JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 21, No. 8, pp. 1489-1494, Aug. 2020

Check for updates

3차원 물리기반 시뮬레이션에서의 비정상적인 동작 해결

홍 기 진[1] · 김 영 봉[2*]
[1]부경대학교 IT융합응용공학과 박사과정
[2]부경대학교 IT융합응용공학과 교수

Resolution of Abnormal Behaviors in 3D Physically-Based Simulation

Gi-Jin Hong[1] · Young-Bong Kim[2*]

[1]Doctor's Course, Department of IT Convergence and Application Engineering, Pukyong National University
[2*]Professor, Department of IT Convergence and Application Engineering, Pukyong National University, Busan 48513, Korea

[요 약]

최근 심층인공신경망 기술의 발전은 전문 지식(domain knoelwdge) 없이도 다차원 정보처리를 가능하게 한다. 그 중에서도 강화학습은 자율주행 차, 드론과 같은 3차원 환경과의 상호작용을 통한 문제 해결 분야에서 활발히 연구되어왔다. 강화학습을 위한 물리 기반 가상 시뮬레이터는 부동 소수점 계산이나 수치 적분상의 오차를 가지며, 이로 인해 현실 세계에서는 물리적으로 불가능한 의사결정자의 움직임을 학습하도록 하는 심각한 문제가 발생할 수도 있다. 따라서 본 연구에서는 잘못된 데이터의 학습과 저장으로 인해 발생하는 강화학습의 문제를 해결하기 위하여 가치 최적화 강화학습 기반의 경험 데이터와 행동 정책 신경망 가중치 간의 관계를 이용하여 데이터를 이전의 상태로 복구하기 위한 구조와 동작 방법을 제안한다.

[Abstract]

The recent development of deep artificial neural network technology enables high-dimensional information processing without domain knowledge. Among them, reinforcement learning has been actively researched in the field of problem solving through interaction with 3D environments such as autonomous vehicles and drones. In many physics-based virtual simulators, numerical calculations errors in floating point or integrations may trigger errors in real engines and bugs in various program. Because of those problems, it can cause serious problems that the decision-maker(agent) learns physically impossible movements. Therefore, we propose a structure and operation method to restore experience data and action policy neural network weights to solve reinforcement learning problems caused by learning and storing incorrect data.

색인어 : 경험 데이터, 강화 학습, 복구, 가상 시뮬레이터, 가중치 값

Key word : Experience Data, Reinforcement Learning, Restoration, Virtual Simulator, Weight Value

http://dx.doi.org/10.9728/dcs.2020.21.8.1489

Received 22 June 2020; **Revised** 17 July 2020
Accepted 31 July 2020

***Corresponding Author; Young-Bong Kim**

Tel: +82-051-629-6248
E-mail: ybkim@pknu.ac.kr

Ⅰ. Introduction

Currently, as remarkable advancement has been made in hardware, the spotlight is focused on machine-learning areas based on deep artificial neural networks, requiring high computation speed and large amounts of data. Recently, many studies have been performed on machine-learning methods using deep artificial neural networks. As commercial models, they have gradually been used in wider applications. Furthermore, recent reinforcement learning-related studies have produced successful results, and subsequently, they have been frequently applied to applications such as resource management of server computers in network communication, and problems requiring thinking like humans, such as the game of go [1], [2].

Reinforcement learning is a machine learning method, whereby a decision-maker(agent), defined in a given environment, recognizes the current situation and collects information, and based on the collected information, creates action policies to solve the problem. In the past, because of hardware limitations of small memory size and slow computation speed, reinforcement learning was applied limitedly in areas having little information to be collected and with only a few variables that could change the situation, such as the maze game and tic-tac-toe [3].

However, as significant advancement has been made in hardware, and fundamental studies have been performed over many years, reinforcement learning has demonstrated successful results in areas requiring large amounts of learning data, such as 3D games and go(baduk), as shown in Fig. 1. Based on these studies, additional technologies that solve problems by interacting with 3D environments, such as autonomous vehicles and drones, are actively studied now [4], [5].

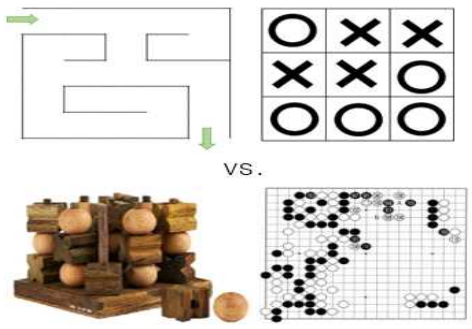

그림 1. 단순한 환경 vs. 복잡한 환경
Fig. 1. Simple Environment vs. Complex Environment

Like other machine learning methods, the reinforcement learning method usually requires repetitive experiments. However, in the real world, when reinforcement learning is performed with a robot as the decision-maker, the cost of the experiment is high, and problems such as accidents or damage may occur. Consequently, using a virtual simulator for reinforcement learning has become a trend.

Unlike the simulators of chess and go, which can discern whether the current state is realistically possible, physics-based simulators can result in various errors. For example, a robot may go through a wall due to a fast time step. This is the result of an error by the physical engine, which is produced due to floating-point error and numerical integration error. The problems of the simulator lead to physically impossible movements by the decision-maker in the real world, and the storing of incorrect experience data. Furthermore, when a high reward value is derived from the incorrect experience data, a serious problem of learning occurs: the action policy produces an incorrect result.

Therefore, this study designs a simulator structure of reinforcement learning, which can restore the previous state of experience data and action policy. This can resolve the aforementioned issues of reinforcement learning induced by incorrect data collection. This study prevents and resolves incorrect reinforcement learning through the relationship between the experience data and action policy neural network's weight, based on the reinforcement learning of value optimization.

Ⅱ. Related Works

2-1 Reinforcement Learning

Reinforcement learning is an area of machine learning, where software agents collect data through behaviors directly in a given environment and create action policies to solve problems through the collected data. This technique is suitable for problems in which learning data is difficult to obtain in advance and when continuous decision-making is required. Traditionally, it has been used in game artificial intelligence or robotics.

Fig. 2 shows the flow of reinforced learning. The decision-maker(agent) decides an action according to the action policy using the given state as an input, and receives the reward and next state values, based on the interaction with the environment [6], [7]. Afterwards, the action policy is updated using the reward value received as a result of the action and reinforcement learning algorithm. Until one episode is completed, the next state value is used as the given state, and the above process is repeated. Here, the values of five variables, state, action, reward, next state, and episode done, are grouped together,

and named ‘experience data’. Reinforcement learning has learned with rewards from interactions with the environment, rather than with well-classified data. So, it takes a lot of trial and error until the agent finds the best action under current situation. In particular, behaviors learned due to incorrect rewards often lead to serious problems.

Recently, a deep artificial neural network has been combined with the reinforcement learning methods, taking this technology a significant step forward. The latest methods are divided into value optimization and policy optimization, according to the target of optimization.

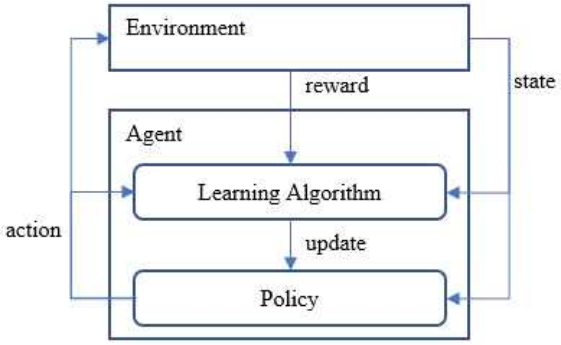

그림 2. 강화학습 흐름도
Fig. 2. Reinforcement Learning Flow

2-2 Deep Q-Network

Deep Q-Network (DQN) [8], [9] is one of the value optimization methods made at Google DeepMind, and a well-known algorithm that has become famous after displaying better performance than humans in the Atari2600 game. DQN is based on Q-Learning and allows deep artificial neural networks to take over the role of Q-functions.

Action policies are learned by using the collected experience data as learning data. In the case of a simulator, local overfitting sometimes occurs when the experience data is used as a training set as-is, because change is small between adjacent frames. Learning data is extracted randomly using the experience replay method, and then a batch training set is created in order to resolve the strong continuity between the experience data. The replay memory makes it possible to learn with less correlation and uniformly distributed data. As shown in Fig. 3, the experience data is saved in the replay memory, and when the space is all filled up, old experience data is taken out, and new experience data is added in, following the first-in, first-out (FIFO) scheme. We want to restore the previous state using the experience data removed from the replay memory.

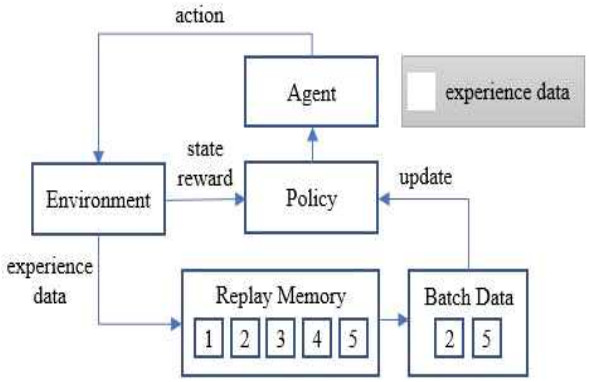

그림 3. 심층 Q-네트워크 흐름도
Fig. 3. Deep Q-Network Learning Flow

2-3 OpenAI Gym

OpenAI Gym [10], [11] is a reinforcement learning platform that provides various environments for the development of general-purpose artificial intelligence. It is widely used in a majority of reinforcement learning studies to check and verify the validity of studies and test the algorithms. OpenAI Gym provides reinforcement learning environments such as a 2D physical environment through Box2D, including home video games like Atari 2600, and a 3D physical environment using Mujoco [12]. OpenAI Gym consistently provides the interfaces of all environments, so that the reinforcement learning algorithm can be easily changed. The basic interface consists of the following,

• Reset: is an interface that resets the reinforcement learning environment, and the initial state values are returned as soon as it is called.

• Render: outputs the current environment on the screen.

• Step: receives the action value as a parameter value, and moves the agent through the action and returns the Next State, Reward, and Episode Done.

However, OpenAI Gym has some problems in implementing real 3D physics and screen output objects directly through high abstraction.

Ⅲ. The Proposed Method

Reinforcement learning platforms such as OpenAI Gym do not support the restoration of experience data when weight values cause incorrect behavior. Therefore, we need to define the learning situation in a 3D physics-based simulator which reinforcement learning algorithms can be applied, and also to propose a 3D simulation system to restore into previous situation when a weight value causes incorrect behavior.

3-1 Management of Experience Data

Fig. 4 shows a diagram for the experience data management proposed in this paper. In the figure, “Environment” refers to the virtual simulator where the problem will be solved. “Policy Script” usually includes the action policies and the value optimization-based deep reinforcement learning algorithm for learning the action policies. “Experiences” of Policy Script correspond to the replay memory of DQN and are logically divided into two, while maintaining a queue data structure. “Policy weight” indicates the weight value of the learning policy neural network. “Policy” refers to the action policy network, “exp_count” is the number of experience data, and “max_count” is the maximum number of experience data.

“Storage” refers to the storage where the restoration values are kept. The “Experiences” stores the old values coming out from the Policy Script’s experiences because they are outdated, and has the FIFO structure like a replay memory. Logically, there are two or more regions. “Policy weights” facilitates restoration of the weight value, by storing the weight value prior to update via learning in the Policy Script. “Exp_count” and “max_count” have the same roles as those of Policy Script. “Block_size” is the size of a block of experiences divided logically, and refers to the amount of experience data that can be saved.

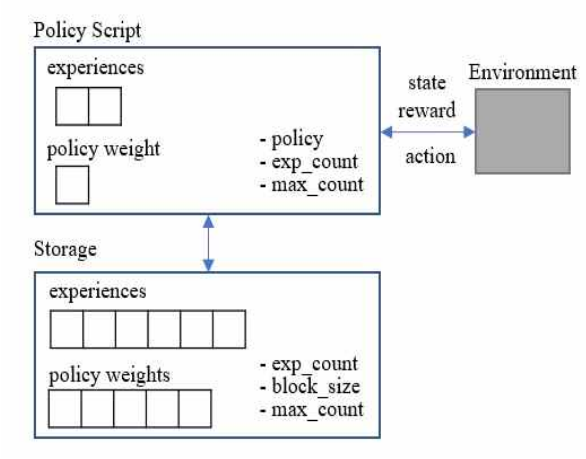

그림 4. 경험 데이터 관리 모델
Fig. 4. Experience Data Managing Diagram

3-2 Storing Experience Data and Weight

As the simulator episodes of reinforcement learning are processed, data is stored in the “experiences” of Policy Script. Furthermore, when the experience data of Policy Script exceeds max_count, the values are saved in the Storage, as shown in Fig. 5. Prior to executing learning based on the experience data, the value of Policy Script’s policy weight is saved in the Storage’s policy weights. Then, the weight is updated through the reinforcement learning algorithm. Afterwards, the old experience data is taken out and stored in “experiences” of Storage. This is called an experience data block.

Suppose the experience data block and the weight value stored most recently in the Storage are Bt and Wt; then, the value of Wt is a value learned by making the batch data from the experience data of Bt-1 and Bt.

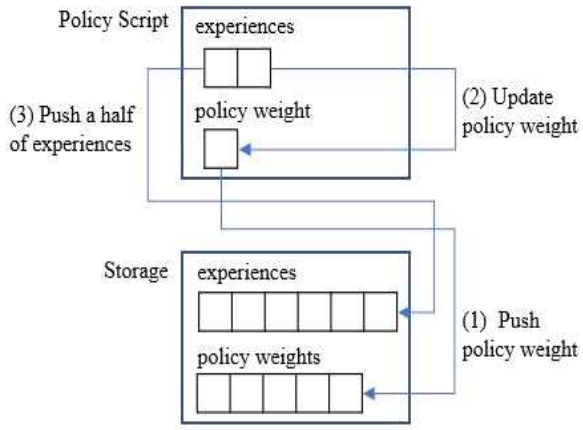

그림 5. 경험데이터와 가중치 값의 저장
Fig. 5. Storing Experience Data and Policy Weight Diagram

3-3 Restoring Experience Data and Weight

Based on the characteristics of Bt and Wt mentioned in the previous section, the Policy Script’s policy weights and experience data can be restored when a physically incorrect action by the decision-maker is observed on the simulator, as shown in Fig. 6. First, the frame N, the desired time point of restoration, is received as an input. Next, the current value of exp_count stored in the Policy Script is subtracted. Suppose the subtracted value is R. Then, the action policy network’s weight, and the experience data, are restored using one of the following three methods, according to the stored value of R.

• When R is a positive number and greater than or equal to block_size:

There is no data exchange with Storage. Also, the experience data of Policy Script is removed, according to the input N. Because the weight value is not changed, an incorrect action by the decision-maker may occur again.

• When R is a positive number and less than block_size:

The experience data of the Policy Script is removed, according to the user input N. Subsequently, the experience block Bt stored in the Storage recently, is added at the end of experiences. Next, the

value of policy weight is changed to Wt saved recently in the Storage.

• When R is a negative number:

After changing the value sign of R to the positive sign, it is divided by the block size. Suppose the quotient excluding the remainder is q; then, first, the Policy Script's experiences are all emptied. Subsequently, the experience data removed earlier by as many as the positive value of R from the total experiences of block B(t-q-1) and a block B(t-q) in the Storage, are stored in the Policy Script's experiences. Next, the value of policy weight is updated with W(t-q-1). Fig. 6 shows a diagram for the above processes.

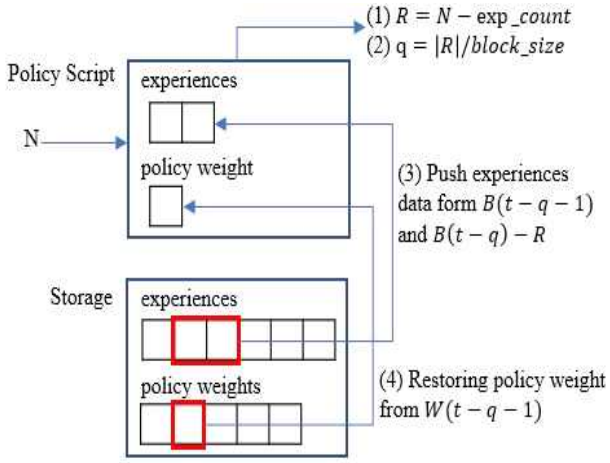

그림 6. 경험 데이터와 가중치의 복구

Fig. 6. Restoring Experience Data and Policy Weight Diagram

Ⅳ. Conclusion

Various studies have demonstrated that reinforcement learning can be applied to complex environments. Even when the states are difficult to define, like those of the real world. Accordingly, studies have been actively carried out for a decision-maker(agent) interacting with a variety of 3D environments. However, because of the cost issue and the risk of accidents, studies are usually performed using a simulator, rather than in the real world. Because a simulator expresses a real world's physical phenomenon, based on discretization using a short time step, it is not free from the floating-point error and numerical integration error. Consequently, physically impossible actions are sometimes displayed. Such a result leads to a serious risk when the action policy is learned as-is. This study has proposed an architecture and operating method to restore the weight value of the learning policy neural network when it leads to an incorrect action. The architecture and operating method are founded on restoring data using the relationship between the experience data and action policy neural network's weights, based on the reinforcement learning of value optimization.

In future works, we try to implement and test a 3D physics-based simulator that applies the Policy Script in this paper. We also add the data restoration function and visualization tool to the existing reinforcement learning platform. Through these upgrades, we can expect the stable reinforcement learning platform that help to overcome the wrong solutions such as abnormal stop of the simulator.

Acknowledgment

This work was supported by a Research Grant of Pukyong National University(2019 year).

References

[1] X. Dutreilh, S. Kirgizov, and O. Melekhova, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow", ICAS, pp. 67-74, 2011.

[2] D. Silver, J. Schrittwieser, and K. Simonyan, "Mastering the game of Go without human knowledge", Nature, Vol. 550.7676, pp. 354–359, 2017.

[3] P. Dayan, and G.E. Hinton, "Feudal reinforcement learning", In Advances in neural information processing systems, pp. 71-278, 1993.

[4] In Yong Hwang, Soo Hyun Wang, Seung Min No, and Doo Seop Eom, "Comparison and analysis of autonomous flight drones algorithm based on deep reinforcement learning", in Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 630-631, 2018.

[5] Hongsuk Yi, Eunsoo Park, and Seungil Kim, "Deep Reinforcement Learning for Autonomous Vehicle Driving", in Proceedings of Symposium of the Korean Information Science Society, pp. 784-786, 2017.

[6] A. Eitan. Constrained Markov decision processes, CRC Press, Boca Raton, Florida, 1999.

[7] Y. Li, "Deep Reinforcement Learning : An Overview", https://arxiv.org/abs/1701.07274, 2017.

[8] V. Mnih Silver, K, Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, J. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D.

Hassabis, “Human-level control through deep reinforcement learning”, Nature, Vol. 518, pp. 529-533, 2015.
[9] Christopher Watkins, and Peter Dayan, “Q-learning”, Machine learning, Vol. 8, pp. 279-292, 1992
[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W.Zaremba, “OpenAI Gym”, https://arxiv.org/abs/1606.01540, 2016.
[11] M. G. Bellemare, “The arcade learning environment: An evaluation platform for general agents”, Journal of Artificial Intelligence Research, Vol 47, pp. 253-279, 2013
[12] Emanuel Todorov, Tom Erez, and Yuval Tassa, “Mujoco: A physics engine for model-based control”, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026-5033, 2012

홍기진(GiJin Hong)

2000년 : 부경대학교 교육대학원 (교육학석사)
2005년 : 부경대학교 전자계산학과 박사수료

2020년~현 재 : 부경대학교 IT융합응용공학과 수료 후 연구생
※관심분야 : 컴퓨터 그래픽스, 3D 컴퓨터 시뮬레이션, 영상처리, 인공지능 등

김영봉(YoungBong Kim)

1989년 : 한국과학기술원 전산학과 (공학석사)
1994년 : 한국과학기술원 전산학과 (공학박사)

1994년~1995년 : 삼성전자 정보기술연구소 선임연구원
1995년~현 재 : 부경대학교 IT융합응용공학과 교수
※관심분야 : 컴퓨터 그래픽스, 3D 컴퓨터 시뮬레이션, 영상처리, 인공지능 등