HOKKAIDO UNIVERSITY

Title	Effect of Viewing Directions on Deep Reinforcement Learning in 3D Virtual Environment Minecraft
Author(s)	Matsui, Taiju; Oyama, Satoshi; Kurihara, Masahito
Citation	PRIMA 2018: Principles and Practice of Multi-Agent Systems. PRIMA 2018. Lecture Notes in Computer Science (LNCS), vol 11224. Springer, Cham, ISBN: 978-3-030-03097-1, ISBN: 978-3-030-03098-8 (eBook), 527-534 https://doi.org/10.1007/978-3-030-03098-8_38
Issue Date	2018
Doc URL	http://hdl.handle.net/2115/72064
Rights	The final publication is available at Springer via https://dx.doi.org/10.1007/978-3-030-03098-8_38
Type	proceedings (author version)
File Information	prima2018.pdf

Instructions for use

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

Effect of Viewing Directions on Deep Reinforcement Learning in 3D Virtual Environment *Minecraft*

Taiju Matsui[1], Satoshi Oyama[12], and Masahito Kurihara[1]

[1] Graduate School of Information Science and Technology, Hokkaido University
[2] Global Institution for Collaborative Research and Education, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan

Abstract. Deep reinforcement learning, which has recently attracted the interest of AI researchers, combines deep neural networks (DNNs) and reinforcement learning (RL). By approximating a function in RL with a DNN, it enables an agent to learn in a complex environment represented by low-level features such as the pixels used in a 3D video game. However, learning from low-level features is sometimes problematic. For example, a small difference in input pixels results in completely different behaviors of an agent. In this study, as an example of such problems, we focus on the viewing directions of an agent in a 3D virtual environment (*Minecraft*) and analyze their effect on the efficiency of deep reinforcement learning.

1 Introduction

Recent developments in deep neural networks (DNNs) have enhanced the capability of DNNs to process high-dimensional data and to serve as vision in autonomous agents. This progress has made it possible for artificial intelligence (AI) to learn behaviors in video games directly from the screen images, which is called visual learning. In particular, Deep Q-Network (DQN) [3] [4], an algorithm proposed by DeepMind for Google, outperformed a human expert player in Atari 2600 games with 2D image data. In 2016, Google DeepMind proposed an asynchronous method [2]. One of the algorithms based on this method, called asynchronous advantage actor-critic (A3C), has surpassed all existing methods in efficiency and gain score.

Deep reinforcement learning (DRL) algorithms combine deep learning (DL) and reinforcement learning (RL). In DL, image data is used to define the representation of states, while RL is used for approximating the outputs of the DNN to outputs of RL. But image data are too complex to use as a representation. For example, image data expressed using RGB has a $\langle \text{height} \rangle \times \langle \text{width} \rangle \times \langle 3(\text{RGB}) \rangle \times \langle 256(0 \sim 255) \rangle$ pattern. However, in most RL environments, visual information available to agents is restricted, which very much complicates the process of representation learning. In the case of learning from first-person view image data

in a 3D virtual environment, agents have limited information about the environment and sometimes misunderstand their states. Furthermore, when pixels change color due to a change in the sight direction, learning from such image data becomes unstable. In order to acquire best action from the first-person views in a 3D environment, an algorithm that can handle this complexity is needed. In this study, as a preparation for developing such algorithm, we investigate differences in learning processes and acquired behavior in relation to the elevation angle of the agent's view when playing *Minecraft*.

The viewing direction is deeply related to the important psychological state of humans and animals such as attention and curiosity. It is also an important means of communication among them. We think controlling the viewing direction is an essential element to build autonomous agents and multiagent systems that operate in complex environments.

2 Related Work

This section describes the research related to deep reinforcement learning and the environment of *Minecraft*.

2.1 Deep Reinforcement Learning

Deep reinforcement learning algorithms approximate policy $\pi(s, a; \theta)$, value function $V(s; \theta)$, $Q(s, a; \theta)$, or both of them with the outputs of deep learning, where s, a, and θ represent state, action, and DNN parameters, respectively. These algorithms combine deep learning and reinforcement learning and use image data for learning specific actions. However, using non-linear approximators such as neural networks decreases the robustness of reinforcement learning. It is well known that loss of robustness is caused by the correlation between data sorted by a time series and a policy or a value function. Two DRL methods, DQN and A3C, solve this problem in their own way.

Deep Q-Network DQNs use two methods for stability. One of them is called experience replay in which tuples, sets of [state, action, reward, next state], are saved to experience replay memory for the last T steps of exploration. The algorithm then learns by mini batches randomly sampled from the experience replay memory every few explorations. Although this method reduces correlations, it has two weak points. One is that the on-policy algorithms are not applicable because the data from exploration is based on a previous policy. The other is that the size of the experience replay memory tends to be large.

The second method is called fixed target network. In DQNs, there is a learning phase after every few explorations. During that phase, a little changes in value functions update the policy greatly, causing loss of robustness of the algorithm. To avoid this problem, the DNN parameters θ are fixed to θ^- during the learning phase. This contrivance is called fixed target network.

Asynchronous Advantage Actor-Critic In A3C, the algorithm contains parallel threads for collecting data by exploration to reduce correlation, and the threads run asynchronously. Since asynchronous threads run apart from each other, the collected data is the same as the randomly sampled data. Each threads computes in the following order.

1. Copy from global network parameter θ to thread's own local network parameter θ^-.
2. Explore the learning environment based on own parameter, and compute the gradient $d\theta$ from the loss function $L(\theta)$. $L(\theta)$ is computed from temporal difference error.
3. After several explorations, send gradient $d\theta$ to global network and update parameter by gradient decent.
4. Return to step 1. and repeat the process until T_{max}.

Here, copying parameter at 1 has the same effect as the fixed-target network of the DQN. These parameters are optimized by gradient decent of the REINFORCE algorithm [6] and use Advantage function for estimating a current state value in this time. Advantage function is as follows:

$$A_t = \sum_{i=0}^{k-1} \gamma^i r_{t+1} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v) \tag{1}$$

Owning Advantage function, the algorithm could use more future data than temporal difference error calculated by Bellman equation. Using Advantage function, the gradient of parameters are calculated by the following formula.

$$d\theta \leftarrow d\theta + \alpha \nabla_\theta \log \pi(s_t, a_t; \theta) A_t + \beta \nabla_\theta H(\pi(s_t; \theta)) \tag{2}$$

$$d\theta_v \leftarrow d\theta_v + \alpha \nabla_{\theta_v} A_t^2 \tag{3}$$

Here, third term of (2) is the entropy term for stochastic normalization.

2.2 Minecraft (Project Malmo)

Minecraft is a videogame sold by Microsoft in a genre called sandbox. Sandbox games have no forced mission, so the playing styles and the environment are flexible. In *Minecraft*, the environment is made of various cubes, but is similar to the real world we live in. The environment has the following features.

- Agents are affected by gravity, but cubes placed in the environment are not, with some exception.
- Enemies are spawned under certain circumstances.
- There is a concept of time. Time affects Brightness of the environment.
- There is a concept of biome and different terrains; various blocks and creatures are existed in each biome.

The agents can perform the following actions.

- Destroy and get almost all blocks by continuous attack. Some blocks require a special tool for destroying.
- Relocate the conquered blocks to the adjacent spaces.
- Combine blocks or items and create new blocks or items, called crafting.
- Use the newly crafted items.

Microsoft Research released a platform called *Project Malmo* [1] at GitHub in 2016. It's aim is to provide a testbed/sandbox for developing AI agents within *Minecraft*. This release provides a framework for interactions between the agent and the environment and facilitates the development of the *Minecraft* experimental environment.

3 Experiment in Minecraft

3.1 Purpose of the experiment

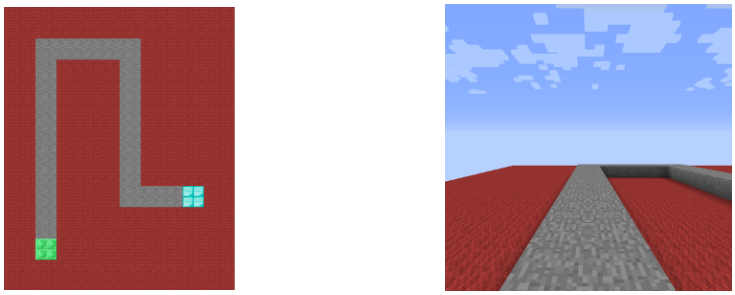
The purpose of the experiment is to investigate the differences in the learning process and acquired behavior in relation to the elevation angle of the agent's view in *Minecraft*.

3.2 Configuration of learning task

Here, we describe the configuration of the task using Project Malmo. The outline of the learning task is to "advance without falling down the road and without branching", as a simple problem is preferred for investigating the effect of changes in the elevation angle. An episode ends when the agent reaches the goal (the end of the road), or falls down the road, or when time is up. The score is calculated and recorded by the advantage function. The environment is composed of a road having several corners, and the agent gets RGB images ($84 \times 84 \times 3$) as a representation of the current state (Fig. 1). The agent can take actions that combine"advance, turn left, turn right" with the exception of taking no action. The reward for the agent is designed as +1.0 when the goal is reached, -1.0 when the agent falls down the road or when time is up, and +0.1 when the agent proceeds down the road for 1 block in *Minecraft* in order to avoid acquiring the behavior of the agent turning in the same spot. In the experiment, we set the elevation angle of the agents' view to $0°$, $-30°$, and $-45°$, then learn during 2 million steps (a step is about 0.1 second) (Fig. 2). Because the information about the road is different from each angle, it is expected that the results of the experiment will also be different.

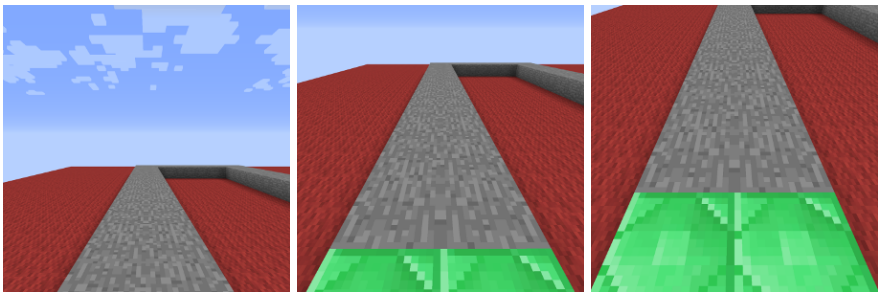
3.3 Experimental results

Difference in learning process Fig.3 shows the average score of every 10,000 steps in the learning process of 2 million steps. It indicates that the more information about the road is caught in the agent's view, the earlier the agent acquires suitable behavior.



Bird's - eye view of the environment **Agent's view**

Fig. 1. Environment using for learning



angle 0° **angle** −30° **angle** −45°

Fig. 2. Each agent's view at different elevation angles

Difference in acquired behavior After the learning process, we collect the data on the average scores, average advanced degrees and the goal arrival rates from 1,000 episodes. The results are presented in Fig. 4. It also shows the significance of the information about the road in the agent's view. These results are similar to the results of the previous experiments 3.3.

The points of falling down The points where the agents fall down are shown in Fig. 5. The agents with a view of 0° fall down at the beginning of the road, in its straight part, because the agent cannot obtain useful information enough to pass. On the contrary, the agents with a view of −30° and −45° can obtain information that helps them reach the goal. However, they also fall down at a certain point. This is because the actions are discrete and binarized to "do" or "not do", which sometimes cause the agents to lose sight at the corners of the road.

3.4 Discussion

In the experiment of learning process, the learning process is affected by the agents' view during the time required for learning. In the experiment 3.3, the

acquired behavior is also affected by the view. This proves that the amount of information necessary to achieve the learning task is extremely important for exploration in a 3D virtual environment by the first-person view. Furthermore, in the experiment 3.3, it is necessary not only to give the appropriate view but also to make the actions continuous. Therefore, we consider that the control of the agents' view is an important issue for exploration in a 3D virtual environment by the first-person view.

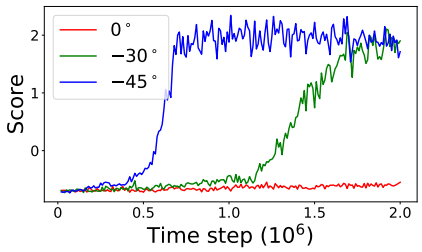


Fig. 3. Average score every 10,000 steps during learning process

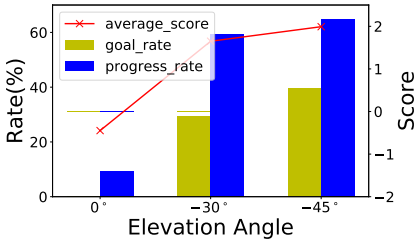


Fig. 4. Average score, progress rate, and goal rate by 1,000 episodes

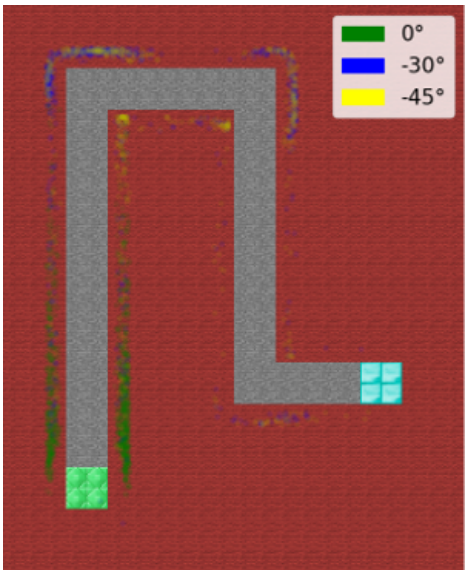


Fig. 5. Difference in falling position

4 Additional experiment

We prepared a more difficult task for evaluating the difficulty of learning the control of agent's view direction. In this new task, the basic rules are the same as in the previous task, but the road is randomly generated with a fixed length. The road does not have any branches or loops, and its terminus is a tower of specific blocks. Fig. 6 is an example of the generated road of such environment. We compared the learning process and acquired behavior with the agent controlling the view direction (the sight-controlling agent), in addition to the agents with fixed angles in our previous experiment.

Fig. 6. Environment for additional experiment

4.1 Result of additional experiment

We compared difference in learning process and acquired behavior. Fig. 7 shows the average scores of the learning process and Fig. 8 shows the evaluations of each agent. Both of them indicate the difficulty of controlling agent's sight direction. In Fig. 7, the sight-controlling agent could not outperform other agents even if it was trained for longer than the others. In Fig. 8, the sight-controlling agent shows the results similar to the agent with a view of 0° and it indicates the training was not successful.

4.2 Discussion on additional experiment

In the additional experiment, the difficulty of controlling agent's sight direction was indicated. It might be caused by vast and discrete action space. The sight-controlling agent has 17 patterns of actions, which are about 3 times as many as the actions of other agents. It can affect the results as the curse of dimensionality. Hence, to learn complex behavior to control site direction, we improve the learning algorithm so that it can solve the problem of the large actions space.

5 Conclusion and Future work

In this paper, we explained the methods of Deep Reinforcement Learning, then investigated the effects of directions of agents' view in 3D virtual environment for the task acquiring behavior using one of the methods, A3C.

As a future work, we suggested controlling agents' view direction and presented the results of the preliminary experiment. Since A3C was proposed in 2016, the methods with various additions to A3C is proposed. One of these methods [5] used curiosity by self-supervised prediction. It calculates internal reward from input images apart from external reward given as reply of action by the environment. We consider concept of internal reward is useful for exploration in 3D virtual environment by first person view. Therefore, we set a goal for proposing an algorithm to control agent's view direction.

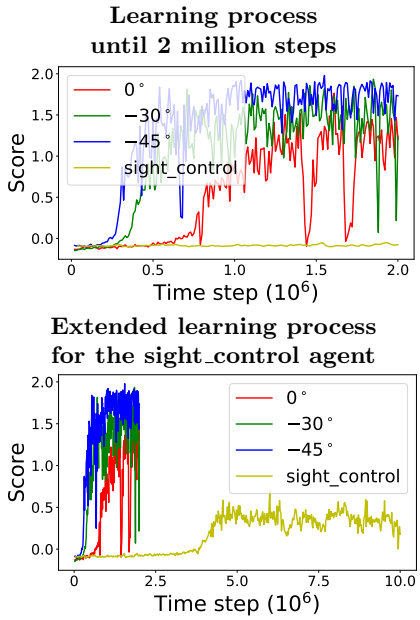


Fig. 7. Average score every 10,000 steps during learning process in additional experiment

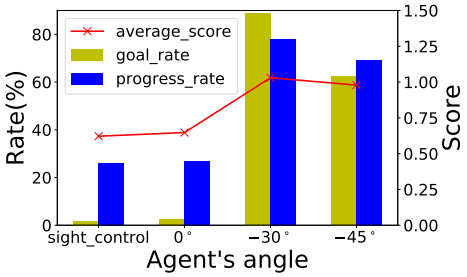


Fig. 8. Average score, progress rate, and goal rate by 1,000 episodes in additional experiment

References

1. Johnson, M., Hofmann, K., Hutton, T., Bignell, D.: The Malmo platform for artificial intelligence experimentation. In: IJCAI. pp. 4246–4247 (2016)
2. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML. pp. 1928–1937 (2016)
3. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning. In: NIPS Deep Learning Workshop (2013)
4. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
5. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: ICML. pp. 2778–2787 (2017)
6. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3), 229–256 (1992)