Computer Science and Information Systems 18(2):461–479 https://doi.org/10.2298/CSIS200212005H

Enhanced Image Preprocessing Method for an Autonomous Vehicle Agent System

Kaisi Huang[1], Mingyun Wen[2], Jisun Park[3], Yunsick Sung[4], Jong Hyuk Park[5], and Kyungeun Cho[6,*]

[1]Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea
hkshks0825@dongguk.edu
[2]Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea
wmy_ncut@dongguk.edu
[3]Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea
jisun@dongguk.edu
[4]Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea
sung@mme.dongguk.edu
[5]Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), 232 Gongneung-ro, Nowon-gu, Seoul, 01811, Korea
jhpark1@seoultech.ac.kr
[6]Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea
cke@dongguk.edu (corresponding author)

Abstract. Excessive training time is a major issue face when training autonomous vehicle agents with neural networks by using images as input. This paper proposes a deep time-economical Q network (DQN) input image preprocessing method to train an autonomous vehicle agent in a virtual environment. The environmental information is extracted from the virtual environment. A top-view image of the entire environment is then redrawn according to the environmental information. During training of the DQN model, the top-view image is cropped to place the vehicle agent at the center of the cropped image. The current frame top-view image is combined with the images from the previous two training iterations. The DQN model use this combined image as input. The experimental results indicate higher performance and shorter training time for the DQN model trained with the preprocessed images compared with that trained without preprocessing.

Keywords: Image preprocessing, Reinforcement learning, Deep Q learning.

* Corresponding author

1. Introduction

Recently, for predicting traffic and training novice drivers, transportation simulation systems have been utilized. It is necessary for such systems to be designed with as much similarity to the real world as possible. For example, autonomous driving vehicle simulations, which can advance research in automotive safety, require highly realistic transportation simulation systems.

Recently, human-driven agent systems have been issued for automatic systems. Most approaches for learning autonomous driving with neural networks use images as input. Neural networks are excellent image classifier algorithms, that perform well when large amounts of data are provided. Reinforcement learning [1] and supervised learning [2] utilize image input for model training.

Deshpande proposed a deep reinforcement learning approach to train autonomous driving vehicles [1]. This approach utilizes environment images, agent speeds, distances to road centers and angles of the heading vectors with respect to direction of the road as the network inputs. To reduce the redundant features that don't affect decision-making of network and make the training converge faster, image preprocessing is necessary.

To reduce the training time and achieve a higher performance training result, this paper proposes a method to preprocess input images for training an autonomous driving vehicle agent with a deep Q network (DQN). In the proposed method, all the information about the roads and junctions that are related to driving a vehicle agent in a virtual environment are first extracted and recorded in an environment information file. A whole top-view image of the virtual environment is redrawn and the vehicle agent is added to this image. This top-view image is then cropped according to the vehicle agent's surrounding environment and utilized to create an input image sequence for training the DQN. The outputs of the DQN are steering and acceleration, which are utilized for the control of the vehicle agent in the virtual environment. The virtual environment returns a reward for training the DQN model in each iteration.

In the following, Section 2 relevant previous studies are discussed. Section 3 presents the main concepts of the image preprocessing for the DQN. Section 4 describes the detailed implementation of the proposed method and the experimental results. Finally, Section 5 concludes the study and discusses future works.

2. Related work

2.1. Imitation learning

Behavioral cloning (BC) is the most common approach to imitation learning. The objective of BC is to learn the relationship between states and optimal behaviors as a supervised learning problem [4–6]. A convolutional neural network (CNN) is frequently used for BC. A CNN learns features automatically from demonstrations given a suitable dataset. Bojarski [2] introduced a system of self-driving cars using a CNN that automatically learns the internal features of input images that it never explicitly trains itself to detect. However, it requires a large dataset for training to guarantee that most states that may occur are covered.

Because of environmental restrictions in real environments, end-to-end control cannot be trained to handle all situations. A virtual simulation environment [17] has been used for training to overcome this problem. In this approach, a virtual environment similar to the real environment is constructed for the virtual simulation. Captured images and the control signals of a vehicle agent are collected. A CNN is trained based on the collected images and control signals, and then used to control a vehicle in the real environment.

2.2. Reinforcement learning

Reinforcement learning algorithms allow agents to learn how to behave differently in different situations. Its aim is to establish a policy that considers the situation and select actions that maximize a reward[7]. Reinforcement learning has been successfully applied to many different tasks such as playing relatively simple Atari games, mastering the relatively more complex game of Go, and controlling robots in real environments.

Inverse reinforcement learning is the most successful imitation learning approach [8,9]. This approach assumes that the behaviors that learners desire to imitate are generated by experts. This approach attempts to estimate a reward function to explain the behaviors of experts [14]. However, obtaining the reward function of inverse reinforcement learning is slow in terms of convergence speed.

A reinforcement learning based driving policy for autonomous road vehicles was proposed in [17]. The DQN is trained to control a vehicle by changing its heading, acceleration, and deceleration. The input of this DQN is a vector that is constructed from vehicle sensors and includes the longitudinal velocity and vehicle position in the lane. However, a virtual environment can occasionally be too complex to extract features for a network.

3. DQN-input image preprocessing approach

In this section, the proposed preprocessing method is described. Section 3.1 gives an overview of the proposed framework. Sections 3.2 and 3.3, describe how to extract the environmental information from a virtual environment and how to redraw the top-view image of the whole environment, respectively. Section 3.4 provides details on how the top-view image is cropped around a vehicle agent in the whole top-view image. Finally, the DQN model structure is described in Section 3.5.

3.1. Overview

The main process of the proposed method is shown in Fig. 1. During the environment information extraction phase, all information about roads and junctions related to a vehicle agent in a virtual environment are collected and written to an environment information file. Subsequently, during the environment redrawing phase, the information in this file is utilized to redraw the whole top-view image of the virtual environment using straight lines and arcs. The location and direction of the vehicle

agent in the virtual environment is added to the whole top-view image. During the top-view image cropping phase, the top-view image is cropped to place the vehicle agent in the center of the cropped image. In each training iteration, the cropped top-view image is utilized as an input to the DQN. The outputs of the DQN are the steering and acceleration of the vehicle agent. These two values are used to control the vehicle agent in the virtual environment, and the virtual environment returns a reward according to the status of the vehicle agent. The DQN model undergoes training in each iteration.

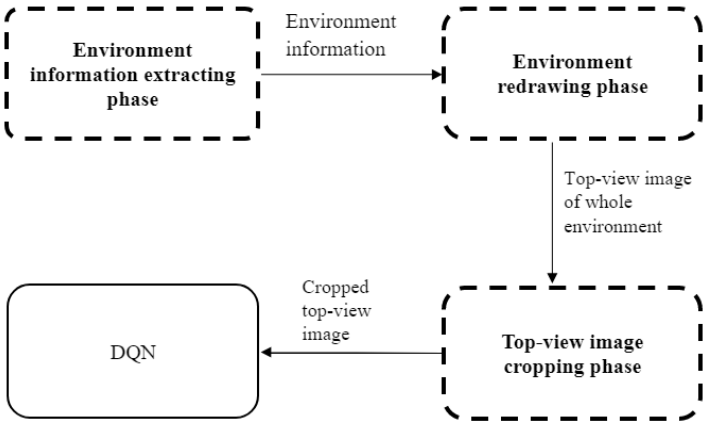

Fig. 1. Processes of the proposed method

3.2. Environmental information extraction

In a virtual environment, diverse types of information exist. The information on the roads and junctions are important for driving a vehicle agent. Roads and junctions comprise the map that cars should drive. One road can be connected to other roads or junctions, and one junction can be connected to multiple roads. All link information exists in the virtual environment. Therefore, these two types of information should be extracted.

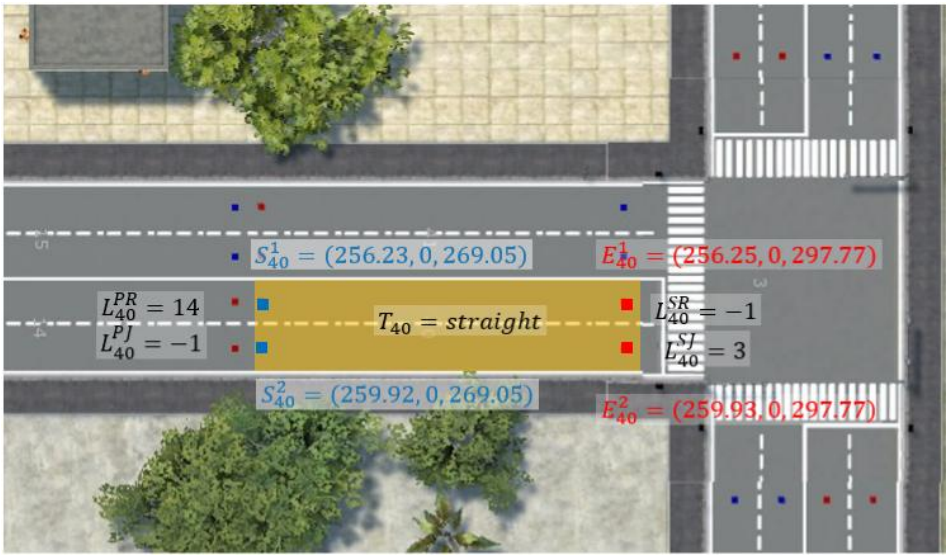

Fig. 2. Example of a 3D road

In this paper, the *i*-th three-dimensional (3D) road R_i is described using four elements [i, L_i, T_i, N_i], defined as follows: the road ID i, link information L_i, road type T_i, and lanes N_i . The link information L_i is expressed as [L_i^{SR}, L_i^{PR}, L_i^{SJ}, L_i^{PJ}], which consists of the successor road ID L_i^{SR}, predecessor road ID L_i^{PR}, successor junction IDL_i^{SJ} and predecessor junction ID L_i^{PJ}. The road type is either straight, corner or merging. The lanes N_i consists of a set of lanes, {N_i^1, N_i^2, ..., N_i^n, ...}, where n describes the order of the lanes. A lane N_i^n is denoted by[S_i^n, E_i^n]. Therefore, each lane contains two elements: start point S_i^n =(x_i^n, y_i^n, z_i^n) and end point E_i^n =(x_i^n, y_i^n, z_i^n).

Fig. 2 shows an example of a 3D road. RoadR_{40}, which is indicated by the yellow area is described byR_{40}=[*40*, L_{40}, *straight*, N_{40}] , where L_{40} =[*-1*, *14*, *3*, *-1*], N_{40} =[N_{40}^1, N_{40}^2],N_{40}^1 =[(*256.23*, *0*, *269.05*), (*256.25*, *0*, *297.77*)], and N_{40}^2 =[(*259.92*, *0*, *269.05*), (*256.25*, *0*, *297.77*)].

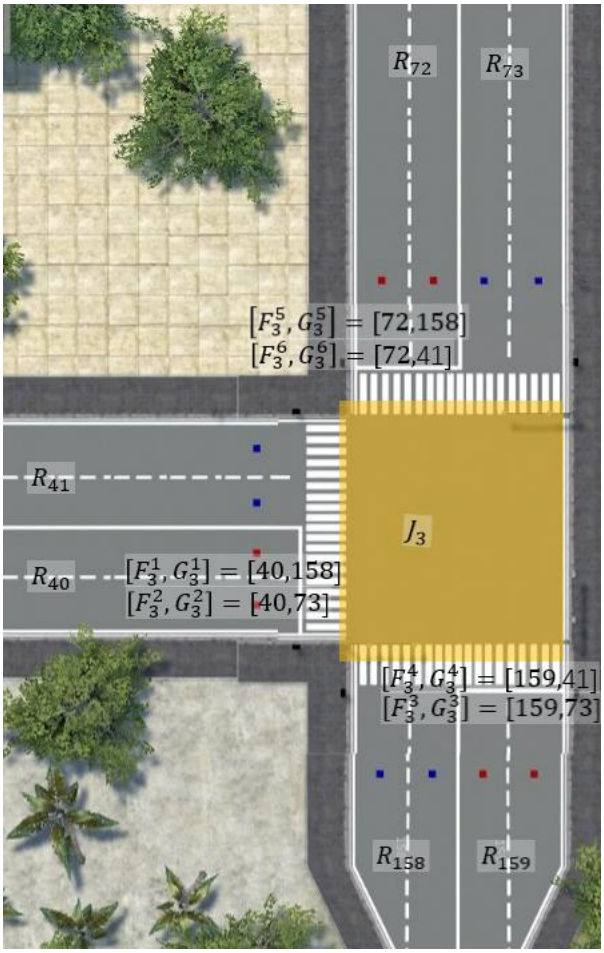

Fig. 3. Example of a 3D junction

The *i*-th 3D junction J_i is expressed as [*i*, C_i], where each 3D junction contains two elements: junction ID *i* , and connections C_i. Connections C_i is a set of connections {C_i^1, C_i^2, ..., C_i^c, ...}, where *c* denotes the order of connections, and lane C_i^c is [F_i^c, G_i^c]. Therefore, each connection contains two elements: the from-road ID F_i^c and the to-road ID G_i^c.

Fig. 3. shows an example of a 3D junction. JunctionJ_3, which is indicated by the yellow area, is described byJ_3=[*3*, C_3] where C_3=[C_3^1, C_3^2, C_3^3, C_3^4, C_3^5, C_3^6],C_3^1=[*40*,*168*], C_3^2=[*40*,*70*], C_3^3=[*159*,*73*], C_3^4=[*159*,*41*], C_3^5=[*72*, *158*], and C_3^6=[*72*, *41*]

3.3. Redrawing of the top-view image of the whole environment

After extracting all the information regarding the roads and junctions in a virtual environment, a top-view image of the whole environment is redrawn. First, as shown in Fig. 4, image width W and image height H are determined. According to the size of the whole environment, one pixel corresponds to a unit length α in the environment. There are three axes X, Y, and Z in the virtual environment. To redraw a top-view image, the Y axis is ignored, the image width W is the size of the environment along the Z axis, and the image height H is the size along the X axis. By considering all the start points and end points of all road information, the minima and maxima, $x_{min}=Min(x_i^n)$, $x_{max}=Max(x_i^n)$, $z_{min}=Min(z_i^n)$, and $z_{max}=Max(z_i^n)$ are extracted. To prevent cropping errors, the corresponding image edge is extended by a number of pixels β. Therefore, image width W $\frac{(z_{\max}-z_{\min})}{\alpha}+2\beta$ and image height H $\frac{(x_{\max}-x_{\min})}{\alpha}+2\beta$. To redraw the image, start points $S_i^n=(x_i^n, y_i^n, z_i^n)$ and end points $E_i^n=(x_i^n, y_i^n, z_i^n)$ $w=((z_i^n-z_{\min})\times\frac{1}{\alpha}+\beta)$ $h=((x_i^n-x_{\min})\times\frac{1}{\alpha}+\beta)$.

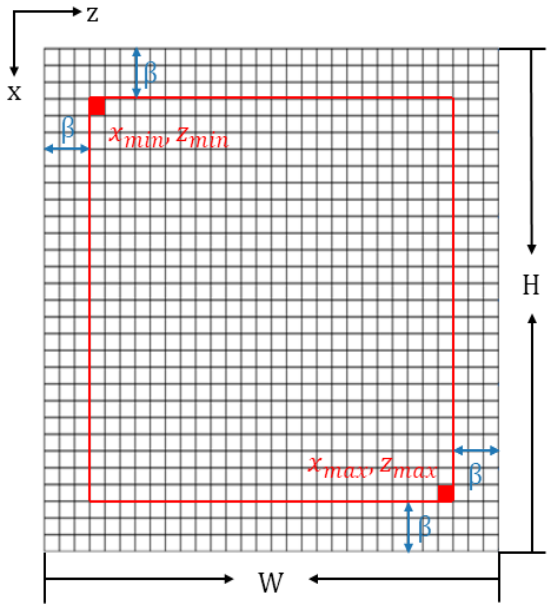

Fig. 4. Image size determination

After converting all locations of the start points and end points of all lanes to pixel coordinates, the top-view image of whole environment is redrawn. First, as shown in Fig. 5, a straight road is represented by a straight line between the start point S_i^n and the end point E_i^n of each lane (red and blue points are only for explanation). A corner road is then represented by drawing an arc between the start point S_i^n and end point E_i^n of each lane.

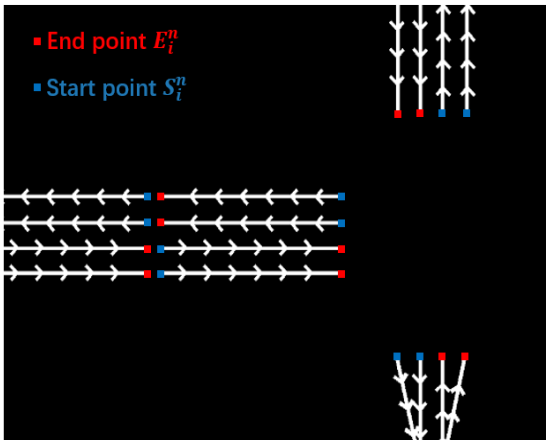

Fig. 5. Drawing road connections

Subsequently, as shown in Fig. 6,adjacent roads are connected by drawing straight lines between their end points E_i^n and start points $S_{L_i^{SR}}^n$.

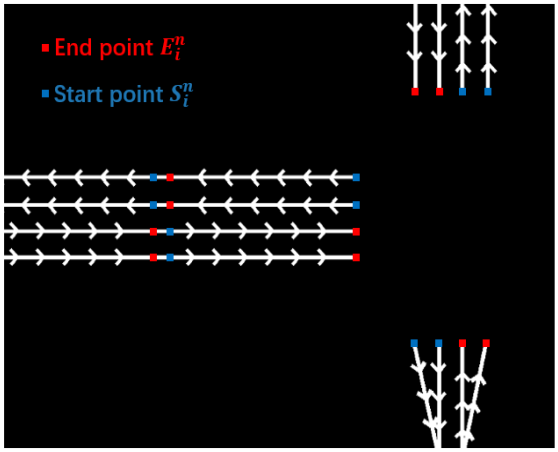

Fig. 6. Drawing connections between the roads and successor roads

Next, as shown in Fig. 7, the straight connection in junction$J_i$is connected by straight lines. The turning connections in the junction are drawn using arcs to connect end points $E_{F_i^c}^n$ and start points $S_{G_i^c}^n$.

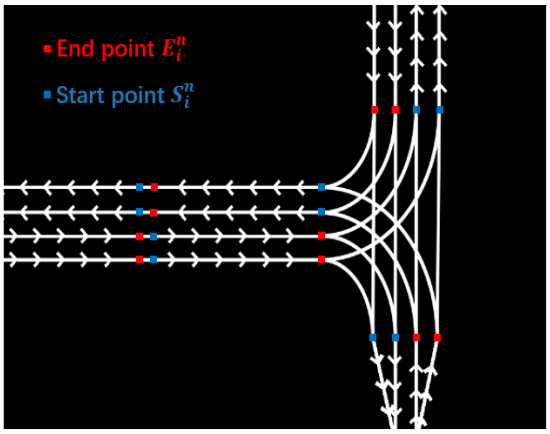

Fig. 7. Drawing junction connections

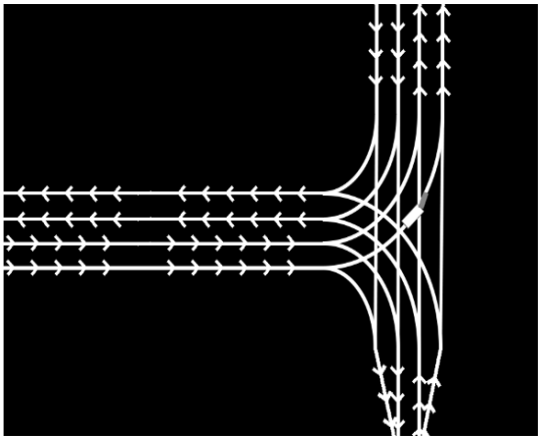

Fig. 8. Adding thevehicle agent

After the top-view image of the whole environment has been drawn, vehicle agent*v* also needs to be added to the image as shown in Fig. 8. The vehicle agent coordinates on the x-axis x_v, and the vehicle agent coordinates on the z-axisz_v, as well as directionD_v are collected for each frame. Therefore, thevehicle agent's pixel coordinate $w_v$and h_v are $(z_v - z_{\min}) \times \frac{1}{\alpha} + \beta$ and $(x_v - x_{\min}) \times \frac{1}{\alpha} + \beta$, respectively. Finally, as shown in Fig. 8, the vehicle agent is located at pixel (w_v, h_v) and rotated to the direction$D_v$in the top-view image of the whole environment.

3.4. Top-view image cropping

To train a DQN model, a vehicle agent only needs to consider the surrounding environments. Therefore, the top-view image of whole environment needs to be cropped to a small area centered on the location of the vehicle agent as shown in Fig. 9. To further reduce the number of features in the top-view image, the vehicle agent should always face in one direction.

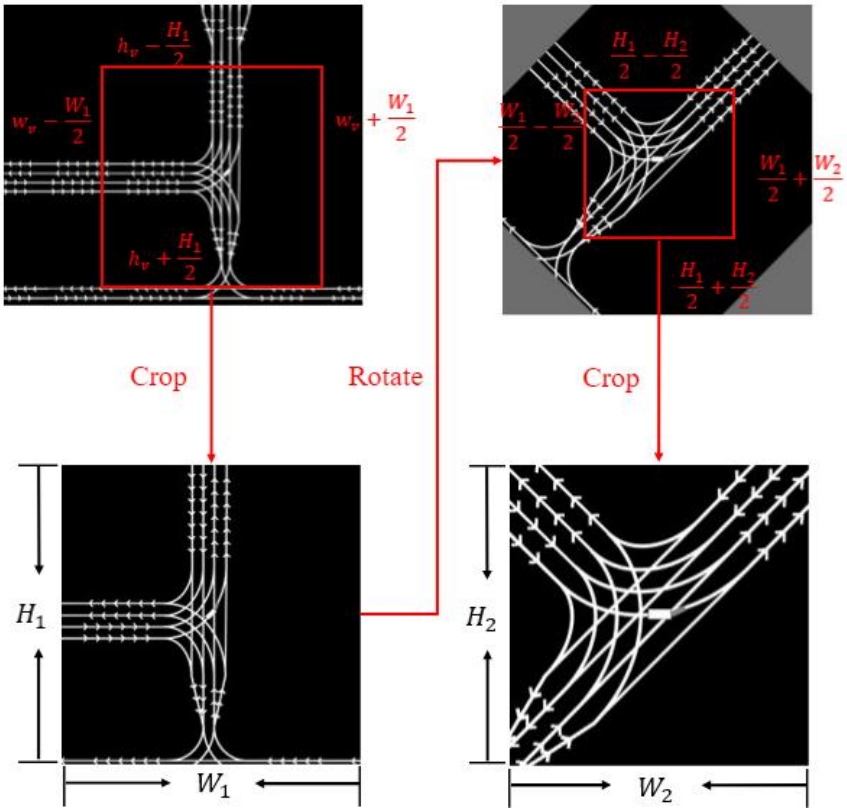

Fig. 9. Top-view image cropping process

To prevent rotation error, a two-step cropping approach is proposed. In the first crop, the width W_1 and height H_1 of the first crop are larger than the width W_2 and height H_2 of the second crop.

The signed angle θ between direction D_v of vehicle agent v and a fixed direction D_0 is utilized to rotate the top-view image after the first crop. When rotating, the resolution of the image should remain the same. The final cropped top-view image is utilized as the DQN input for each frame.

A single frame of a cropped top-view cannot contain information about the vehicle agent's speed. The cropped top-view image is converted to grayscale and stacked with the previous two frame images which have also been converted to grayscale. Fig. 10

shows a three frames top-view image. The time interval between the adjacent frames should be kept fixed.

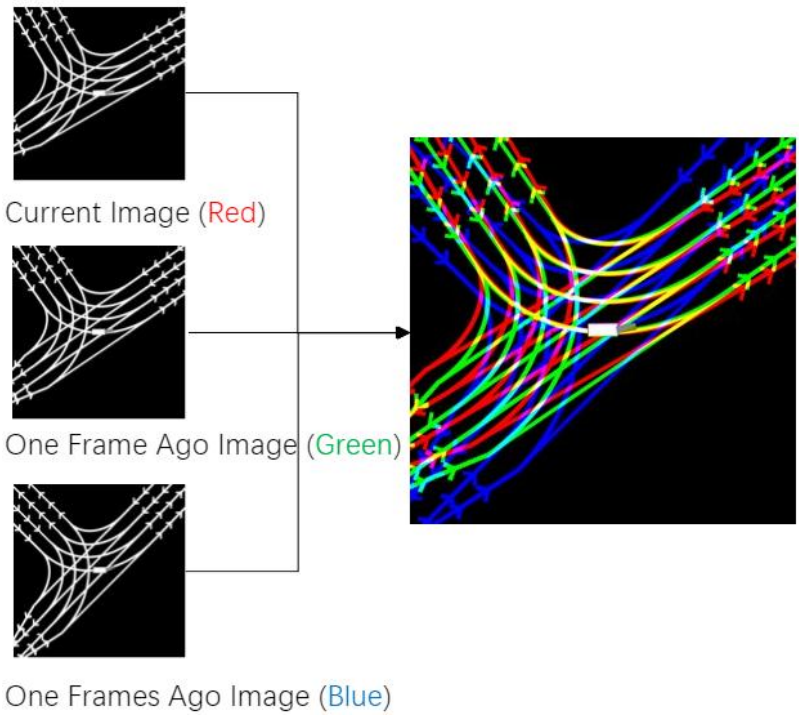

Fig. 10. Top-view image of three stacked adjacent frames

3.5. DQN architecture with DQN-input images

The structure of a DQN is shown in Fig.11. The DQN-input image is a 240×240 top-view image and the outputs are steering and acceleration values. Steering values range from -1 to 1, in steps of 0.1. Acceleration values range from −1 to 1 in steps of 0.2. This network utilizes categorical cross entropy as the loss function. The details of the parameters of each layer are listed in Table 1.

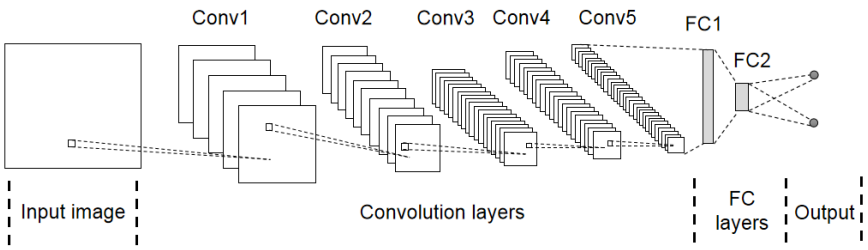

Fig. 11. DQN architecture

Table 1. Parameters of the layers of a DQN

Convolution Layer	Filters	Kernel Size	Stride
Conv1	24	5×5	2×2
Conv2	36	5×5	2×2
Conv3	48	5×5	2×2
Conv4	64	3×3	1×1
Conv5	64	2×2	1×1
Fully-connected Layer	Neurons		
FC1	36,864		
FC2	128		

The function used to calculate the DQN training reward is

$$R(V_t, \theta_t, D_t) = V_t \times (\cos(\theta_t) - \sin(\theta_t)) \quad \frac{|D_t - D_0|}{D_0} \quad (1)$$

where V_t is the vehicle agent's speed along the road direction, $\theta_t$is the angle between the heading of the vehicle agent and the direction of the road, $D_t$is the distance from the road's left edge to the vehicle agent's center, and $D_0$is the half of the road width.

As the number of training iterations increases, exploratory rate $\varepsilon_t$is decreased by

$$\varepsilon_t = Max(\varepsilon_t) \quad \frac{t \times (Max(\varepsilon_t) - Min(\varepsilon_t))}{T} \quad (2)$$

where $Max(\varepsilon_t)$ is the maximum of $\varepsilon_t$and $Min(\varepsilon_t)$ is the minimum of ε_t.In addition, T is the total number of frames that the vehicle agent has trained and t is current frame.

4. Experiments

In this section, the experimental results are described. Section 4.1 introduces the experimental goal and environment. Section 4.2 presents the results of the top-view image of the whole environment. Section 4.3 presents the samples of a cropped top-view image, and Section 4.4 presents the vehicle agent obtained using a DQN.

4.1. Experimental goal and environment

The experimental goal is to train a DQN model to control a vehicle agent in a virtual environment. To evaluate the proposed method, the results obtained using the preprocessed three-frame top-view image are compared with the results obtained using captured top-view image generated by the simulator without any preprocessing and a preprocessed single-frame top-view image. The rewards of each DQN trained with one of the three types of DQN inputs were calculated, and the number of successfully reached destinations of a vehicle agent controlled by each trained DQN was analyzed.

To train the DQN model, 300,000 frames were used. The maximum exploratory rate was 0.3 and the minimum exploratory rate was 0.01. The replay memory stored the most recent 5000 frames of data. For every frame, the batch size for the DQN was 32.

The performance of the proposed method was verified on a desktop computer with an i7-7740X 4.30GHz CPU, an NVIDIA GeForce GTX 1080 graphics card, and a 16.0GB DDR4 RAM. The virtual environment simulator shown in Fig.12 runs on Unity3D.

Fig. 12. Virtual environment simulator used for the experiment

4.2. Results of the top-view image redrawing

In the experiment environment, there were eight junctions and a total of 162 roads: 140 straight, 7 corner, and 15 merging roads. The captured top-view image is shown in Fig. 13, and Fig. 14 shows the final redrawn top-view image of the whole environment.

Fig. 13. Capture top-view image of the whole environment

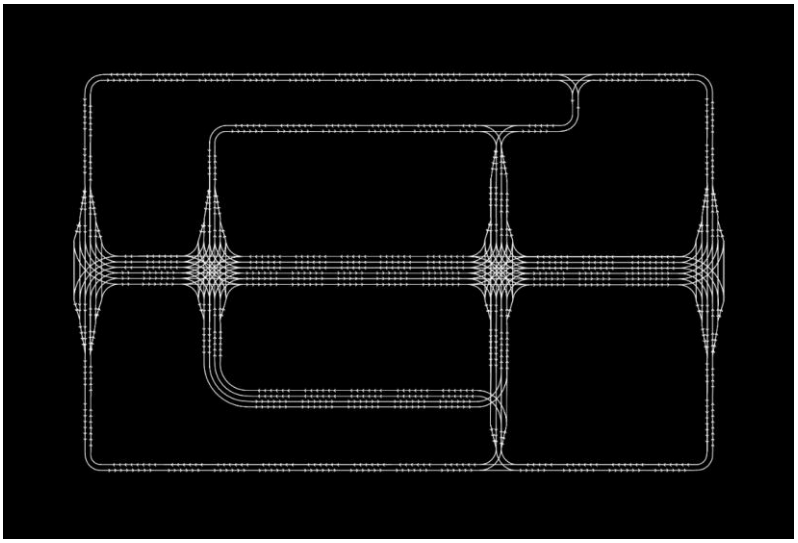

Fig. 14. Redrawn top-view image of the whole environment

In the redrawn image, the white lines represent road lanes and the arrows on the lines indicate the direction of the lanes. The areas with white crossing lines are the junctions which contain many road connections. In contrast to the captured top-view image in Fig. 13, information that is clearly not related to training a vehicle agent DQN model has been removed from the redrawn top-view image of the whole environment.

4.3. Results of top-view image cropping

Captured top-view image samples, single-frame top-view image samples, and three-frame top-view samples are shown in Figs. 15, 16, and 17, respectively. The left panels in these figures show the vehicle agent driving on a straight road. The middle panels show the vehicle agent turning right at a corner, and the right panels show the vehicle agent turning right at a junction.

Fig. 15. Captured top-view image samples

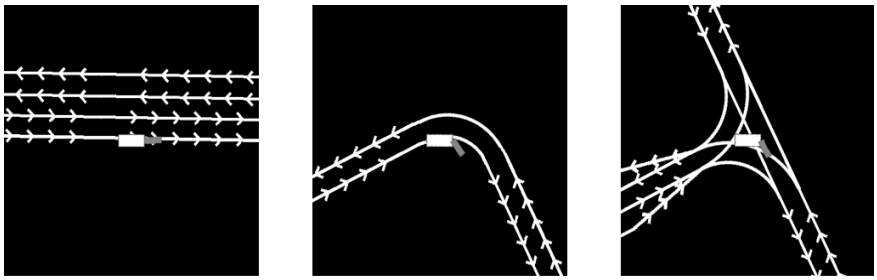

Fig. 16. Single-frame top-view image samples

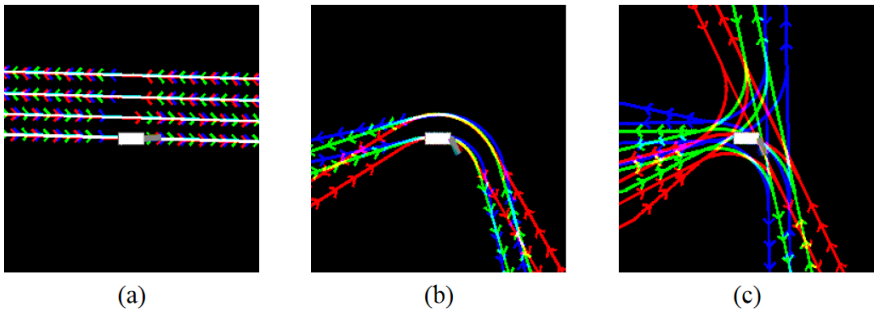

Fig. 17. Three-frame top-view image samples

4.4. Result of training using a DQN

Three DQN models were trained with 300,000 frames of captured images, single-frame images, or three-frame images. The mean rewards of the three DQN models after 10,000 frames were processed are shown in Table 2. The mean reward of the DQN model trained with the proposed three-frame images was 221.3% higher than that of the DQN model trained with the captured top-view images.

Table 2. Comparison of the mean DQN model rewards after 10,000 frames

Input image type	Mean reward
Captured	1.6244
Single frame	4.1360
Three frames	5.2193

To evaluate the performance of the DQN models trained with the three types of input images, experiments were carried out on the same map. Each DQN model was used to control a vehicle agent to a destinations 100 times, and the number of successfully reached destination is listed in Table 3. These results show that the DQN model trained with the proposed three-frame top-view images has a success rate that is 712.5% higher than that of the DQN model trained with the captured top-view images.

Table 3. Comparison of the number of successfully reach destinations

Input image type	Example
Raw	8
Single frame	49
Three frames	65

5. Conclusions

This paper proposed an image preprocessing method for DQN image inputs to train an autonomous vehicle agent in a virtual environment. In this method, the information of the roads and junctions related to a vehicle agent in a virtual environment is extracted and recorded in a file. Using this information, the size of the top-view images is determined. In the top-view image, roads and junctions are drawn as straight lines and arcs. The virtual environment obtained vehicle agent locations and directions for every frame. The top-view image is then cropped so that the vehicle agent is in the center of the cropped image. The outputs of the DQN are steering and acceleration values, which can be utilized to control the vehicle agent in the virtual environment.

In the experiment, after the DQN was trained for 300,000 frames, the mean training reward of the DQN model with the proposed top-view images reached 5.2193. The number of successful navigations using the DQN model trained with the proposed top-view images was 712.5% higher than tthat obtained by the DQN model trained with the captured top-view images. These results show that the proposed DQN input image preprocessing method can substantially improve the performance of the DQN model.

In future work, we will explore more applications of the proposed image preprocessing method. Further, we will test the model using other neural networks that use image as input.

Acknowledgment. This research was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIP) (2018R1A2B2007934) and was supported by the Dongguk University Research Fund of 2020.

References

1. Niranjan Deshpande, Anne Spalanzani: Deep Reinforcement Learning based Vehicle Navigation amongst pedestrians using a Grid-based state representation. IEEE Intelligent Transportation Systems Conference, 2081-2086. Auckland, New Zealand. (2019)
2. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao and Karol Zieba: End to End Learning for Self-driving Car. (2016). [Online]. Available: https://arxiv.org/pdf/1604.07316
3. Ahmed Hussein, EyadElyan, Mohamed Medhat Gaber and Chrisina Jayne: Deep Imitation Learning for 3D Navigation Tasks. Neural Computing and Applications, Vol. 29, 389-404. (2017)
4. Dean A. Pomerleau: Alvinn: An Autonomous Land Vehicle in A Neural Network. Advances in Neural Information Processing Systems, 305-313. Denver, United States of America. (1989)
5. Stéphane Ross, Geoffrey J. Gordon and J. Andrew Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning. International Conference on Artificial Intelligence and Statistics, 627-635. Fort Lauderdale, United States of America. (2011)
6. Dean A. Pomerleau: Efficient Training of Artificial Neural Networks for Autonomous Navigation. Neural Computation, Vol.3, 88-97. (1991)
7. Ge, Y., Zhu, F., Huang, W., Zhao, P., Liu, Q.: Multi-Agent Cooperation Q-Learning Algorithm Based on Constrained Markov Game. Computer Science and Information Systems, Vol. 17, No. 2, 647–664. (2020)
8. Andrew Y. Ng and Stuart Russell: Algorithms for Inverse Reinforcement Learning. International Conference on Machine Learning, Vol. 1, 2. Palo Alto, United States of America. (2000)
9. Brian D. Ziebart, Andrew Maas, J.AndrewBagnell and Anind K. Dey: Maximum Entropy Inverse Reinforcement Learning. Association for the Advance of Artificial Intelligence, Vol. 8, 1433-1438. Chicago, United States of America. (2008)
10. Volodymyr Mnih, KorayKavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, IoannisAntonoglou, Helen King, Dharshan Kumaran, DaanWierstra, Shane Legg and Demis Hassabis: Human-level Control through Deep Reinforcement Learning. Nature, Vol. 518, 529-533. (2015)
11. Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver and DaanWierstra: Continuous Control with Deep Reinforcement Learning. (2015). [Online]. Available: https://arxiv.org/abs/1509.02971
12. Burr Settles: Active Learning Literature Survey. Computer Sciences Technical Report 1648. (2009)
13. Umar Syed and Robert E. Schapire: A Reduction from Apprenticeship Learning to Classification. Advances in Neural Information Processing Systems, 2253-2261, Vancouver, Canada. (2010)

14. Jonathan Ho, Jayesh K. Gupta and Stefano Ermon: Model-free Imitation Learning with Policy Optimization. International Conference on Machine Learning, 2760-2769. New York, United States of America. (2016)
15. Zongwei Zhou, Jae Shin, Lei Zhang, SuryakanthGurudu, Michael Gotway, and Jianming Liang: Fine-tuning Convolutional Neural Networks for Biomedical Image Analysis: Actively and Incrementally. Conference on Computer Vision and Pattern Recognition, 7340-7351. Honolulu, United States of America. (2017)
16. Makantasis, Konstantinos, Maria Kontorinaki, and IoannisNikolos: A Deep Reinforcement Learning Driving Policy for Autonomous Road Vehicles. (2019). [Online]. Available: https://arxiv.org/abs/1905.09046
17. Chenyi Chen, Ari Seff, Alain Kornhauser, Jianxiong Xiao: Deepdriving: Learning affordance for direct perception in autonomous driving. IEEE International Conference on Computer Vision, 2722-2730. Santiago, Chile. (2015)
18. Bloem, Michael, and Nicholas Bambos: Infinite time horizon maximum causal entropy inverse reinforcement learning. IEEE Conference on Decision and Control, 4911-4916. Santiago, Chile. (2014)
19. Fan, H., Xia, Z., Liu, C., Chen, Y., Kong, Q: An Auto-tuning Framework for Autonomous Vehicles. (2018). [Online]. Available: https://arxiv.org/abs/1808.04913
20. Sasaki, Fumihiro, Tetsuya Yohira, and Atsuo Kawaguchi.: Sample efficient imitation learning for continuous control. International Conference on Learning Representations.(2018)
21. Kim, B., Farahmand, A. M., Pineau, J., Precup, D: Learning from limited demonstrations. In Advances in Neural Information Processing Systems, 2859-2867. (2013)
22. Ho, Jonathan, and Stefano Ermon.: Generative adversarial imitation learning. Advances in neural information processing systems, 4565-4573. (2016)

Kaisi Huang received his B. Eng. Degree in Digital Media Technology in 2017 from North China University of Technology, Beijing, The People's Republic of China. Since September 2017, he is in the M. Eng. course at the Department of Multimedia Engineering, Dongguk University, Seoul, Republic of Korea. He has done some works mainly associated with NUI (Natural User Interface) applications, VR (Virtual Reality) applications and artificial intelligence with deep learning.

Mingyun Wen received the B. Eng. Degree in Digital Media and Art in 2016 from North China University of Technology, Beijing, China and the M.Eng. in Multimedia Engineering in 2018 from Dongguk University, respectively. She is a Ph.D. candidate in the department of Multimedia Engineering in Dongguk University now. Her main research interests include 3D simulation of large virtual environment, artificial intelligence and deep learning.

Jisun Park received her B. Eng. Degree in Multimedia in 2016 from Dongguk universitiy computer science institute, Seoul, Republic of Korea and M. Eng. Degrees in Multimedia Engineering in 2018 from Dongguk University, Seoul, Republic of Korea. Since September 2018, she is in the Dr. Eng. course at the Department of Multimedia Engineering, Dongguk University, Seoul, Republic of Korea. She has done many works mainly associated with 2D/3D artificial intelligence with deep learning.

Yunsick Sung, currently an associate professor in the Department of Multimedia Engineering at Dongguk University-Seoul, Seoul, Republic of Korea. He received his BS degree in the Division of Electrical and Computer Engineering from Pusan National

University, Busan, Republic of Korea, in 2004, the M. S. degree in Computer Engineering from Dongguk University-Seoul, Seoul, Republic of Korea, in 2006, and the Ph. D. degree in Game Engineering from Dongguk University, Seoul, Republic of Korea, in 2012. He was employed as a member of the Researcher at Samsung Electronics in the Republic of Korea between 2006 and 2009. He was a postdoctoral fellow at the University of Florida, Florida, USA, between 2012 and 2013. His research interests are focused on the areas of superintelligence in the fields of immersive softwares, UAVs, security, and music using demonstration-based learning and deep learning.

Jong Hyuk Park received Ph.D. degrees in Graduate School of Information Security from Korea University, Korea and Graduate School of Human Sciences from Waseda University, Japan. From December, 2002 to July, 2007, Dr. Park had been a research scientist of R&D Institute, Hanwha S&C Co., Ltd., Korea. From September, 2007 to August, 2009, He had been a professor at the Department of Computer Science and Engineering, Kyungnam University, Korea. He is now a professor at the Department of Computer Science and Engineering and Department of Interdisciplinary Bio IT Materials, Seoul National University of Science and Technology (SeoulTech), Korea. His research interests include IoT, Human-centric Ubiquitous Computing, Information Security, Digital Forensics, Vehicular Cloud Computing, Multimedia Computing, etc.

Kyungeun Cho is a full professor at the Department of Multimedia Engineering at the Dongguk University inSeoul, Republic of Korea since Sept. 2003. She received her B. Eng. Degree in Computer Science in 1993, herM. Eng. and her Dr. Eng. Degrees in Computer Engineering in 1995 and 2001, respectively, all from DonggukUniversity, Seoul, Korea. During 1997–1998 she was a research assistant at the Institute for Social Medicine atthe Regensburg University, Germany, and a visiting researcher at the FORWISS Institute at TU-MuenchenUniversity, Germany. Her current research interests are focused on the areas of intelligence of robot and virtualcharacters and real-time computer graphics technologies. She has led a number of projects on robotics and gameengines and also has published many technical papers in these areas.

Received: February 12, 2020; Accepted: January 10, 2021.