文章编号：0253-374X(2022)S1-0119-09 **DOI**：10. 11908/j. issn. 0253-374x. 23707

基于卷积神经网络感知功能的极端场景生成方法

Kun GAO[1]，Hans-Christian REUSS[2]

（1. 斯图加特汽车工程与车辆发动机研究所(FKFS)，斯图加特 70569，德国；
2. 斯图加特大学 汽车工程学院(IFS)，斯图加特 70569，德国）

摘要：近年来，基于卷积神经网络深度学习的感知算法在自动驾驶车辆环境感知系统中发挥着越来越重要的作用。由于在神经网络训练过程中，训练数据无法覆盖所有极端场景，因此如何保证基于深度学习的感知算法在极端场景下的安全性和可靠性，仍是一个亟待解决的问题。传统的基于真实行驶里程的验证方法，在获取极端场景数据上危险性高，经济性差，因此很难检验驾驶功能在极端场景下的性能。基于虚拟场景的仿真验证方法，虽然可以通过设置场景参数来生成大量测试场景，但是通过简单的参数组合并不能有效的生成极端场景。本文展示了一种在虚拟环境中生成极端场景的方法，用于训练和测试基于深度卷积神经网络的车道线识别算法。首先将场景特征用参数进行表示，然后使用deep Q-learning 强化学习的方法，来生成极端场景的参数组合。通过与随机组合以及成对组合场景参数的方法进行对比，可以看出该基于强化学习的场景生成方法可以更有效地生成极端场景，因此可提高自动驾驶感知功能的测试效率，同时可为卷积神经网络提供更多的极端场景训练数据。

关键词：自动驾驶；极端场景；卷积神经网络；强化学习
中图分类号：TP389. 1 **文献标志码**：A

Corner Cases Generation for Virtual Scenario-based Testing of CNN-based Autonomous Driving Function

Kun GAO[1]，*Hans-Christian REUSS*[2]
（1. Research Institute of Automotive Engineering and Vehicle Engines Stuttgart(FKFS)，70569 Stuttgart，Germany；2. Institute of Automotive Engineering(IFS)，University of Stuttgart，70569 Stuttgart，Germany）

Abstract： Deep learning-based perception algorithms have gained importance in autonomous vehicle perception systems in recent years. Since the training data cannot cover all critical scenarios and corner cases，how to ensure the safety and reliability of deep learning-based perception functions in crucial scenarios is still an open challenge. Conventional approaches test the driving functions in real-life environments，which can be risky and uneconomic to validate in corner cases. Virtual scenario-based simulation validation approaches can generate a large number of test cases by setting test scenario parameters，but the purely combinatorial parameter cannot effectively generate corner cases. In this paper，we present a novel approach to generating corner cases in a virtual environment for validation of a CNN (Convolutional Neural Network) -based lane detection function. We represent the scene features with parameters，and then use the deep Q-learning reinforcement learning approach to generate the parameter combinations of corner cases. In addition，by comparing with the approaches of random combination and pairwise combination of scene parameters，our approach can generate corner cases more efficiently and improve the testing efficiency of the autonomous driving perception functions.

Key words： automated driving； corner case； convolutional neural network； reinforcement learning

Autonomous driving is one of the key topics in research and industry. Through many sensors (e. g. vision sensors，radar，lidar，etc.)，self-driving vehicles can recognize their surroundings and identify potential hazards to ensure a safe driving. Since convolutional neural networks can automatically extract features with generalization ability and robustness，it is increasingly used in environment perception functions，especially vision-based environment perception functions in autonomous

收稿日期：2022-09-10
第一作者：Kun GAO(1990—)，男，理学硕士，主要研究方向为汽车机电一体化。E-mail：kun. gao@fkfs. de
通信作者：Hans-Christian REUSS (1959—)，男，教授，工程学博士，主要研究方向为汽车机电一体化。
E-mail：hans-christian. reuss@ifs. uni-stuttgart. de

driving. However, how to effectively test the safety and reliability of deep learning-based perception algorithms, especially in corner cases, is still an open challenge.

Conventional validation methods require the accumulation of a certain number of safe driving miles. According to Lipinski[1], the self-driving vehicle must complete a distance of 6.14 billion km of safe driving range on public roads before they can be awarded approval for series production. A considerable part of the driving scenarios on public roads is common scenarios, which are not very helpful for validating the algorithm. Furthermore, the number of corner cases in real-live driving is limited. It is unsafe for drivers and other traffic participants when acquiring data of corner cases. This is not an option to validate autonomous driving functions with such a long test distance in terms of economy, feasibility, and safety.

Another validation method is the scenario-based validation method. By validating the autonomous driving function in different test scenarios as well as critical scenarios to determine the safety and reliability of the driving functions. For the scenario-based method, a large number of test scenarios can be constructed effectively and reproductively through the combination of scenario parameters.[2] Moreover, critical scenarios and corner cases can be generated in the virtual world by utilizing the simulation software. The scenario-based virtual validation method can be effortlessly combined with Model-in-the-Loop and Software-in-the-Loop test methods, to support the full development process from the system development stage to the system testing stage. However, the scenario describing parameter space usually very huge, the corner case scenarios cannot be generated effectively by simple combination of scenario parameters. For this reason, an effective method is needed to search for corner case scenario parameter combinations in the massive parameter space.

1 Corner case and scenario-based validation method

The deep learning algorithms in autonomous driving need to be trained with a large quantity of data. To improve the reliability and safety of deep learning algorithms in real-live driving, more data is needed, especially the data from uncommon scenarios or even dangerous scenarios in the real world. These scenarios or scenes are called corner cases. According to Kowol et al.[3], corner cases include anomalies, unknown objects or outliers, which are outside of training data of deep learning algorithm. Moreover, corner cases are relative to the normal driving scenarios. Corner cases occur infrequently and usually represent that the vehicle is in critical situation. Since a corner case is also a scenario, a scenario-based method can be used to generate corner cases in different situations.

Scenario-based validation method plays an essential role in the testing of autonomous driving functions in recent years. Scenarios are generally to substantiate test cases for autonomous driving functions[4]. Therefore, the scenario-based validation method can be used to generate uncommon scenarios as corner cases to support the training and validation of deep learning functions in autonomous driving. There are many scenario description methods in the current research field, among which the representative ones are:

The Pegasus project[5] presented a 6-layer scenario description model to categorize the traffic elements. Exchangeable and reproducible scenarios can be defined with the 6-layer model.

Ebner[6] allocates three components to the scenario:

- Ego-vehicle: defines some parameters of the ego vehicle itself.
- Traffic participants: other vehicles, pedestrians, bicycles, etc.
- Environmental elements: roads, infrastructure, weather, light, and other objects.

The scenario description method provides the basic elements needed to describe the scenario. The scenario elements can be selected to generate the required study scenarios. Then, the scenario extraction method is used to select the hazard scenarios as corner cases.

Ahmed[7] et al. proposed a narrative-based scenario generation methodology to generate challenging scenarios in virtual environment. Ahmed et al. used CARLA virtual environment for a generation s set of scenarios of different weather conditions and different number of traffic participants for analyzing the driving performance of an Artificial Intelligent (AI) agent in simulation. Challenging scenarios are represented by the complexity of the scenario. Song[8] et al. establish an approach for critical test scenario identification by selecting the parameters of critical scenarios with an optimization model. The optimization model generates the next new scenario with distinct parameter values, which works as the test scenario for the simulation platform. Kowol[3] et al. purposed an approach to generate synthetic corner cases using a human-in-the-loop. In the test loop, a human as the driver controls the vehicle according to the semantic segmented images from an AI function. Another human as the safety officer, observes the original images to determine whether the driver has misunderstood the semantic segmented images. When the safety officer thinks that the driver has misunderstood the scene, the scene will be stored as corner cases. These corner case generation methods employ simulation tools to rapidly generate a large number of virtual scenarios that provide data for training, validation, and testing of AI functions. However, these methods have some weaknesses, such as the inefficiency and low automation of corner case generation.

Motivated by the study of Kowol[3], we focus on automatic corner cases generation for CNN-based autonomous driving function and employ a deep Q-learning approach to generate parameter combinations of relevant and critical scenes. In this paper, we introduce a method for corner case generation in the virtual world and used CNN-based lane detection as an example. Environmental elements are the main factors affecting CNN-based lane detection. Therefore, in this approach, we use Ebner′s scenario description method[6] to model the scene. The lane detection algorithm uses the encoder-decoder U-Net to perform image semantic segmentation of lane lines and backgrounds. The network architecture is shown in the Fig. 1.

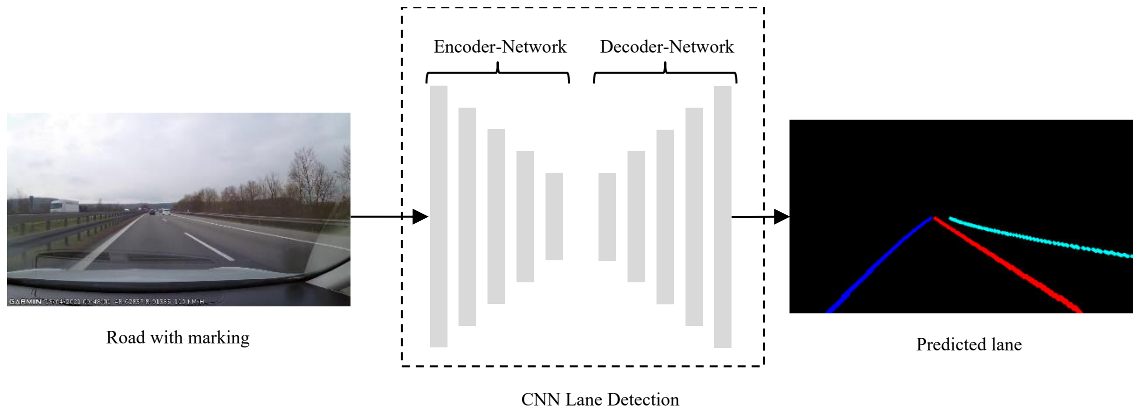

Fig. 1　CNN-based lane detection

The input is the front-view camera image. The network performs a binary classification of each pixel point of the image to determine whether it belongs to the lane line or the background. The difference between the predicted lane lines and Ground Truth is calculated to determine whether the lane lines are correctly detected. Since the number of lane line pixels is significantly less than the number of background pixels in the road scene, the training data is imbalanced. Dice Loss[9] is used to calculate the difference between the predicted lane lines and ground truth. Dice Loss takes a value between 0 and 1. The larger the difference between the predicted value and the target value, the more inaccurate the classification is, and the closer the dice loss is to 0.

$$\text{Dice loss}=1-\frac{2|X|\cap|Y|}{|X|+|Y|}$$

The recognition target for the lane detection algorithm is the lane marking. Therefore, environmental elements are mainly considered for description of the scene, for example, road curvature, lighting conditions, weather conditions, etc.

In this paper, corner cases are defined as: when the lane detection algorithm cannot correctly recognize the lane line in this scene, then the scene is saved as a corner case. A threshold of the difference between the predicted and the target can be used to determine whether the lane lines are correctly detected.

2 Concept for corner cases generation

In this section, the corner cases generation method for the CNN-based lane detection algorithm is presented. The environment parameters are used to describe the scene, and then modeling software is used to transform the scene parameters into virtual camera pictures. The system under test is a CNN-based lane detection algorithm. The test results are analyzed to determine whether the input scenes are corner cases.

2.1 Corner case generation process

The corner case generation process is as follows:

(1) Initialize the environment parameters;

(2) Generate virtual scenes;

(3) Execute the test;

(4) Analyze the test results;

(5) Store the parameters of corner cases.

The environment parameters are first randomly initialized by using a stochastic algorithm to generate the initial scene parameter combinations. The scene parameters are through the API of the scene modeling software as input into the scene modeling software to generate the virtual scene. The front camera pictures of the virtual scene are acquired by the virtual camera.

During the execution of the test, the camera pictures are entered the lane detection network as input data and the lane pixels in the pictures are predicted by the CNN algorithm. The difference between the predicted lane line and the target lane line is analyzed to determine whether the scene is a corner case. In the corner case, the CNN network cannot recognize the lane lines correctly. Therefore, the combination of corner case parameters can be stored on the computer to be subsequently retrained to the network. The performance of the CNN network can be improved.

2.2 Deep Q learning in corner cases generation

After storing the parameters of the corner cases, the new combination of parameter can be generated by looping. This is a decision process to decide how to update the parameters and the combination of parameters. So we can take an action to update a scene parameter, and the state after the decision is the virtual world scene, which can be generated after updating the parameter.

How to adopt an effective strategy to update the parameter combination can be solved by using Q-learning reinforcement learning. When the state-action pairs have a large dimensionality, deep Q-learning can be used to approximate the Q-table in Q-learning. Therefore, to improve the testing efficiency as well as to generate the corner cases more efficiently, the deep Q-learning reinforcement learning algorithm was used in our work.

In the above corner cases generation process (4), the test result is the difference between the predicted lane line and the target lane line. Therefore, the scene parameters and the corresponding test results can be used as input to generate the next parameter combination using deep Q-learning reinforcement learning. A corner case generation module based on deep Q-learning can be added after (4) to generate new scene parameters.

Fig. 2 shows the corner case generation process using deep Q-learning reinforcement learning. It can be assumed that the corner case generation module is an intelligent agent, which can change the scene parameters (states) through a series of actions (actions) and then generate a new scene parameter. After the action of the agent, we can define that the

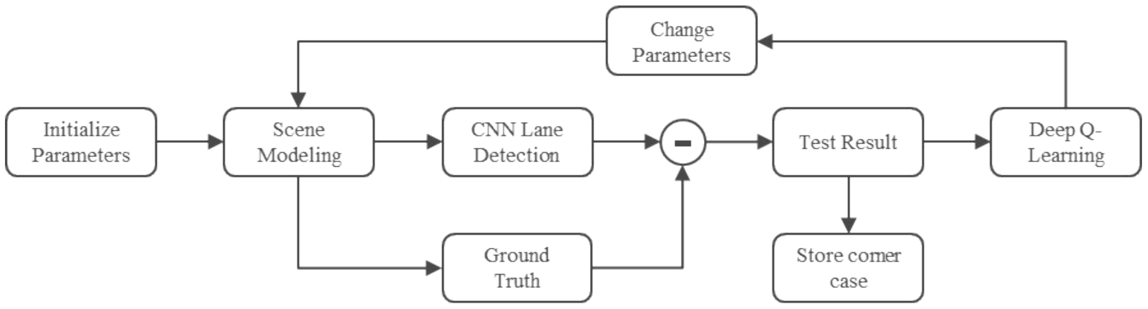

Fig. 2　Corner cases generation

environment gives a positive reward when the agent generates a corner case and gives a penalty when it generates a normal scene. The intelligent agent interacts with the environment and constructs a Q-table corresponding to the rewards and penalties, so that the agent can prioritize the actions that generate corner cases according to the Q-table. Q-table is updated with the formula:[10]

$$Q^{\text{new}}(s_t,\ a_t)=(1-\alpha)Q(s_t,\ a_t)+\alpha\Big(r_t+\gamma\max_a Q(s_{t+1},a)\Big)$$

where: r_t is the reward when moving from the state s_t to the state s_{t+1}; α is the learning rate $(0<\alpha<1)$; γ is the discount factor $(0<\gamma<1)$. The learning rate determines the extent to which the newly acquired information replaces the old information. The discount factor determines the importance of future rewards. The smaller the discount factor, the more short-sighted the Q-learning Agent is, and only cares about the rewards currently available.

The parameter space consisting of environmental parameters in this work is huge, and the corresponding Q-table will be too large to be completely stored in computer memory. Thus, a neural network can be used here to approximate the Q-table.

During the training process of deep Q-learning, states, actions, environmental rewards and the next state can be stored in an experience pool, and then a small batch of experience can be randomly sampled from the experience pool as training data and used to update the estimation of the Q-value. This method of using experience to train a neural network is called Experience Replay[11]. Through experience replay, the intelligent agent can learn from previous experience, not just from current experience, and avoid forgetting what it has already learned. In addition, traditional deep-Q-learning uses only one neural network to estimate the Q-table. When the training data have fluctuations, it will cause inaccurate estimation of the updated Q-value according to the formula[10], which will affect the deep-Q-learning performance and robustness. Therefore, a fixed Q-targets-Network[11] can be adopted as a second neural network to calculate the maximum Q-value for the next state. The Q-value in fixed Q-targets-Network is updated with the formula:

$$y_t=r_t+\gamma\max_a Q(s_{t+1},a)$$

The parameters in fixed Q-targets-Network are generally updated less frequently to avoid Q-value is overestimated or underestimated, which improves the robustness of the algorithm. The network is trained using mean squared error as the loss function.

3　Implementation

Corner cases are a challenge for deep Learning algorithms. Generating as many reproducible corner cases as possible can support improving the performance of training and inference. Collecting corner cases in the real world is not an option, because corner cases occur infrequently and are usually dangerous when they occur. For this purpose, we used the autonomous driving simulator CARLA[12] to generate a virtual world in this work. CARLA is an open source virtual scenario generation software whose scenarios are modeled using Unreal Engine and can generate different road traffic scenarios to develop and test autonomous driving AI algorithms. CARLA provides flexible APIs to define scenario parameters such as roads, traffic participants, weather, lighting, and road water. Fig. 3 shows the virtual scenes generated in CARLA.

As a case study for the visual detection algorithm, we mainly consider the parameters that have an impact on the camera image, such as Weather, lighting, road condition, and sensor failure are the main influencing factors of camera-based lane detection. Considering the used CARLA (Version 0. 9. 10) only cloudiness, precipitation, wind intensity, sun azimuth and fog as weather factors can be applied. The cloudiness and wind intensity have less influence on camera-based lane detection. The precipitation in CARLA lacks the distortion caused by raindrops on the windshield compared the real scene. Therefore, we only consider the sun azimuth and fog as weather factors in this case study. Lighting, road condition and sensor failure are adjusted using the parameters in the CARLA API. The following 6 scene parameters and their value ranges are defined in this paper.

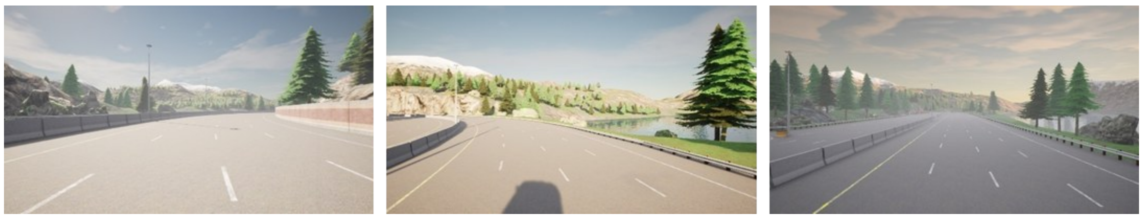
Fig. 3 CARLA virtual scenes

- Road curvature: when the road curvature is negative, it is a left-turn road; when the road curvature is equal to 0, it is a straight road; when the road curvature is positive, it is a right-turn road. The value range [−1, 1], where -1 means left turn, 0 means straight ahead, and +1 means right turn.

- Ambient brightness: set by the sun altitude angle in CARLA, its value range is [−90, 90], where −90 is late night, +90 is noon

- Sun azimuth: indicates the azimuth of the sun in the sky in the CARLA world, and its value range is [0, 360°].

- Fog density: weather parameter in CARLA world, indicating the visibility of vehicles, the range of values is [0, 100]. When this parameter is 0, the weather is clear, and the visual distance is unrestricted. In contrast, when this parameter is 100, the visual distance is highly limited, approximately around 10 meters.

- Water on the road: indicates whether there is water on the road surface, the value range is [0, 1], where 0 means the road is dry and 1 means there is visible water on the road.

- Blurring caused by the dirty lens: the value range [0, 1], where 0 means the lens is clean and the screen is normal; 1 means the lens is dirty and the screen is blurred.

Different scenes can be generated by changing the above six parameters and the corresponding combinations of parameters in the process of corner case generation. The agent in deep Q-learning model proposes to change the scene parameter so that different combinations of parameters are created. The agent can make 6 actions with 6 scene parameters. After executing an action, the environment generates a new state and rewards, which is in the CARLA virtual world. The environment rewards are related to the predicted result of the lane detection algorithm. The relation between actions and states can be approximated by a neural network. The agent in Q-learning always chooses the optimal action to get the maximum reward. However, only exploiting the current knowledge may lead to sub-optimal behavior. For the exploration of new options, we apply an epsilon-greedy approach to ensure that the agent has a probability to choose a random action for exploration, instead of always exploiting with prior knowledge.

Metric for corner cases: A threshold can be set to determine whether the lane lines in a scene are correctly detected or not. In this work, we assume that if the dice loss between the predicted and true values is greater than 0. 5, the lane lines of the scene will be considered as not correctly recognized.

4　Results

4.1　Number of corner cases

To test the deep Q-learning corner cases generation approach, we generated 20, 000 test scenes. The number of scenes in which lane cannot be accurately recognized is 11148, of which 56% are corner cases. As a comparison, we used two other approaches to generate test scenes:

- Random Testing: 20, 000 test scenes were generated using a combination of randomly selected scene parameters. The number of scenes in which the lane cannot be recognized is 4833, accounting for 24%.

- Combination Testing[13] : We used pairwise testing to combine discrete scene parameters, by the Microsoft PICT Tool to generate 2nd order combination test cases. A total of 18281 test scenes were generated, with 4092 scenes being corner cases, accounting for 22%.

Therefore, the deep Q-learning reinforcement learning method generates a higher percentage of corner cases. The corner cases generation approach is more effective. Fig. 4 shows the comparison of the corner cases generated by the random testing, combination testing and the deep Q-learning reinforcement learning method.

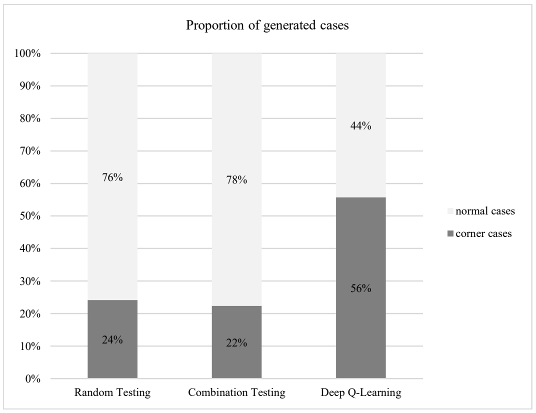

Fig. 4　Comparison of corner cases generated by random testing, combination testing and deep Q-learning

4.2　Analysis of corner cases

Different scene parameters have different effects on the CNN-based lane detection algorithm. Fig. 5 shows the effects of ambient brightness, water on the road, and dirty lens on the tested CNN algorithm. It can be observed that the lane detection algorithm is not accurate when the ambient brightness is low, when there is no water on the road (reflective effect of the dry road), and when the lens causes blurred images due to dirty lenses.

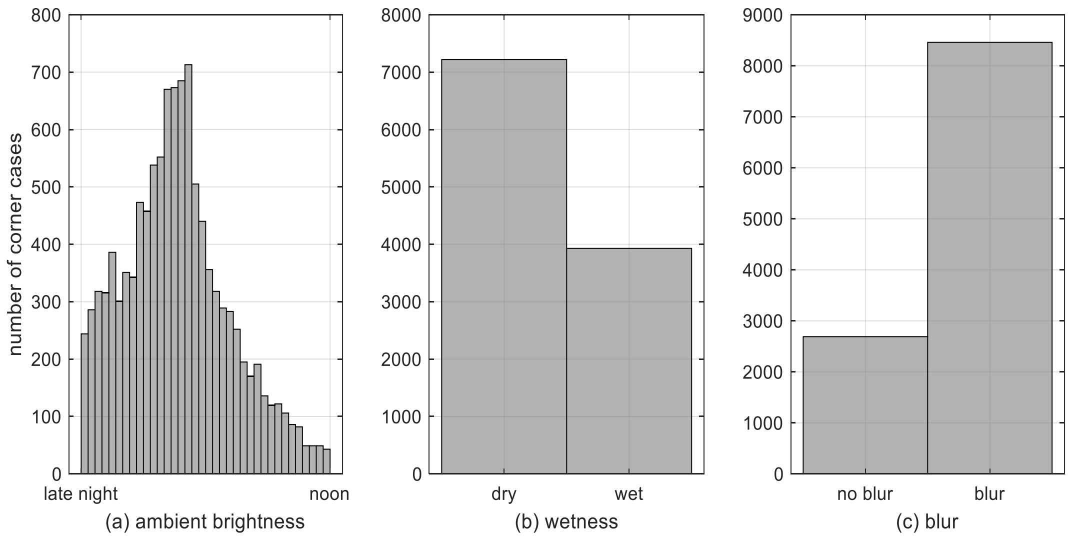

Fig. 5　Corner cases scene parameter distribution

Some of the generated corner cases are shown in Fig. 6. For example, (a) shows the blurred image caused by dirty lens, (b) the reflection phenomenon of water on the road surface under sunlight, (c) waterlogged road surface at night, and (d) dry road surface under high ambient brightness.

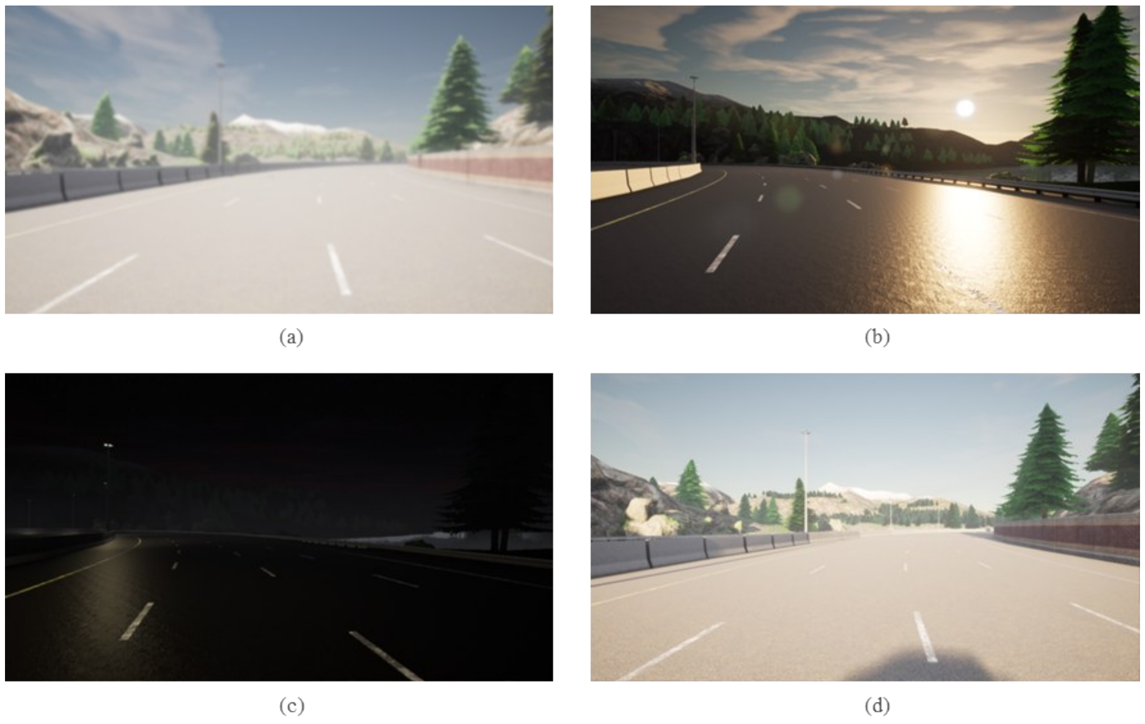
(a) (b) (c) (d)

Fig. 6　Examples for generated corner cases

5　Conclusions and outlook

In this paper, we demonstrate a method to validate CNN-based perception functions through virtual tests and employ deep Q-learning reinforcement learning to generate corner cases. By comparing with the methods of random testing and combination testing, this method can generate corner cases effectively and improve the testing efficiency.

In the future some potential improvements can be explored:

(1) More parameters can be considered to describe the environment and increase the environment details, such as precipitation, snow, the number of lanes, road users, etc.

(2) In this work we mainly analyze the scene in the corner case, and the scenario parameters are independent of time. Future research can be extended to dynamic scenarios, such as scenario parameters changing with time.

(3) Other reinforcement learning methods can be used to analyze the corner case in perception, decision making, and control algorithms in automated driving.

Reference:

[1] LIPINSKI D. Wie gut ist genug? —Pegasus Projekt, 2019[EB/OL]. https://www. pegasusprojekt. de/files/tmpl/Pegasus-Abschlussveranstaltung/PEGASUS_Abschlussveranstaltung_Wie_gut_ist_gut_genug.pdf.

[2] BAUMANN D, PFEFFER R, SAX E. Automatic generation of critical test cases for the development of highly automated driving functions[C]//2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring). IEEE, 2021: 1-5. DOI: 10.1109/VTC2021-Spring51267.2021.9448686.

[3] KOWOL K, BRACKE S, GOTTSCHALK H. A-Eye: Driving with the Eyes of AI for Corner Case Generation[J]. arXiv preprint arXiv, 2202: 2022.10803. DOI: 10.48550/ARXIV.2202.10803.

[4] ULBRICH S, MENZEL T, RESCHKA A, *et al*. Defining and substantiating the terms scene, situation, and scenario for automated driving [C]//2015 IEEE 18th international conference on intelligent transportation systems. IEEE, 2015: 982-988. DOI: 10.1109/ITSC.2015.164.

[5] PEGASUS. Projekt zur etablierung von generell akzeptierten gütekriterien, werkzeugen und methoden sowie szenarien und situationen zur freigabe hochautomatisierter fahrfunktionen[R]. 2019. https://www. pegasusprojekt. de/files/tmpl/pdf/PEGASUS_Abschlussbericht_Gesamtprojekt.PDF.

[6] EBNER A. Referenzszenarien als grundlage für die entwicklung und bewertung von systemen der aktiven sicherheit[D]. Berlin: Technischen Universität Berlin, 2014. https://d-nb. info/1067387501/34.

[7] AHMED M, SALEH K, ABOBAKR A, *et al*. Scenario

generation-based training in simulation：pilot study［C］//2019 IEEE International Conference on Systems，Man and Cybernetics（SMC）. IEEE，2019：1239-1244. DOI：0.1109/SMC.2019.8913985.

［8］ SONG Q，TAN K，RUNESON P，*et al*. Critical scenario identification for realistic testing of autonomous driving systems［J］. 2022. DOI：10.21203/rs.3.rs-1280095/v1.

［9］ JADON S. A survey of loss functions for semantic segmentation［C］//2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology（CIBCB）. IEEE，2020：1-7. DOI：10.1109/CIBCB48159.2020.9277638.

［10］ WATKINS C J C H，Dayan P. Q-learning［J］. Machine Learning，1992，8(3)：279-292. DOI：10.1007/BF00992698.

［11］ MNIH V，KAVUKCUOGLU K，SILVER D，*et al*. Playing atari with deep reinforcement learning［J］. arXiv preprint arXiv，2013:1312.5602. DOI：10.48550/arXiv.1312.5602.

［12］ DOSOVITSKIY A，ROS G，CODEVILLA F，*et al*. CARLA：An open urban driving simulator［C］//Conference on robot learning. PMLR，2017：1. DOI：10.48550/arXiv.1711.03938.

［13］ Kuhn D R，Bryce R，Duan F，*et al*. Combinatorial testing：Theory and practice［J］. Advances in Computers，2015，99：1.