Robot Reinforcement Learning for Automatically Avoiding a Dynamic Obstacle in a Virtual Environment

Phuong Chu, Hoang Vu, Donghyeon Yeo, Byeonggwon Lee, Kyhyun Um, and Kyungeun Cho*

Dept. of Multimedia Engineering, Graduate Schools of Dongguk University, 26, Pil-dong 3-ga, Jung-gu Seoul, 100-715, Republic of Korea
cke@dongguk.edu

Abstract. In a virtual environment, a robot can serve people by bringing things to them. However, when a robot moves within a house, it collides with a dynamic obstacle. These collisions make it difficult for a robot to complete its mission. We therefore apply reinforcement learning to the robot to make it more intelligent. Consequently, the robot can automatically move to avoid the dynamic obstacle in order to successfully complete its mission.

Keywords: Reinforcement learning, virtual environment, automatic, dynamic obstacle, virtual robot.

1 Introduction

Nowadays, humanoid robots can help people execute many tasks. However, robots have limited physical capabilities and 'intelligence'; consequently, they are unable to perform many desirable tasks. Robots should therefore be improved in both physical and artificial intelligence aspects. In this study, we focus on how to make a robot more intelligent. We employ a humanoid robot ('Nao' by Aldebaran), a real house with furniture, and a program to control the robot's behavior, such as moving, rotating, grasping an object, and releasing it. The robot is expected to move within the house and fetch the object for a human. When it moves, it should avoid colliding with the dynamic obstacle that randomly move. However, the robot does not know how to avoid the obstacles when navigating to the object and bringing it to the human.

If we use the A* [1] algorithm to find the path, the robot cannot avoid the obstacle because it may be situated in the shortest path chosen by the robot. Our approach to solving this problem is reinforcement learning. Nevertheless, another problem occurs. If we use a reinforcement learning algorithm, the robot must move innumerable times in as many periods; furthermore, the learning time is likewise very long. A real robot such as Nao cannot implement this immense task. We therefore must create a virtual house that emulates the real house.

Within our virtual house, a virtual human can stand in any position. A virtual object, such as a ball, is situated in the house; it can be placed in any position. In addition,

* Corresponding author.

a moveable object—a virtual dynamic obstacle—can randomly move around the house. The virtual robot should navigate to the object, grasp it, and bring it to the human. When the virtual robot moves, it can collide with the moveable object. It will therefore not successfully complete the task. To address this issue, we apply reinforcement learning [4-8] to the virtual robot so that it can automatically move to avoid the dynamic obstacle. We then apply the result of this virtual experiment to the real humanoid robot.

2 Related Work

In controlling a robot, path finding is very important. Many path finding studies exist [1-3]. We could employ the A* or Dijkstra algorithm [1], which focus on the shortest path. However, a problem occurs if either of these algorithms is used for mapping with dynamic obstacles. In many cases, the obstacle remains in the shortest path; therefore, the robot cannot move to its target without colliding with the obstacle.

The proposed algorithm was inspired by the work of [2,3]; however, rather than use a genetic algorithm, we provide a new approach. The reinforcement learning algorithm is very useful for learning human-robot interactions [4]. This algorithm can be used in a virtual environment for increasing the learning rate and saving time. The result is then applied in a real environment [4]. In this study, we combine reinforcement learning with path planning. We divide the robot movements into two parts. In the first part, the robot mission is to navigate to a ball and to grasp it. In the second part, the robot is to navigate to the human and give the ball to him/her.

3 Virtual Learning Approach

3.1 Virtual Environment Configuration

Our virtual environment was comprised of a virtual house with three rooms—a bedroom, living room, and kitchen—as shown in Figure 1. Each room contained various furniture objects, such as a bed, sofa, stove, chair, and table, which were all static. We used a 9 x 5 grid to calculate the respective positions of the robot, human, ball, and obstacle. We used a two-dimensional coordinate. We converted the positions of all objects from three-dimensional coordinates to two-dimensional ones. All positions of the same cell were converted to only one position in the two-dimensional coordinate. Therefore, when two objects remained in one cell, we recognized that they collided with each other.

All of the static objects had fixed positions in the house; therefore, some cells in the grid were permanently occupied. Not all moveable objects could move in these cells. The positions of all objects in this situation are shown in Table 1. In each period, the respective positions of the human and ball were static; however, they could be changed in a different period. The dynamic obstacle could randomly move within the house. Each time, it moved to the nearest empty cell in one of four directions: up, down, left, or right. The human stood in any position and waited for the robot to bring the object. The robot could move in any of the same four directions as the dynamic obstacle.

Table 1. Object positions within the virtual house

Object	Position
Robot	(0,0)
Human	(4,4)
Obstacle	(5,1)
Ball	(8,4)
Bed	(0,3), (0,4), (1,3), (1,4)
Sofa	(2,2), (2,3), (2,4)
Stove	(8,0), (8,1)
Table	(8,3), (8,4)
Chair	(6,4)

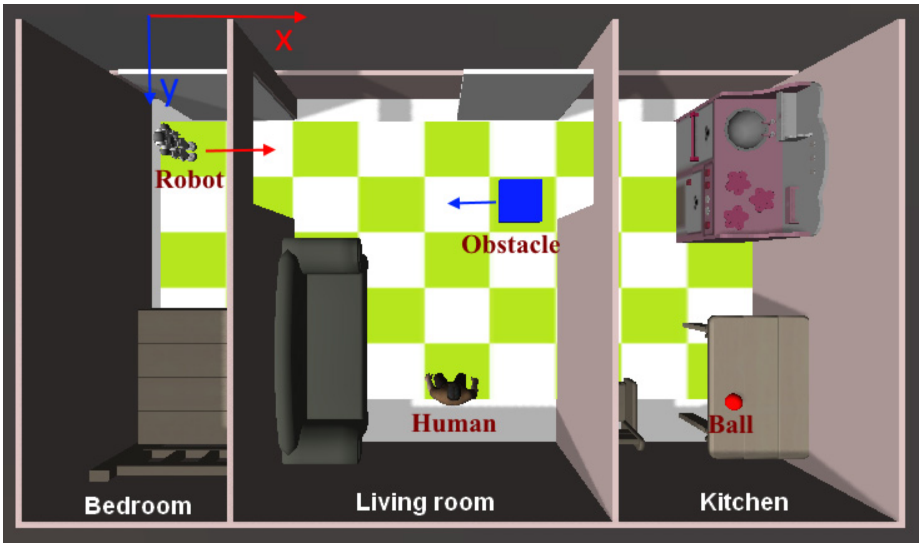

Fig. 1. Virtual house

3.2 Application of Reinforcement Learning to the Virtual Robot

Reinforcement learning was used to find an optimal action-selection policy for the robot. The reinforcement learning algorithm includes a function that calculates the quality, Q, of a state-action combination:

$$Q: S \times A \rightarrow R \tag{1}$$

where S is a set of states, A is a set of actions, and R is the result.

We used a Q table for saving the priorities of all states of the robot's actions (2). Each value in the table depended on the positions of the robot, human, obstacle, and ball, as well as on the moving direction of the robot. We initialized all values of the Q table as zero.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \times \left(r_{t+1} + \gamma \times maxQ_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right) \tag{2}$$

where s_t, a_t are the previous state and previous moving action, respectively, and s_{t+1}, a_{t+1} are the next state and next moving action, respectively. In addition, r_{t+1} is the reward observed after the robot implements moving action a_t in state s_t, $maxQ_t(s_{t+1}, a_{t+1})$ is an estimate of optimal future value, α is the learning rate, and γ is the discount factor.

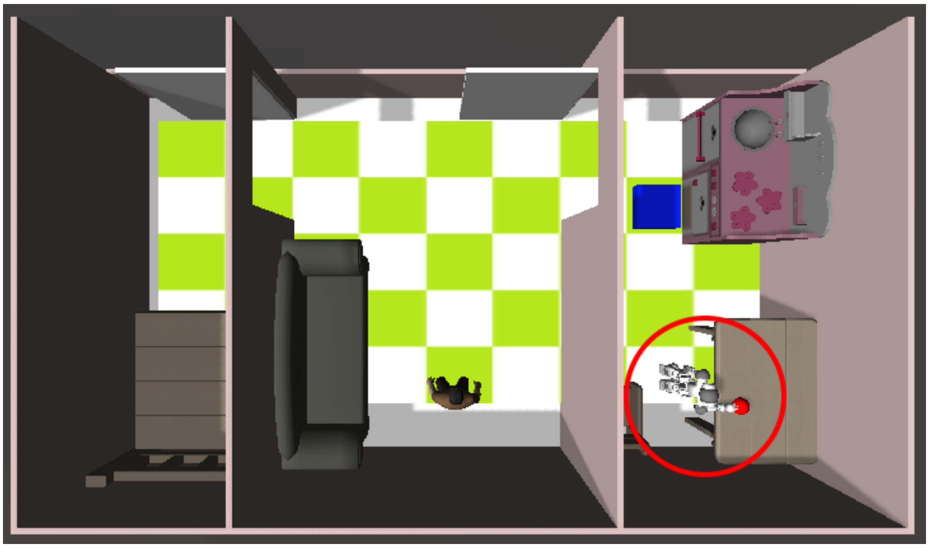

Fig. 2. Robot grasps the ball

After any given step, the value in the Q table was updated. In any situation, the robot chose the moving direction with the highest priority in the Q table. If all values were equal, the robot randomly chose one of the four directions. If the robot moved to a position near the ball, the robot grasped the ball, as shown in Figure 2. The robot then continued moving.

If the robot and human were in the same position, and the robot was holding the ball, the robot gave the ball to the human, as shown in Figure 3. In this case, the robot received a positive reward. In the second scenario, the robot met the human but did not bring anything to him/her. This incurred a bad result and a negative reward for the robot. In the third scenario, the robot received a negative reward if it collided with the dynamic obstacle, as shown in Figure 4. The period was completed in all three cases. The next period randomly began with another situation.

When each period ended, the robot received a new experience. In the same situation, if the robot received a negative reward during the last period, it chose a different path in the subsequent period. This learning was repeated many times to provide the robot with many experiences. When the robot obtained a sufficiently large number of experiences, in each period, it chose the direction that would most effectively help it receive a positive reward for each situation. After a few million periods, the robot could automatically move to complete the mission and avoid the dynamic obstacle.

When it was evident that the virtual robot could intelligently move, we exported the Q table data into a database. We then imported this database into a module that we used to control the real robot. We situated a camera on the ceiling of the real house to obtain a map. In the real robot control module, we recognized the positions of all real objects and converted them into two-dimensional coordinates, as was done for the virtual environment. In each situation, we read the values from the Q table and provided the real robot with the optimal choice of the direction in which to move; i.e., up, down, left, or right.

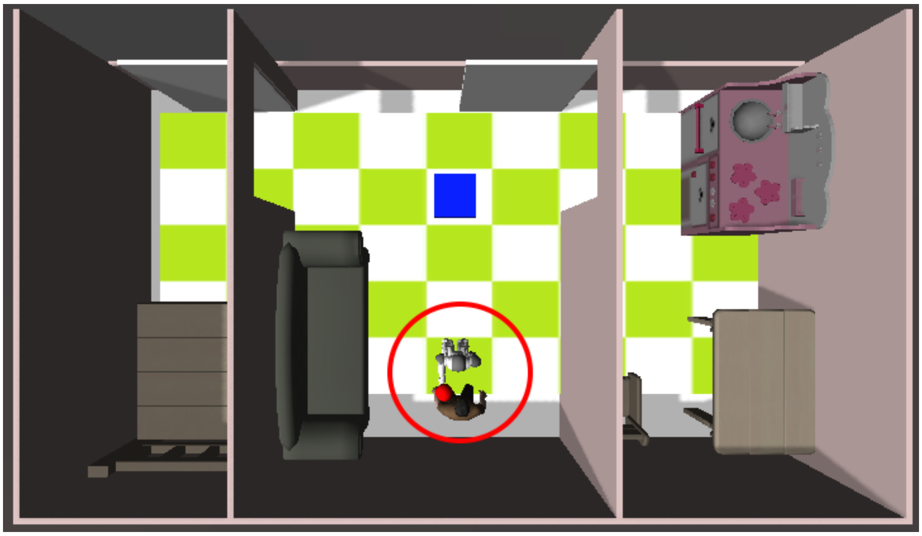

Fig. 3. Robot gives the ball to the human

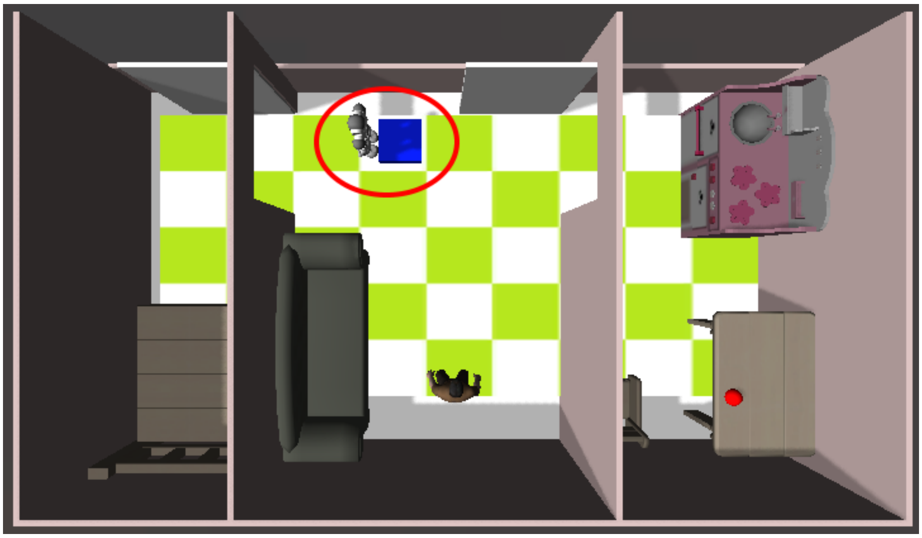

Fig. 4. Robot collides with the dynamic obstacle

4 Experiment

We conducted an experiment to evaluate the proposed approach. We counted the number of collisions that occurred between the virtual robot and dynamic obstacle in 10,000 respective periods, as shown in Figure 5. From the first to the ten-thousandth period, the number of collisions between the virtual robot and obstacle was very high. The number of periods had collision took 85% of the total of all periods. However, it very quickly decreased within the next one million periods. By approximately the one-millionth period, the number of periods that included a collision was only 30%. From that point, the percentage gradually decreased from 30% to less than 20%.

We additionally counted the number of successful missions of the virtual robot. A period was deemed successful if the virtual robot navigated to the ball's position, grasped it, and brought it to the human without colliding with the obstacle. A period was regarded as a failure in two cases: if the robot collided with the obstacle; and if the robot met the human but did not bring anything to him/her.

In our experiment, the second type of mission failure rarely occurred because the dynamic obstacle was always moving and the robot usually collided with it. In addition, after learning from a few thousand periods, the robot 'knew' that it should not meet the human without bringing anything to him/her. The numbers of successful periods are shown in Figure 6. When the virtual robot began learning, only 15% of the periods were successful. After a few million periods, the percentage of success was greater than 80%.

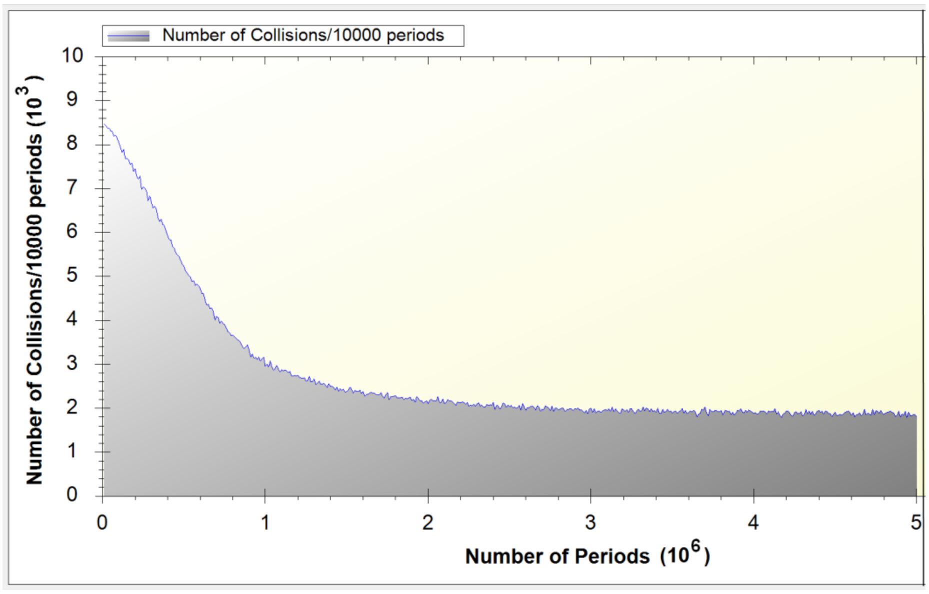

Fig. 5. Number of collisions between the virtual robot and dynamic obstacle per 10,000 periods

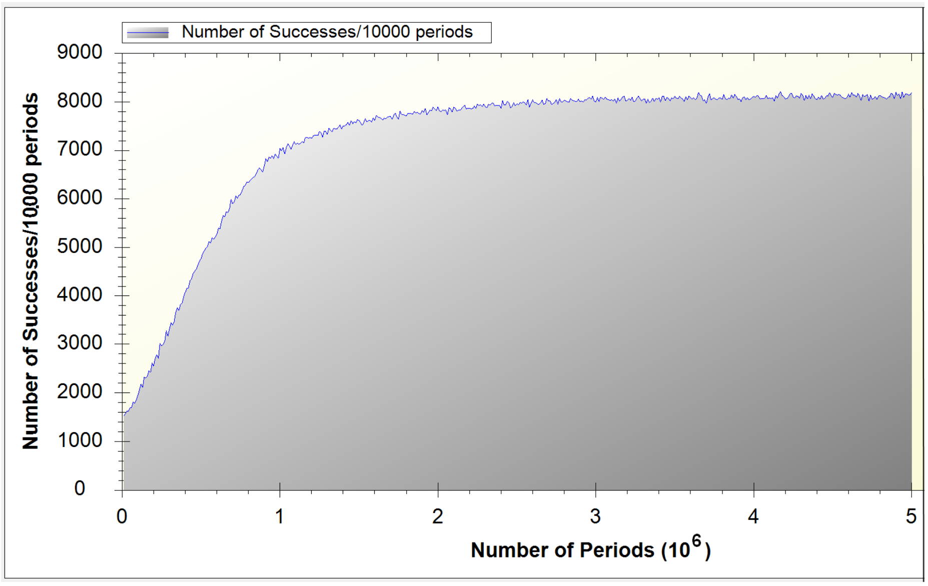

Fig. 6. Number of successes of the virtual robot per 10,000 periods

5 Conclusion

In this paper, we presented a novel approach to making a robot more intelligent. A robot is typically intended to provide some household function, such as fetching objects for a human. An effective robot must automatically avoid dynamic obstacles while moving. To achieve this objective, reinforcement learning cannot be directly applied to a real robot because it would be immensely time-intensive and the robot's physical ability is limited.

To address the above issues, we applied reinforcement learning to a robot in a virtual environment, which increased the learning speed. The virtual robot became more intelligent after each period. At the start of learning, the virtual robot could only randomly move because it had no experience. After learning through a few million periods, the virtual robot selected in each step the optimal path for avoiding the dynamic obstacle. When we change the grid for calculating the positions of all objects more finely. To this end, however, a larger number of learning periods would be required, and the learning time would considerably increase. Nevertheless, at times, the path chosen by the virtual robot was not the shortest one. We saved the data from the Q table into a database for application to a real environment. In future research, we intend to improve the learning to increase the percentage of successful periods and use more than one obstacle.

Acknowledgments. This work was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A2009148).

References

[1] Zhang, Z., Zhao, Z.: A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm. International Journal of Smart Home (2014)

[2] Zou, X., Ge, B., Sun, P.: Improved Genetic Algorithm for Dynamic Path Planning. International Journal of Information and Computer Science (2012)

[3] Achour, N., Chaalal, M.: Mobile Robots Path Planning using Genetic Algorithms. In: ICAS 2011: The Seventh International Conference on Autonomic and Autonomous Systems (2011)

[4] Sung, Y., Cho, S., Um, K., Jeong, Y., Fong, S., Cho, K.: Human-Robot Interaction Learning using Demonstration-based Learning and Q-learning in a Pervasive Sensing Environment. International Journal of Distributed Sensor Networks (2014)

[5] Sung, Y., Ahn, E., Cho, K.: Q-learning Reward Propagation Method for Reducing the Transmission Power of Sensor Nodes in Wireless Sensor Networks. Wireless Personal Communications (2013)

[6] Even-Dar, E., Mansour, Y.: Learning Rates for Q-learning. Journal of Machine Learning Research 5 (2003)

[7] Smart, W.D., Kaelbling, L.P.: Practical reinforcement learning in continuous spaces. In: Proceedings of ICML 2000, pp. 903–910 (2000)

[8] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4 (1996)