silver, J. Schrittwieser, K. Simonyan et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[17] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," in *Proceedings of the British Machine Vision Conference 2017*, pp. 1–13, April 2017.

[18] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, "Intrinsically motivated reinforcement learning for human–robot interaction in the real-world," *Neural Networks*, vol. 107, pp. 23–33, 2018.

[19] T. Guo, N. Jiang, B. Li, X. Zhu, Y. Wang, and W. Du, "UAV navigation in high dynamic environments: a deep reinforcement learning approach," *Chinese Journal of Aeronautics*, vol. 34, no. 2, pp. 479–489, 2021.

[20] J. Woo, C. Yu, and N. Kim, "Deep reinforcement learning-based controller for path following of an unmanned surface vehicle," *Ocean Engineering*, vol. 183, pp. 155–166, 2019.

[21] J. Wang, S. Elfwing, and E. Uchibe, "Modular deep reinforcement learning from reward and punishment for robot navigation," *Neural Networks*, vol. 135, pp. 115–126, 2021.

[22] P. N. Srinivasu, A. K. Bhoi, R. H. Jhaveri, G. T. Reddy, and M. Bilal, "Probabilistic Deep Q Network for real-time path planning in censorious robotic procedures using force sensors," *Journal of Real-Time Image Processing*, vol. 18, no. 5, pp. 1773–1785, 2021.

[23] P. Li, X. Ding, H. Sun, S. Zhao, and R. Cajo, "Research on dynamic path planning of mobile robot based on improved DDPG algorithm," *Mobile Information Systems*, vol. 2021, pp. 1–10, 2021.

[24] J. C. Jesus, J. A. Bottega, M. A. S. L. Cuadros, and D. F. T. Gamarra, "Deep deterministic policy gradient for navigation of mobile robots in simulated environments," in *Proceedings of the 2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 362–367, Belo Horizonte, Brazil, Dec. 2019.

[25] W. Wang, Z. Wu, H. Luo, and B. Zhang, "Path planning method of mobile robot using improved deep reinforcement learning," *Journal of Electrical and Computer Engineering*, vol. 2022, Article ID 5433988, pp. 1–7, 2022.

[26] Y. Xing, R. Young, G. Nguyen et al., "Optimal path planning for wireless power transfer robot using area division deep reinforcement learning," *Wireless Power Transfer*, vol. 2022, pp. 1–10, Article ID 9921885, 2022.

[27] R. Huang, C. Qin, J. L. Li, and X. Lan, "Path planning of mobile robot in unknown dynamic continuous environment using reward-modified deep Q-network," *Optimal Control Applications and Methods*, p. 2781, 2021.

[28] J. Yu, Y. Su, and Y. Liao, "The path planning of mobile robot by neural networks and hierarchical reinforcement learning," *Frontiers in Neurorobotics*, vol. 14, p. 63, 2020.

[29] Y. Wang, Y. Fang, P. Lou, J. Yan, and N. Liu, "Deep reinforcement learning based path planning for mobile robot in unknown environment," *Journal of Physics: Conference Series*, vol. 1576, no. 1, Article ID 012009, Jun. 2020.

[30] H. Zhang, P. Wang, C. Ni, N. Cheng Hongbo Zhang, and N. Cheng, "Deep Q network algorithm based on sample screening," *International Conference on Computer Application and Information Security*, vol. 12260, pp. 130–135, 2022.

[31] X. Ruan, C. Lin, J. Huang, and Y. Li, "Obstacle avoidance navigation method for robot based on deep reinforcement learning," in *Proceedings of the 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, vol. 2022, pp. 1633–1637, Chongqing, China, March 2022.

[32] I. Kim, S. H. Nengroo, and D. Har, "Reinforcement learning for navigation of mobile robot with LiDAR," in *Proceedings of the 5th International Conference on Electronics, Communication and Aerospace Technology, ICECA*, pp. 148–154, December 2021.

[33] G. Brockman, *OpenAI Gym*, 2016.

[34] M. Chevalier-Boisvert, B. Dzmitry, L. Salem et al., *BabyAI: A platform to study the sample efficiency of grounded language learning*, 2019.

[35] J. D. Co-Reyes, G. Abhishek, S. Suvansh et al., "Guiding policies with language via meta-learning," *ICLR*, pp. 1–10, 2019.

[36] M. Mul, D. Bouchacourt, F. A. I. Research, and E. Bruni, "Mastering Emergent Language: Learning to Guide in Simulated Navigation," pp. 1–14, 2019, https://arxiv.org/abs/1908.05135.

[37] M. Hutsebaut-Buysse, K. Mets, and S. Latré, "Fast Task-Adaptation for Tasks Labeled Using Natural Language in Reinforcement Learning," pp. 1–10, 2019, https://arxiv.org/abs/1910.04040.

[38] A. Goyal, I. Riashat, S. Daniel et al., "Infobot: transfer and exploration via the information bottleneck," *ICLR*, pp. 1–21, 2019.

[39] M. Al-Shedivat, L. Lee, R. Salakhutdinov, and E. P. Xing, "On the complexity of exploration in goal-driven navigation," vol. 0, 2018, https://arxiv.org/abs/1811.06889.

[40] R. Chourasia and A. Singla, "Unifying Ensemble Methods for Q-Learning via Social Choice Theory," pp. 1–11, 2019, https://arxiv.org/abs/1902.10646.

[41] W. Shang, A. Trott, S. Zheng, C. Xiong, and R. Socher, "Learning World Graphs to Accelerate Hierarchical Reinforcement Learning," 2019, https://arxiv.org/abs/1907.00664.

[42] A. Goyal, S. Sodhani, J. Binas, X. Bin Peng, S. Levine, and Y. Bengio, "Reinforcement learning with competitive ensembles of information-constrained primitives," 2019, https://arxiv.org/abs/1906.10667.

[43] D. Chen, Q. Yan, S. Guo, Z. Yang, X. Su, and F. Chen, *Learning Effective Subgoals with Multi-Task Hierarchical Reinforcement Learning*, 2019.

[44] R. Raileanu and T. Rocktäschel, "RIDE: rewarding impact-driven exploration for procedurally-generated environments," *ICLR*, vol. 2020, pp. 1–21, 2020.

[45] A. Campero, R. Raileanu, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette, "Learning with amigo: adversarially motivated intrinsic goals," *ICLR*, vol. 2021, pp. 1–19, 2021.

[46] S. Mirchandani, S. Karamcheti, and D. Sadigh, "ELLA: exploration through learned language abstraction," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 1–19, NeurIPS, 2021.

[47] H. Zhu, I. C. Paschalidis, and M. E. Hasselmo, "Feature extraction in Q-learning using neural networks," in *Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 3330–3335, Melbourne, VIC, Australia, December 2017.

[48] S. Phaniteja, P. Dewangan, P. Guhan, A. Sarkar, and K. M. Krishna, "A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots," in *Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1818–1823, December 2017.

[49] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *Proceedings of the IEEE International Conference on Computing, Communication and Automation, ICCCA*, pp. 350–354, Greater Noida, India, Dec. 2017.

[50] A. Kilin, P. Bozek, Y. Karavaev, A. Klekovkin, and V. Shestakov, "Experimental investigations of a highly maneuverable mobile omniwheel robot," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, Article ID 172988141774457, 2017.

[51] M. O. Tătar, C. Popovici, D. Mândru, I. Ardelean, and A. Pleșa, "Design and development of an autonomous omnidirectional mobile robot with Mecanum wheels," in *Proceedings of the 2014 IEEE International Conference on Automation*, pp. 1–6, Quality and Testing, Cluj-Napoca, Romania, July 2014.

[52] K. Krinkin, E. Stotskaya, and Y. Stotskiy, "Design and implementation Raspberry Pi-based omni-wheel mobile robot," in *Proceedings of the 2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*, pp. 39–45, St. Petersburg, Russia, November 2015.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[54] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large Scale visual recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[55] K. He, G. Gkioxari, P. Dollár, R. Girshick, and R.-C. N. N. Mask, in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, Cambridge, MA, USA, June 2017.

[56] S. Ren, K. He, R. Girshick, J. Sun, and R.-C. N. N. Faster, "Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, Article ID 11371149, 2016.

[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, Las Vegas, NV, USA, June 2016.

[58] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Santiago, Chile, December 2015.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, Columbus, OH, USA, June 2014.

[60] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

[61] M. Tan and Q. V. Le, "Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks," 2020, https://arxiv.org/abs/1905.11946.

[62] M. Moghaddam, M. Charmi, and H. Hassanpoor, "Jointly Human Semantic Parsing and Attribute Recognition with Feature Pyramid Structure in EfficientNets," *IET Image Process*, 2021.

[63] Y. Zhou, J. Shi, X. Yang, C. Wang, S. Wei, and X. Zhang, "Rotational objects recognition and angle estimation via kernel-mapping CNN," *IEEE Access*, vol. 7, pp. 116505–116518, 2019.

[64] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, vol. 11, no. 10, pp. 1141–1151, 2002.

[65] M. Pfeiffer, S. Shukla, M. Turchetta et al., "Reinforced imitation: sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[66] S.-H. Hsu, S.-H. Chan, P.-T. Wu, K. Xiao, and L.-C. Fu, "Distributed deep reinforcement learning based indoor visual navigation," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 2532–2537, Madrid, Spain, October 2018.

[67] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning; towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6252–6259, Brisbane, Australiadoi, May 2018.

[68] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[69] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning," *End-to-end Decentralized Multi-Robot Navigation in Unknown Complex Environments via Deep Reinforcement Learning*, 2019.

[70] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6180–6190, 2020.

[71] F. Leiva and J. Ruiz-Del-Solar, "Robust RL-based map-less local planning: using 2D point clouds as observations," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5787–5794, 2020.

[72] Q. M. Qadir, T. A. Rashid, N. K. Al-Salihi, B. Ismael, A. A. Kist, and Z. Zhang, "Low power wide area networks: a survey of enabling technologies, applications and interoperability needs," *IEEE Access*, vol. 6, Article ID 77454, 2018.

[73] M. Shahjalal, M. K. Hasan, M. M. Islam, M. M. Alam, M. F. Ahmed, and Y. M. Jang, "An overview of AI-enabled remote smart- home monitoring system using LoRa," in *Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, vol. 2020, pp. 510–513, Fukuoka, Japan, Feb. 2020.

[74] G. Sartoretti, J. Kerr, Y. Shi et al., "PRIMAL: pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[75] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning," in *Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 2493–2500, Tianjin, China, Aug 2019.

[76] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," 2018, https://arxiv.org/abs/2207.10417.

[77] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 79–86, Vancouver, BC, Canada, Dec. 2017.