2020 17th International Conference on Ubiquitous Robots (UR)
June 22-26, 2020. Kyoto, Japan

Outdoor Robot Navigation System using Game-Based DQN and Augmented Reality

Sivapong Nilwong and Genci Capi, *Member, IEEE*

Abstract— **This paper presents a deep reinforcement learning based robot outdoor navigation method using visual information. The deep q network (DQN) maps the visual data to robot action in a goal location reaching task. An advantage of the proposed method is that the implemented DQN is trained in the first-person shooter (FPS) game-based simulated environment provided by ViZDoom. The FPS simulated environment reduces the differences between the training and the real environments resulting in a good performance of trained DQNs. In our implementation a marker-based augmented reality algorithm with a simple object detection method is used to train the DQN. The proposed outdoor navigation system is tested in the simulation and real robot implementation, with no additional training. Experimental results showed that the navigation system trained inside the game-based simulation can guide the real robot in outdoor goal directed navigation tasks.**

I. Introduction

Mobile robots operating in indoor or outdoor environments require suitable navigation algorithms to complete their tasks. Conventional outdoor navigation algorithms utilize global positioning systems (GPS) or magnetic compasses to guide the robot. Implementations of GPS devices vary in robotics applications such as autonomous agricultural robot [1], a low-cost small transporter robot [2], and a railway maintenance robot [3]. Despite their simplicity, these conventional sensors are mostly unreliable in different conditions. For instance, typical GPS devices such as GPS-enabled smartphones have an average error of 4.9 m under open sky [4]. Additionally, GPS devices have less accuracy when they are used near large structures such as tall buildings. Complexity of navigation algorithms can also cause problems for mobile robots in outdoor environments. According to the second law of software evolution, systems become more complex as their programs evolve [5]. Varied conditions and decision choices in outdoor environments can make navigation systems more complex, difficult to maintain and have high possibility of software errors.

In recent years, reinforcement learning (RL) algorithms have been widely employed for their capabilities of learning through trial-and-error experiences. RL algorithms can be found implemented in many robotics works such as human-robot interaction system [6], path planning for mobile robots [7], and robot manipulator controllers [8]. RL algorithms are also employed for vision-based navigation tasks, such as [9] and [10] that guided mobile robots using visual information from their front cameras. One significant RL algorithm is the Deep Q Network (DQN). DQN combines reinforcement learning with deep neural networks. Performances of DQN in making decisions have been proved by playing different video games using raw visual data [11]. The DQN agent could play different video games at human-level. Some robotics works such as the soccer robots [12] also employed DQN for their robot systems. RL algorithms normally train their artificial agents to perform different tasks by experiencing trial-and-error consequences. This trial-and-error method can cause damage to robots if RL agents are directly trained on robots. Simulations are mostly preferred for training in RL, despite great differences between simulation and real environments [13], especially for DQNs which require long training time.

From performances of DQN in making decisions from visual data, navigation complexities can be reduced by DQN. Simulation environments which can resemble real environments are needed to train DQNs. Therefore, this paper presents an outdoor robot navigation system using visual data and DQN. Camera is used as the only sensor of the robot to minimalize the sensor reliability problems. The algorithm complexity problems are reduced by having DQN learn the navigation tasks instead of traditional hard coding. Since simulation environments are needed for DQN training, the FPS game-based ViZDoom platform is implemented. ViZDoom offers fast and simple simulation environments that share similar perspectives with the mobile robot camera [14]. Another advantage of ViZDoom is that it makes easy development of the outdoor simulated environments, which are not easy to build in simulation. The trained DQNs from the simulation are directly applied to the real robot without modification. Simple marker-based augmented reality (AR) method is employed in the robot to reduce differences between simulation and real environments. An object detection method is used in the AR for robot marker detection. The navigation system aims to guide the robot within the short distances of 2-10 m, since GPS devices are less reliable in smaller areas. Experiments include DQN tests in the simulation and real robot. Experimental results report efficiency of our navigation system for simple navigation tasks of outdoor mobile robots.

The paper is organized as follows: Section 2 describes the proposed navigation system and its components, Section 3 explains the experiments and shows experimental results, and Section 4 concludes the contents and discuss future works.

S. Nilwong is with the Graduate School of Science and Engineering, Hosei University, Koganei, Tokyo 184-8584 Japan (e-mail: sivapong.nilwong.46@stu.hosei.ac.jp).

G. Capi is with the Department of Mechanical Engineering, Hosei University, Koganei, Tokyo 184-8584 Japan (corresponding author to provide phone: +81-42-387-6148; fax: +81-42-387-6148; e-mail: capi@hosei.ac.jp).

978-1-7281-5715-3/20/$31.00 ©2020 IEEE

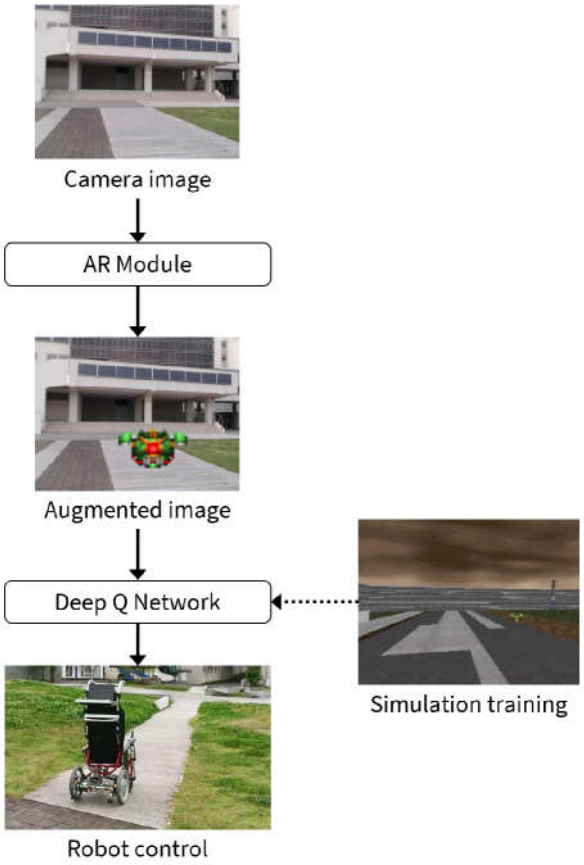

Figure 1. Diagram of the outdoor robot navigation system.

Figure 2. Wheelchair robot platform.

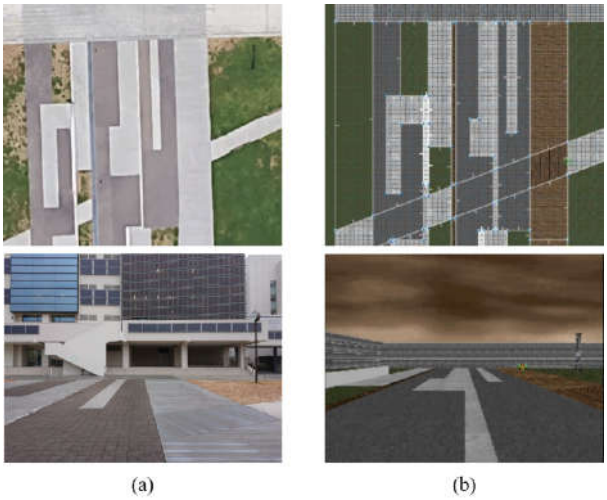

Figure 3. Comparisons between (a) the real world and (b) the simulation.

II. Vision-Based Robot Navigation System

The flowchart of proposed vision-based outdoor robot navigation system is shown in Fig. 1. Basically, our navigation system uses the DQN which maps the camera images to robot action decisions to reach the goal. DQN for guiding the robot is trained in the game-based simulation before its usage. The trained DQN does not know the real environments because it was trained in the simulation. The simple marker-based AR algorithm and a simple color-based object detector are used to enhance robot perspectives. Camera image is fed to the AR module, where the goal object in the image is detected. The goal is replaced with the simulated object that DQN was trained with. Therefore, our DQN decides proper actions for the robot from augmented camera images. There are three main components in our navigation system, including DQN, simulation environments, and the AR module. Further details of the robot and navigation system components are described in following subsections.

A. Wheelchair Robot Platform

The performance of outdoor navigation algorithms is implemented in the wheelchair robot. Fig. 2 shows the robot and its attached hardware. The robot platform consists of a laptop, two Yamaha AC motors, and a motor controller. Laptop is placed on the placeholder at the rear of the robot. Two AC motors are located at the left and the right rear wheels. The controller for two AC motors is placed beneath the seat. Logitech C920 camera is used as the only sensor for the robot. Location of the camera is on the acrylic plate, top of the robot. The laptop is the robot main controller. The camera and the motor controller are connected to the laptop via USB cables. Visual data are gathered by the camera to be used on our navigation system and control the motors. The data from the camera is in the form of RGB images at the size of 320×240 pixels. Two AC motors can be controlled to move the robot forward, turn left, or turn right.

B. ViZDoom Simulation Environments

ViZDoom is a software platform for machine learning research based on the FPS game Doom [14]. Simulation environments in ViZDoom can be created faster and easier than building simulations from their bases. Doom game itself is lightweight and can be processed fast on typical personal computers. ViZDoom is highly customizable. There are many editing tools for Doom and ViZDoom. We used ViZDoom and the editing tool named Doom Builder [15] to create a simulation environment for the training of our DQN. The created environments are projected to the artificial agent in first-person perspective, which is the same perspective that the robot sees through the attached camera.

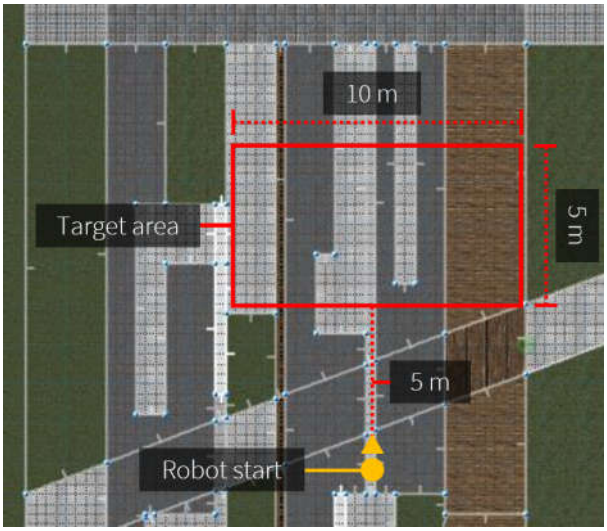

Figure 4. Simulation area with the agent (robot) start and the target area.

TABLE I. GAME SIMULATION SETTINGS

Simulation properties	Applied settings
Step limit per run	250 steps
Time per step	1/35 s
Agent input	320×240×3 (RGB image)
Agent actions	3 (forward, turn left, turn right)

The simulation environment was created with the field of Hosei University, Koganei campus, Japan, as the reference. Details of the environment are not precisely the same as the real world. Textures and objects used in the environment are all the game resources from ViZDoom. Fig. 3 compares the differences between the simulation and the real field. Similarities between two environments can be found such as grass and floor. As based on a video game, interactions between the environment and artificial agents are more oriented into games. Agents in the simulation can interacts by moving and gathering items. The simulation can respond to the agents in many ways, such as increasing or decreasing agent status, giving rewards, and the agent termination.

The navigation task includes only a simple goal reaching in short distances. The simulation area is limited to approximately 20×18 m, as seen in Fig. 4. This simulation area was used for both DQN training and experiments. The robot starting location and orientation are fixed, so as the target area. The target area is 5 m far from the robot starting position. The robot initially faces toward the target goal location. The target area has the size of 10×5 m, with the goal object randomly generated inside it. Green grass areas are set to instantly terminate the agents if stepped on.

Table 1 shows settings of the simulation environment. The step limit is set to 250 steps in each simulation instance, in which one step is set to 1/35 s. This means one simulation

TABLE II. REWARDS PROVIDED BY SIMULATION

Agent states	Provided rewards
Moved	– (distance_x + distance_y) – 3
Dead (step on grass)	– 2,000
Goal	+ 2,000

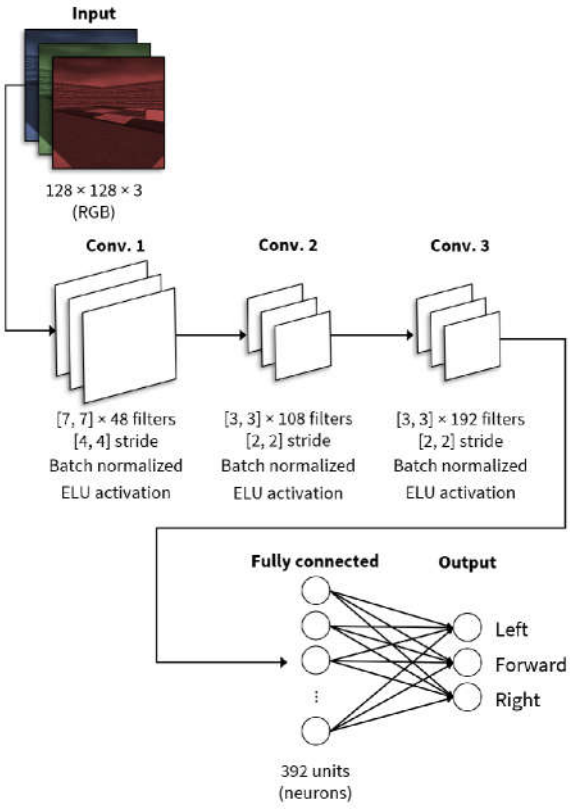

Figure 5. Structure of the DQN for navigation.

instance has the time limit of approximately 7 s. Visual data that agents see from the simulation is in the form of RGB frames at the size of 320×240 pixels. Agents can commit 3 actions, including moving forward, turning left, and turning right, according to possible movements of the real robot. Rewards from the environment are stated in Table 2. Agents tend to move toward the goal as fast as possible, since in each step -3 is added to the accumulated reward. Distances between the agent and the goal are also calculated as rewards, which become more negative as the agent moves away from the goal. Termination state is also considered bad. The large negative reward of -2,000 will be given if the agent moves into the grass and terminated. On the other hand, the rewards are increased as the agent moves toward the goal, with the big reward of +2,000 provided as the agent reaches the goal. A simulation instance has three termination conditions: (1) the agent reaches the goal, (2) the agent is terminated, and (3) the simulation exceeds the step limit.

TABLE III. DQN TRAINING SETTINGS

Training properties	Applied settings
Episodes	1201
Steps per episode	250
Initial exploration parameter	1.0
Exploration decay	0.000192
Learning rate	0.00024
Reward discount	0.949
Memory batch per training	96
Memory size	1,000,000

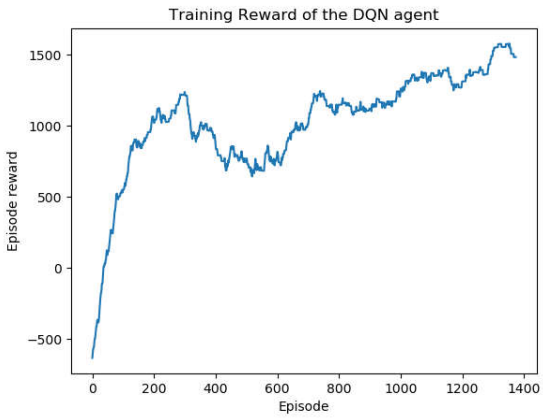

Figure 6. Training rewards of the DQN, convolved every 128 episodes.

C. Deep Q Network for Navigation

DQN shares the same principle as other RL algorithms, which is the ability of learning by experiencing consequences of actions committed to the environments. DQN is the combination of RL and artificial neural networks [11]. The strategy of DQN in finding optimal policies is the greedy strategy, same as its predecessor, q-learning. The optimal policies that q-learning finds tend maximize rewards from consequences of actions [16]. The vanilla version of DQN was implemented for our navigation system. The vanilla DQN replaces q-table in q-learning with neural networks. The neural networks in DQN generate the action values for artificial agents, in which actions with the highest action values are selected and committed. Training strategies in the vanilla DQN, including the experience replay and the target-action value function were applied to the training of our DQN. We used two equations from [11] to calculate target action-values of experiences in experience replay, depending on states of experiences:

$$y_i = r_i \tag{1}$$

$$y_i = r_i + g \times max_{a'} Q_n(s_{i+1}, a') \tag{2}$$

where y_i is the target-action value of the experience i, r_i is the reward from committing an action stored in the experience, g is the reward discount, and $max_{a'} Q_n(s_{i+1}, a')$ is the maximum target-value from the target-value function in the consequence state of the experience. Target action-value will be calculated by (1) if the training episode of DQN is terminated at the given step. Calculation from (2) will be applied to other cases.

Structure of out DQN is displayed in Fig. 5. Visual data is fed through the input layer, continuing into convolutional layers, then processed through a fully connected layer and output one of three possible actions. The input size of the DQN is 128×128×3, which is refers to the size of RGB image. Our DQN considers a single RGB image as an input, instead of a traditional stack of luminance frames [11]. The single RGB image was employed to reduce effects of framerate differences between the real robot camera and the simulation. However, the use of single RGB image input for DQN can cause temporal limitation problems and reduce DQN performances. Convolutional layers in our DQN were inspired by the pyramidal design [17]. The first convolutional layer has a large filter size of 7×7, with the stride of 4×4. The later convolutional layers apply small filters of 3×3, with stride 2×2. The number of filters starts at 48 for the first convolutional layer. For the deeper convolutional layers, the number of filters has increased to 108 and 192 in the second and the third convolutional layers, respectively. Batch normalization and ELU activation are attached after each convolutional layer. Our DQN structure was optimized through trial-and-error process.

During training and navigation, the input RGB images have the first 20 rows on the top of images cropped. The removed areas mostly are the sky or the background buildings, which have no impact on the robot navigation. The cropped images are resized to 128×128×3 pixels, then fed to the DQN. As aforementioned, we implemented the vanilla DQN and its training process for our DQN. Variables for our DQN training are different from [11], as described in Table 3. The training variables were set through trial-and-error procedures. We used the "Basic" scenario in ViZDoom [14] to test our DQN structures and training settings, before tuning the DQN to our navigation simulation. The number of episodes was set to 1201, with the step limit of 250 steps per episode. The epsilon-greedy algorithm was implemented with the exploration parameter of 1.0 and the exploration decay rate of 0.000192 for each performed action step. Learning rate of the network was set to 0.00024. The rewards for calculating the target-action values were discounted by the factor of 0.949. The DQN was trained in every step of actions with experiences from the memory, in which the experiences were sampled in a batch of 96. The memory size was set to be able to store 1,000,000 experiences at maximum. Episode rewards during training are convolved and shown in Fig. 6. The DQN robot agent chose proper actions and received more rewards in later episodes.

D. Marker-Based Augmented Reality

Augmented reality algorithms are separated into various categories. The basic AR algorithms typically use markers and computer vision methods to augment camera vision [18].

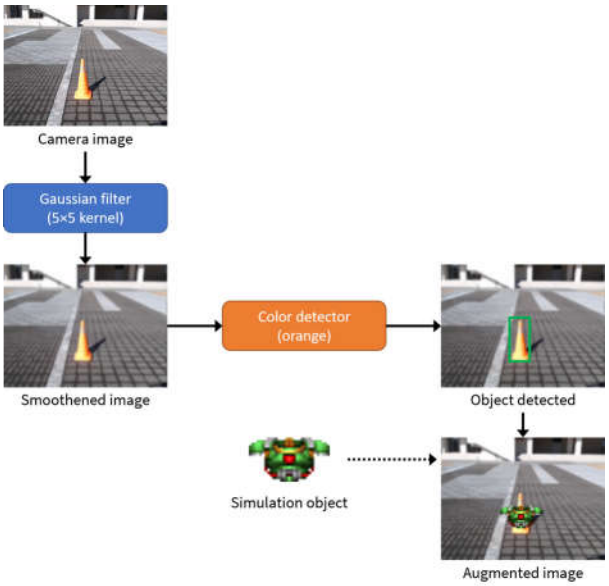

Figure 7. Flows of the augmented reality module.

Figure 8. The goal objects: (a) the real traffic cone, (b) the green armor.

The marker-based AR algorithm is employed to improve the robot vision in our navigation system. A simple color-based detector was used in our AR to detect the marker objects in camera images. The color-based object detector was selected over neural network-based detection algorithms due to its lightweight and simplicity. Neural network and deep learning detection methods such as the method in [19] require the detectors to be trained. Neural network-based methods also consume more computation resources than simple color detectors when used. Flows of the AR module is illustrated in Fig. 7. We selected the goal object which has different color to the environment. Camera image is smoothened through the filter to reduce noises. We used a gaussian filter with the kernel size of 5×5 to smoothen camera images. The object detector then finds the goal by detecting object colors. The goal object is detected and replaced with the object from the simulation, which resized according to the goal object.

Since an object is needed to be detected as the real goal, we used a traffic cone as the goal object. The traffic cone was chosen because of its appearances. The appearances of traffic cones are the same from all directions. Colors of the traffic cones are usually unique and recognizable in almost every environmental condition. Our AR module replaces the traffic

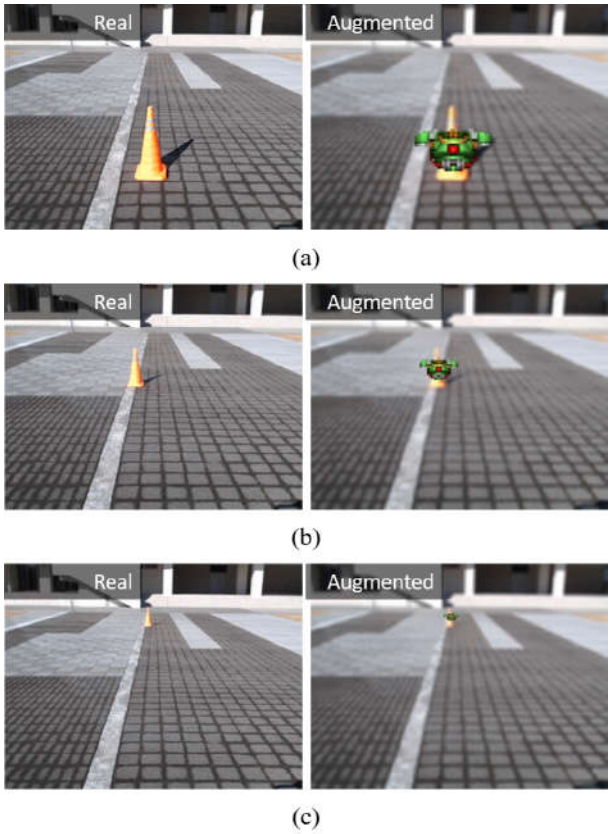

Figure 9. Camera images (left) and augmented images (right) at (a) 2 m, (b) 5 m, and (c) 10 m.

cone in Fig. 8 (a) with the green armor from the simulation in Fig. 8 (b). Fig. 9 shows the traffic cone in camera images and their augmented images at different ranges. We augmented camera images by replacing the traffic cone with the green armor, before feeding the augmented image to the DQN for robot navigation. The replaced object was smaller as the distance increased.

III. Experimental Results

The navigation system was tested with simple navigation tasks in two environments: the simulation and the real outdoor environment. The robot had to move to the goal, which was randomly placed in front of the robot. The robot and the goal setups were the same as the training environment in Fig. 4, in which there were a target area for random goal locations. The goal object was randomly placed in the target area. For the simulated experiments, the green armor in Fig. 8 (b) was used as the goal. The traffic cone in Fig. 8 (a) was used as the goal for the real robot. The starting position of the robot was the same as in the training environment. Simulation experiments included the navigation using the trained DQN. Real robot tests had an addition of the AR module to augment camera images. We measured the robot performance based on the reaching the target success rate and crash rate, since the rewards in the real environment cannot be measured in the same way as in the simulation. Success rate was measured by the times that our robot reached the goal. Crash

TABLE IV. EXPERIMENTAL RESULTS OF THE NAVIGATION SYSTEM

Measurements	Experimental results	
	Simulation	*Real robot*
Number of experiments	100	50
Success count	98	13
Success rate	98%	26%
Crash count	1	3
Crash rate	1%	6%
Maximum distance	10 m	4 m

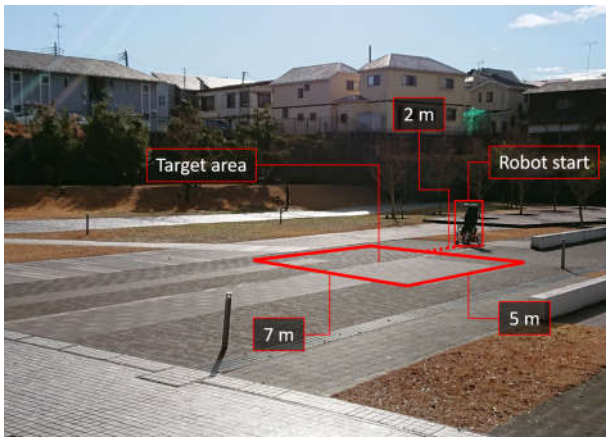

Figure 10. Experimental area in the real robot experiments.

(a)

(b)

(c)

Figure 11. Wheelchair robot reaching the goal object at (a) 3 m, (b) 2 m and (c) reached the goal object.

rate was measured by the times that our robot moved into the grass Experimental results from the simulation and the real robot are presented in Table 4.

A. *Simulation Experiments*

The outdoor navigation system was tested inside the same simulation with the same simulation settings as in the training. We set the goal randomly in the target area. Goal distances were in the range of 5-10 m, same as in the training environment. The robot initial location is the at the same place, with the same orientation in every test. Goal locations were changed every time the test started. The robot had to move to the goal which was placed in front of it. Our robot was tested in the simulation for 100 times. From the simulation results, the robot successfully reached the goal 98 times out of 100, or 98% success rate was achieved. The robot moved into the grass 1 time, which resulted in 1% crash rate. Another 1% was due to the time out, in which the robot moved over the designated step limit of 250 steps.

Though 98% success rate was achieved, there was a crash rate of 1% in simulation tests. It is suspected that the robot crashed because the robot lacks the knowledge of prior image frames. As implemented in [11], the DQN applied a 4-frame stack as an input for playing video games. However, we implemented only a single RGB image frame as the DQN input. The reason for the other fail to reach the goal due to the time out is related with the nature of DQN. Because DQN has a greedy policy as its core, it tried its best to achieve the most reward possible by avoiding taking risky actions that may cause large negative rewards.

We directly implemented the DQN of the simulation robot into the real robot, since our DQN achieved 98% success rate in the simulation experiments.

B. *Real Robot Experiments*

Fig. 10 shows the real experimental area, with the target area and the starting point included. We randomly placed the goal object in the target area. The robot started at the same location, with the same orientation in every test. Target area in real experiments was closer to the robot, in which the distances were reduced to 2-7 m instead of 5-10 m in simulation tests. Width of the target area was also changed to 7 m to match with the field of view in the robot camera. The closer distances were chosen to test the DQN from the shortest distance, which are typically easier for the robot. We gradually increased goal distances as the experiments proceed. The robot was tested for 50 times in the real environment. Augmented images from the camera were used as states. Robot performed an action according to the state and its knowledge from the simulation. A single action of the robot performed for 0.5 second before moving on to the next state and commit the next action.

From experimental results (Table 4), the robot successfully reached the goal object 13 times, resulted in 26% success rate. We counted the success when the robot reached the goal object, as displayed in Fig. 11. The robot moved into the grass 3 times, which resulted in 6% crash rate. Other 68% of the real robot results were time out. The robot was designated with the

time out status if the robot stuck in the same area for longer than 1 minute. All success attempts in the 26% success rate were from the navigation within the maximum distance of 4 m.

As seen in Fig. 9, the robot could detect the goal object up to 10 m. However, the DQN could not guide the robot to the goal beyond 4 m. There are several reasons behind these results. First is the difference between the real environment and the simulation. Our DQN use convolutional neural networks to generate actions according to states, which is the whole augmented images that also include the background environments. This difference in backgrounds could reduce performances of the DQN. Secondly, the augmented reality module could possess several problems since we employed a simple color-based object detector. Pedestrians could be mistaken as the goal object if they were wearing clothes with similar colors to the goal object while walking into camera frames. Another problem could come from the DQN training. The training of DQN aims to have DQN generate and select actions based on action values. Some unseen states could cripple DQN performances by a large margin.

Despite the failure of 74%, the success rate of 26% showed that the simulation made from a video game can be used to train the DQN to guide the real robot for simple navigation tasks. However, assists from some other algorithms such as augmented reality, are highly recommended for real robot implementations.

IV. Conclusion

The vision-based outdoor robot navigation system was proposed. We used DQN to guide the robot to the short distance goals from visual data. The implemented DQN was trained in the game-based simulation environment, then directly employed to the real robot without any changes. Marker-based AR and color-based detector were used to augment the robot vision by replacing the real goal with the goal from the simulation. The navigation system was tested in the simulation and the real outdoor environments. Experimental results from the simulation showed the high effectiveness of the navigation system inside the simulation. The real experiments showed a potential of the game-based DQN and a simple marker-based AR method for simple navigation tasks in short distances.

Future works can include the use of policy-based or actor-critic deep reinforcement algorithms, which could be more suitable for real robot implementations. More navigation tasks are to be included further, such as navigation in complex environments or guiding the robot to do multiple tasks.

References

[1] K. Shaik, E. Prajwal, S. B, M. Bonu and V. R. Balapanuri, "GPS Based Autonomous Agricultural Robot," 2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C), Bangalore, 2018, pp. 100-105.

[2] M. Abdullah-Al-Kaiser, D. J. Paul, A. I. Khan, C. Shahnaz and S. A. Fattah, "A cost effective GPS guided autonomous object transporter robot for disaster management and industrial automation," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 2637-2641.

[3] N. Mahfuz, O. A. Dhali, S. Ahmed and M. Nigar, "Autonomous railway crack detector robot for bangladesh: SCANOBOT," 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, 2017, pp. 524-527.

[4] F. van Diggelen and P. Enge, "The World's first GPS MOOC and Worldwide Laboratory using Smartphones," Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015), Tampa, Florida, September 2015, pp. 361-369.

[5] M. M. Lehman, "Laws of software evolution revisited," *Software Process Technology*, Springer Berlin Heidelberg, 1996, pp. 108-124.

[6] H. Modares, I. Ranatunga, F. L. Lewis and D. O. Popa, "Optimized Assistive Human–Robot Interaction Using Reinforcement Learning," in *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 655-667, March 2016.

[7] T. Yan, Y. Zhang and B. Wang, "Path Planning for Mobile Robot's Continuous Action Space Based on Deep Reinforcement Learning," 2018 International Conference on Big Data and Artificial Intelligence (BDAI), Beijing, 2018, pp. 42-46.

[8] K. Hwang, J. Lee, Y. L. Hwang and W. Jiang, "Image base visual servoing base on reinforcement learning for robot arms," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, 2017, pp. 566-569.

[9] X. Ruan, D. Ren, X. Zhu and J. Huang, "Mobile Robot Navigation based on Deep Reinforcement Learning," 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 6174-6178.

[10] P. Yue, J. Xin, H. Zhao, D. Liu, M. Shan and J. Zhang, "Experimental Research on Deep Reinforcement Learning in Autonomous navigation of Mobile Robot," 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), Xi'an, China, 2019, pp. 1612-1616.

[11] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, 2015, pp. 529–533.

[12] J. Kim, B. Kim, J. Yoon, M. Lee, S. Jung and J. y. Choi, "Robot Soccer Using Deep Q Network," 2018 International Conference on Platform Technology and Service (PlatCon), Jeju, 2018, pp. 1-6.

[13] L. Tai, G. Paolo and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 31-36.

[14] M. Kempka, M. Wydmuch, G. Runc, J. Toczek and W. Jaśkowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, 2016, pp. 1-8.

[15] J. W. Anderson, Doom Builder: An Illustrated Guide, July 2004.

[16] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, issue 3–4, pp. 279–292, May 1992.

[17] I. Ullah and A. Petrosino, "About pyramid structure in convolutional neural networks," 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, 2016, pp. 1318-1324.

[18] R. Yang, "The study and improvement of Augmented reality based on feature matching," IEEE 2nd International Conference on Software Engineering and Service Science, Beijing, 2011, pp. 586-589.

[19] D. Hossain, G. Capi, M. Jindai and S. Kaneko, "Pick-place of dynamic objects by robot manipulator based on deep learning and easy user interface teaching systems," *Industrial Robot*, vol. 44, issue 1, pp. 11-20, January 2017.