Wen *et al. Hum. Cent. Comput. Inf. Sci.* (2020) 10:24
https://doi.org/10.1186/s13673-020-00231-z

Human-centric Computing and Information Sciences

RESEARCH Open Access

A scenario generation pipeline for autonomous vehicle simulators

Mingyun Wen, Jisun Park and Kyungeun Cho*

*Correspondence:
cke@dongguk.edu
Department of Multimedia Engineering, Dongguk University-Seoul, 30 Pildong-ro 1-gil, Jung-gu, Seoul 04620, Republic of Korea

Abstract

To develop a realistic simulator for autonomous vehicle testing, the simulation of various scenarios that may occur near vehicles in the real world is necessary. In this paper, we propose a new scenario generation pipeline focused on generating scenarios in a specific area near an autonomous vehicle. In this method, a scenario map is generated to define the scenario simulation area. A convolutional neural network (CNN)-based scenario agent selector is introduced to evaluate whether the selected agents can generate a realistic scenario, and a collision event detector handles the collision message to trigger an accident event. The proposed event-centric action dispatcher in the pipeline enables agents near events to perform related actions when the events occur near the autonomous vehicle. The proposed scenario generation pipeline can generate scenarios containing pedestrians, animals, and vehicles, and, advantageously, no user intervention is required during the simulation. In addition, a virtual environment for autonomous driving is also implemented to test the proposed scenario generation pipeline. The results show that the CNN-based scenario agent selector chose agents that provided realistic scenarios with 92.67% accuracy, and the event-centric action dispatcher generated a visually realistic scenario by letting the agents surrounding the event generate related actions.

Keywords: Artificial intelligence, Scenario generation, Convolutional neural network, Autonomous driving

Introduction

Autonomous driving has been a hot research topic since the end of the last century [1] because it promises many benefits, such as increased safety, reduced traffic congestion, and time savings. The first thing to consider when developing human-centric autonomous vehicles is safety [2]. The development of autonomous vehicles in the real world faces many problems, such as bad weather and difficulties in data collection.

The rapid development of computer hardware and artificial intelligence has allowed the development of simulators that provide efficient and convenient virtual environments for data collection and algorithm testing. In the past few decades, a variety of simulators have been developed for various purposes in machine learning [3], such as training employees [4, 5], soldiers [6], collecting training datasets [7], training models, and testing algorithms [8–11]. Compared with real-world testing,

simulation greatly reduces labor costs and time, as well as the risk of environmental factors or human error breaking expensive equipment. To build a simulator for autonomous vehicles equipped with various kinds of sensors, in addition to realistic visual effects [11], realistic simulation of possible real-world scenarios is also essential. A *scenario* usually means a series of *actions* of *agents* occurring over a period of time. In other words, a simulator should model various scenarios encountered by a vehicle in the real world. In this way, a vehicle model validated by such a simulator is more likely to succeed when applied to the real world. However, existing simulators [8, 10] have failed to achieve this goal. For example, in the simulator reported in Ref. [8], aside from walking on sidewalks and crosswalks, pedestrians do not interact with the vehicles. In the simulator reported in Ref. [10], pedestrians are not even modeled. Therefore, scenario generation still requires significant work.

The use of comprehensive mathematical descriptions to explain the scenarios occurring on roads is challenging because of their diversity and complexity [12, 13]. Deep learning provides a simple and efficient way to train mathematical models to obtain solutions to various problems [14], providing that good quality training data is available. In this paper, we integrated a convolutional neural network (CNN) with our scenario generation pipeline to evaluate whether the selected agents for scenario generation can achieve realistic results.

This paper proposes a pipeline for generating various kinds of scenarios in a simulator for autonomous vehicles. The simulator is used to provide a complete testing environment for the development of algorithms for autonomous vehicles, and various realistic scenarios are expected to be generated. A scenario is generated around the autonomous vehicle in a specific area described by a scenario node. The generation process is as follows. First, a scenario map consisting of many scenario nodes is generated. Each scenario node contains actions that must be invoked when the autonomous vehicle enters the corresponding area of each scenario node. Next, an event is triggered by a collision event detector or the execution of a specified action. Then the CNN-based scenario agent selector selects agents to generate scenarios considering their relative positions and directions in the virtual environment. Finally, to promote the development of scenarios, an event-centric action dispatcher is utilized to guide the selected agents to react automatically when events occur, for example, car accidents.

The proposed scenario generation pipeline makes the following contributions to the development of simulators for autonomous vehicles: (1) it can generate scenarios in real time according to information concerning the agents around the autonomous vehicle instead of generating a scenario over the entire virtual environment; (2) it is the first attempt to generate a scenario including cars, pedestrians, and animals, i.e., not just cars or only cars and pedestrians; and (3) it is the first time that CNN has been used to select agents to generate scenarios for autonomous driving simulators.

This paper is organized as follows. The next section presents a review of related works about scenario generation. Section "Scenario generation pipeline" presents the proposed scenario generation pipeline. Section "Experiments and analysis" shows the experiments and results. Section "Conclusion" concludes the paper.

Related works

In this section, existing scenario generation methods are discussed. Even though scenario generation has been extensively studied, there are few studies involving autonomous driving simulators. The discussed scenario generation methods are related to various domains and are classified as heuristic-based methods, annotation-based methods, script-based methods, and graphical-user-interface (GUI)-based methods.

Heuristic-based methods are often used in scenario generation. These can be divided into rule-based methods and genetic algorithm (GA)-based methods. In rule-based methods, usually a set of constraints between the actions of a character is defined [15], and a multi-layered architecture is applied to model a character's behavior in a virtual environment [16]; in addition, a cognitive model is included to locate the composite actions in action sequences. These methods focus on decision making without considering the movements of the agents. In Ref. [17] a planning-based scenario generation system for describing hierarchical tasks focusing on generating a partially perturbed environment instead of character actions was described. In Ref. [18], scenarios were generated by specifying scenario generation rules using functional L-systems. In the studies reported in Refs. [17, 18], scenarios were generated in an offline way that cannot dynamically change with the simulation status. In contrast, in the study in Ref. [19], a heuristic search technique was used to generate complicated multi-actor behaviors. However, this approach cannot accept user input and requires the user to have intensive domain knowledge. In contrast, in Ref. [20], the generation of different character behaviors based on personalities in earthquake scenarios was proposed. However, the evacuation points of the scene were predefined in an extensible markup language (XML) file, and the preparation of these files is time-consuming. The simulation of agent behavior in emergency scenarios usually focuses on movement only [21–25]. GAs have also been utilized for scenario generation [4, 6, 26–28] where they have been used to search for scenarios that maximize a set of evaluation criterion. However, in these approaches, scenario generation occurs in an offline manner.

Annotation-based scenario generation methods are also frequently used. For example, in Ref. [29], situations annotated with the actions that characters can perform were used. The characters can carry out basic actions, and, when they enter a new situation, additional actions allowed by the new situation are added to the characters using a probabilistic mechanism to ensure that the characters react appropriately. In Ref. [30], a similar method of scenario generation was reported. However, the environmental objects were annotated with character actions. The advantage of annotation-based scenario generation is that it makes action planning in the virtual environment very efficient and straightforward. Nevertheless, it requires significant work to make annotations when the environment is very large, and only rigid and repeatable scenarios can be generated.

Some researchers have utilized script-based methods. For example, in Ref. [5], the ATTAC tool was introduced to allow non-technical users to create scenarios. The ATTAC-L modeling language can translate the user-specified scenario into XML files that can be interpreted by a game engine. In addition, in Refs. [31, 32], the PRESTO script, which can be used to describe the behavior of non-player characters (NPCs) in a virtual environment, was introduced to control multi-agent actions. The script-based approach uses a predefined set of sequences to describe the scenario, so significant

modifications to the scripts are often required when small changes need to be applied. Therefore, script-based methods cannot be easily generalized in different scenarios.

There are also methods based on graphical user interfaces (GUI). For example, in Ref. [33], a visual authoring tool, CANVAS, which allows users to make multi-actor scenario within minutes, was presented, and, in Ref. [34], a user-centered interface to model scenarios on driving simulators was proposed. In contrast, Ref. [35] focused on generating scenarios containing cooperative tasks. However, the trajectories of agents are specified by users, which is tedious work. Thus, although GUI-based scenario generation allows non-technical users to create scenarios, the manual authoring of all the scenarios is a time-consuming and tedious process.

In this paper, we present an efficient scenario generation pipeline in a virtual environment for autonomous driving. In the scenario module, scenarios are classified as custom or automatic scenarios. The custom scenario is under the full control of the user via a GUI-based authoring tool. GUI-based event generation allows users to produce creative scenarios. In contrast, automatic scenarios are generated automatically considering the status of the surrounding agents in a large virtual environment. The custom scenario will not be introduced in this paper because many researchers have already covered this area [15, 34]. However, automatic event generation will be illustrated in Sect. "Scenario generation pipeline". To ensure the integrity and diversity of the scenario, an event-centric action dispatcher module is proposed.

Scenario generation pipeline

In this section, the scenario generation pipeline is explained. Given that a virtual environment should provide sufficient space for autonomous driving, the simulation of the whole environment is not feasible, not just because of computational limitations but because large scenarios are difficult to manage [36]. Therefore, the scenario simulation focuses on generating one scenario near the autonomous vehicle. The scenario generation pipeline is focused on the scenario map generator, activation of the scenario node, CNN-based scenario agent selector, and event-centric action dispatcher.

Scenario generation pipeline

The scenario generation pipeline is shown in Fig. 1. The simulator is divided into the *virtual environment* and *control system*. The *virtual environment* provides the simulation environment, and the *control system* controls the movement of agents in the virtual environment.

The location and direction of the autonomous vehicle is calculated by the *autonomous vehicle monitor*. The *scenario node finder* finds the nearest scenario node that lies on the road that the autonomous vehicle is on or about to be on; this is achieved by utilizing a scenario map. The *collision handler* delivers the collision messages from the *virtual environment* to the scenario module. The destination positions of the chosen agents are decided by the *target position generator* utilizing heuristic methods based on the relative location and direction of the selected agents and the structure of the surrounding roads. The information about roads and agents in an area is extracted by an *information extractor* given the center position and radius of the target area. The basic execution of agent actions is controlled by the *agent basic action controller* when they are not within the

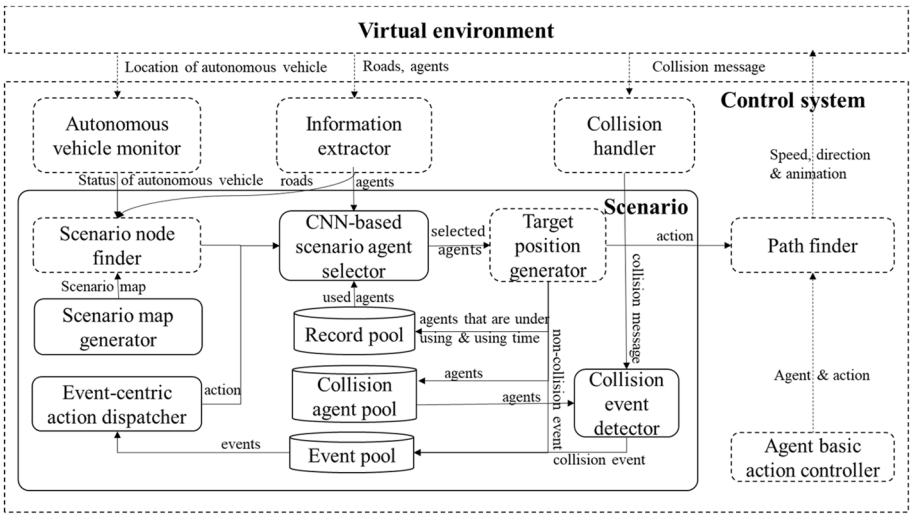


Fig. 1 Overview of the scenario generation pipeline

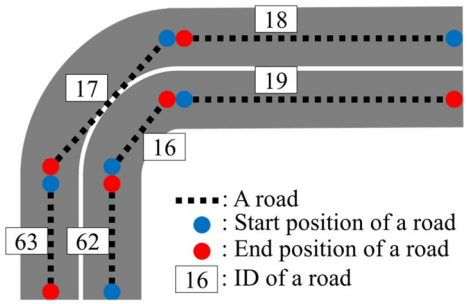


Fig. 2 Representation of a curved road

area of activated scenario node. The *path finder* generates control information, including the speed, direction, and animation, for controlling the movement of the agents in the virtual environment.

Scenario map generator

To generate scenarios on roads, a scenario map is generated. Let $R = (p_s, p_e, R_p, R_n, J_p, J_n)$ denote a road in the virtual environment, in which p_s and p_e denote the start- and end-points of road R, respectively, R_p and J_p denote the neighboring road and junction, respectively, connected to p_s, and R_n and J_n denote the neighboring road and junction, respectively, connected to p_e. The vehicles in the virtual environment should drive from the start-point to the end-point of a road. Figure 2 shows a representation of a curved road. The shape of a curved road is rounded but represented by multiple straight lines.

A junction is used to solve the situation in which a road is connected to multiple roads. A junction consists of a set of road pairs. Let $J = \{(R_i, R_o)_1, (R_i, R_o)_2, \ldots, (R_i, R_o)_N\}$ denote a junction for which R_i is the road whose end-point connects to the junction, and R_o is the road whose start-point is connected to J. Vehicles can move from R_i to R_o, and,

at a junction, a vehicle may choose one road to follow from multiple roads, as shown in Fig. 3.

The roads in the virtual environment can, thus, be connected with each other to generate a directed graph. The directed graph generation process is shown in Algorithm 1. First, all roads are analyzed to obtain the road pairs. One road pair includes two neighboring roads in which the end-point of the first road is connected to the start-point of the second road. All road pairs are added to the graph iteratively. Then, a directed graph is created in which a vertex denotes a road. A list is associated with a vertex to save adjacent roads whose start-points are connected to the end-point of the vertex.

Algorithm 1: Directed graph generation of roads
Input: All roads, junctions
Output: Directed graph G
Create empty list road_pairs
repeat
Add road pair (R_p, R) to road_pairs if R_p exists
Add road pair (R, R_n) to road_pairs if R_n exists
until all road are analyzed
repeat
Add all road pairs of (R_i, R_o) in current junction to road_pairs
until all junctions are analyzed
Create graph G
repeat
Add first road of a pair into G as a vertex
Add second road of a pair into the list that is associated with the vertex of first road
until all road pairs are added into directed graph

The scenario map is a set of scenario nodes. Its generation involves traversing the graph. The depth-first search algorithm is utilized to traverse through the directed graph. Let d be the distance between two neighboring scenario nodes. Points are sampled on the roads in intervals of distance d. At the position of the sampled point, a scenario node is created. When the remaining length, d_r, of a road is not sufficient to sample a point, the length of the road next to it is added to d_r to calculate the first scenario node. Because junctions are ignored in the calculation of distance in this method, the actual distance between two scenario nodes can be longer than d. During generation, the identities (IDs) of scenario nodes generated on each road are saved. During simulation, the scenario only occurs around the autonomous vehicle. Given the ID of the road

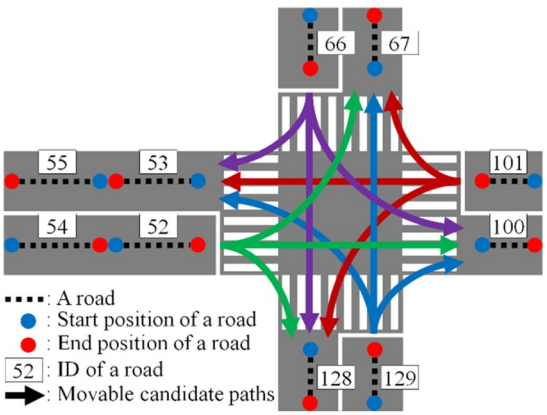


Fig. 3 Relationship representations of roads and junctions

that the autonomous vehicle is currently running on, the scenario nodes that are close to the autonomous vehicle can be efficiently found. Each scenario node is assigned with an initial action. The probability of being chosen for each initial action can be configured by the user. When a scenario node is activated, the initial action of a scenario node is scheduled.

CNN-based scenario agent selector

When an action is chosen and executed by a scenario node or event-centric action dispatcher, the appropriate agents for the execution of the action must be determined. The choice of the right agent to perform the action is critical for achieving a natural scenario simulation. For example, when the scenario requires the vehicle to perform a *hithuman* action, the human should not be behind the vehicle because the vehicle cannot change its orientation significantly in a short time. To ensure that selected agents can perform a given action naturally, we use CNN to evaluate the rationality of the selected agents because CNN has been shown to perform classification tasks well [37].

The input data format of the CNN is an image. However, to reduce the amount of data transfer between the virtual environment and the control system, only the raw data containing the state of the agents and roads is transferred to the control system. To convert the received raw data into a pictorial format, a top-view image of the entire virtual environment is pre-read into memory before simulation. The input image is generated by cropping a portion of the top-view image and placing the current scenario node on its center position. The width and height are both the same as the diameter of the scenario node. The agents near the scenario node are also pasted into the image based on their positions and directions using repository resources of vehicles, animals, and pedestrians.

The CNN structure is illustrated in Fig. 4. We define the structure of the building block as Conv-ReLU-Pooling-Batch normalization. The CNN consists of five building blocks followed by dropout and flatten layers, as well as two groups of Dense-ReLU-Batch normalization layers and a dense layer. Then, the output is fed into a Softmax layer to output the classification result. The building block incorporates a convolutional layer, a ReLU action layer, a max pooling layer, and a batch normalization layer. The number of filters, n, is chosen empirically.

Collision event detector

An event is a situation resulting from agent-executed actions. Different reactions can be generated based on different events. There are two types of events based on the response

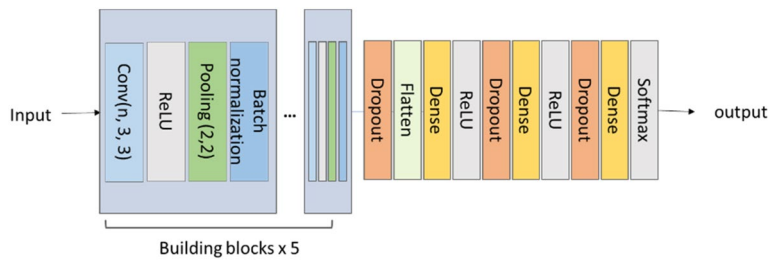


Fig. 4 Structure of the CNN

time of the surrounding agents. The first type of event is a non-collision event, which occurs directly when the owners of the actions are executing the actions. Non-collision events include quarreling, fighting, and dancing. When the agents execute these kinds of actions, the surrounding agents can react directly. Another type of event is an accident event. The triggering of this event requires the owners of the actions to finish the actions, for example, collision between vehicles. For the former, the events related to the actions are directly added to the event pool. For the latter, the agents who execute the actions related to the collision event are saved in a collision agent pool. Agents in the collision agent pool have an expiration time, and, when the expiration time is up, the agent is removed from the pool. The scenario module must monitor the execution status to generate accidents. Thus, a collision event detector is proposed. The collision event detector processes collision messages. A collision message contains agents that collide together, as well as the position where they collide. When the same agent appears in the collision agent pool and collision message, the actions of the agents executed by the scenario have caused an accident. Thus, the agent will be removed from the collision event detector, and the collision message is also removed. Then, a new event is generated and added to the event pool.

Event-centric action dispatcher

When there are events in the event pool, the agents surrounding the position of these events are expected to react appropriately. The event-centric action generator assigns actions to the surrounding agents through a series of actions that are pre-grouped by event categories. The purpose of this is to promote the development of scenarios by dispatching actions related to events that occur near the autonomous vehicle. This is achieved by either recognizing actions in videos related to traffic using existing reports [38, 39] or intuitive authoring.

The process of generating the corresponding actions for surrounding agents according to the event is intuitive. The scenario module iteratively analyzes the existing events and collects the surrounding agents in the place where an event is generated. Random reactions are assigned to these agents, and the used agents and the execution times for the reactions are recorded to prevent the repeated assignment of new reactions.

Experiments and analysis

Because there are no autonomous driving simulators in virtual environments featuring scenario generation, we could not compare the performance of our pipeline with those of others. In this section, the experimental methods and results are given first, and, subsequently, the experimental analysis is presented.

Experimental methods

The simulation experiments were carried out on a desktop computer with an i7-6700 3.40 GHz CPU and a NVIDIA GeForce GTX 1060 graphics card. The virtual environment was developed in Unity, and the control system was implemented in Python. The communication between virtual environment and the control system was achieved by transmission control protocol (TCP).

To verify the proposed scenario generation pipeline, we show the results of the scenario map generator, the CNN-based scenario agent selector, and the event-centric action dispatcher. Because the collision event detector serves the event-centric action dispatcher, the performance can be evaluated with the event-centric action dispatcher. The CNN-based scenario agent selector and event-centric action dispatcher are addressed in the experimental section.

CNN-based scenario agent selector

Although there are many free surveillance videos containing traffic scenes on the Internet, we found it difficult to obtain training data from these videos. Specifically, because of the different camera shooting angles, the quality of these videos is uneven, and it is not possible to provide reliable training data for the CNN model. Therefore, the training data were collected from the simulator and labeled by an expert. Because CNN requires image input, sending images between Unity 3D and Python slows the simulation; thus, we used the method described in "Scenario generation pipeline" and "Experiments and analysis" to synthesize input images in Python. The range of the collected data was 64 m by 64 m, correspondingly, the size of the generated input image was 128 by 128. The collected data stores information about agents and roads surrounding the autonomous vehicle, and the information about the agent includes the location and direction, whereas the road information includes the road start-point, end-point, and direction. We grouped the actions that can cause events into three categories based on the agent type, as shown in Table 1. The movements of pedestrians and animals are similar and consider their orientations, but their movements are different from those of vehicles.

By letting the same event happen repeatedly in the virtual environment, we collected data for the three events. We randomly selected agents from the agents in the vicinity of the autonomous driving vehicle according to the types of agents needed for the event; this allowed us to include a range of cases. The data were artificially labeled as positive or negative to determine whether the selected agents participating in the event were appropriate. Most of the data were labeled as negative and randomly discarded to equalize the data set. After labeling, we obtained the training data set and the test data set, as shown in Tables 2 and 3, respectively.

Definition of event-centric actions

We grouped the partial actions of agents into three groups based on the designed consequences. The response actions are defined by analyzing related videos, for

Table 1 Grouping of actions by agent type

Category	Name	Subject type	Target object types
Action type 1	Take vehicle	Human	Vehicle
	Be hit	Human	Vehicle
	Run to vehicle	Human	Vehicle
Action type 2	Fight	Human	Human
	Quarrel	Human	Human
	Be attacked	Human	Human, animal
	Talk	Human	Human
Action type 3	Hit	Vehicle	Human, vehicle, animal

Table 2 Training data set

Category	Positive data	Negative data	Total
Action type 1	482	620	1102
Action type 2	718	906	1624
Action type 3	482	618	1100

Table 3 Testing data set

Category	Positive data	Negative data	Total
Action type 1	40	60	100
Action type 2	46	54	100
Action type 3	39	61	100

Table 4 Source actions of events

Event type	Human	Vehicle
Accidents	Be hit, cross the road, take vehicle, run to vehicle	Hit, turn, drive forward, drive backward, change lane, rush to accelerate, rush to decelerate, rush to brake, lose control, stop
Conflict	Fight, quarrel	
Show	Sing, dance	

Table 5 Response actions to events

Event type	Human	Vehicle	Animal
Accidents	Get out of vehicle, run, walk, cross the road, take a picture, wait, check car, fight, phone call, check injured, talk	Drive forward, stop	Run
Conflict	Run, walk, take a picture, wait, phone call, cheer, check injured, talk, get out of vehicle	Drive forward, stop	Run
Show	Wait, walk, cross the road, get out of vehicle, take a picture, cheer	Drive forward	Run

example, we analyzed accident videos to summarize typical responses. The results are shown in Table 4. For example, the *run to vehicle* and *vehicle hit* actions may result in accident events, the *quarrel* and *fight* actions cause conflict events, and the *sing* and *dance* actions may result in a show event because they would attract other agents.

To allow the agent to react realistically to an event that has occurred, we defined the response actions of events, as shown in Table 5. For example, when an accident occurs, the driver may get out of the vehicle to check the injured pedestrian, check the car, stop, or continue driving.

Experimental results

In this section, we show the results of the scenario map generator, CNN-based scenario agent selector, and event-centric action dispatcher. Because collision event detection serves the event-centric action dispatcher, it is not evaluated here.

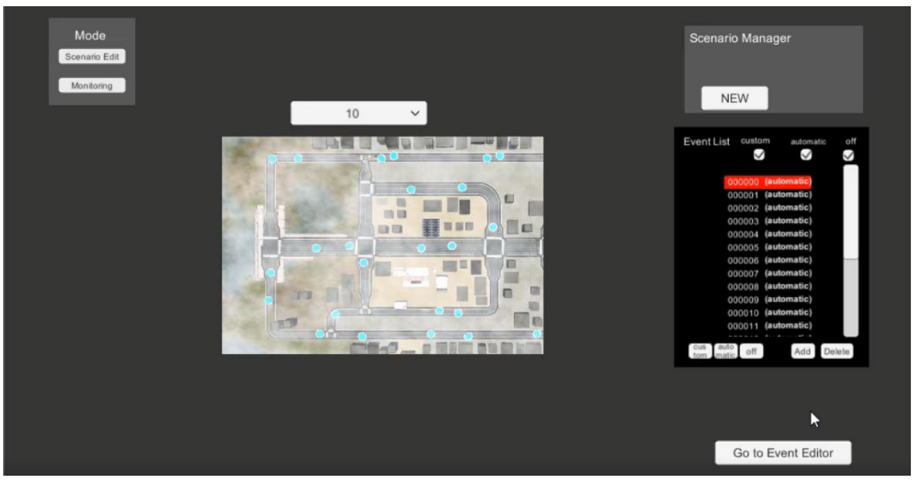


Fig. 5 Generated scenario map

Table 6 Accuracy compared with SVMs having different kernels

Methods	Action type 1 (%)	Action type 2 (%)	Action type 3 (%)	Average (%)
SVM(L)	71	64	68	67.67
SVM(P)	78	73	79	76.67
SVM(RBF)	86	74	78	79.33
CNN	90	96	94	92.67

Results from the scenario map generator

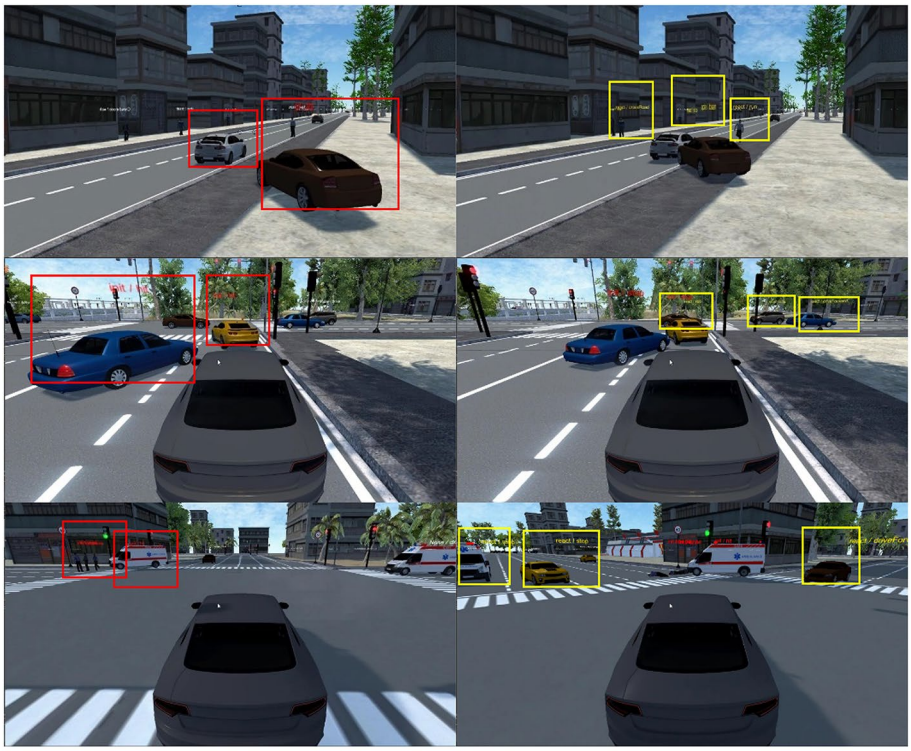
The output of the scenario generator is shown in Fig. 5. Each blue point in the scenario map represents one scenario node. To reduce computational cost, the scenario is only simulated in the region surrounding the scenario node when the autonomous vehicle enters the scenario area. The right part of the image shows the list of scenario nodes. The user can choose a scenario node to change the type of scenario node to a custom scenario.

Evaluation of CNN-based scenario agent selector

To verify the performance of the CNN-based scenario agent selector, we compared the predicted accuracy with three kinds of support vector machines (SVMs) based on different kernel functions using the testing data. The input of the SVMs is different from that of the CNN, i.e., a vector including information about agents and roads around the autonomous vehicle, as mentioned in "Results from the scenario map generator" section. The results of this comparison are shown in Table 6. The results show that the radial basis function (RBF)-based SVM performed better than the linear function(L)-based SVM and polynomial function(P)-based SVM. The performance of the CNN on the test data sets was significantly better than those of the SVM-based methods.

a Agents selected by CNN. **b** Accident after execution.

Fig. 6 Execution of an initial action in one scenario node. **a** Agents selected by CNN, **b** Accident after execution



a Selection of scenario agents. **b** Actions of surrounding agents.

Fig. 7 Execution of the event-centric action dispatcher. **a** Selection of scenario agents, **b** Actions of surrounding agents

The trained model was applied in the virtual environment to select agents in the generated scenario. In Fig. 6a, the agents selected by CNN are labeled with red boxes. Figure 6b shows the accident event after execution.

Evaluation of event-centric action dispatcher

Figure 7 shows examples of accidents simulated in the virtual environment using the proposed event-centric action dispatcher. In the left column of Fig. 7, two vehicles were selected as the scenario agents (shown in the red box). After execution, an accident occurred, and the scenario generation module received the collision message and

triggered the actions related to the accident involving the surrounding agents (shown in the yellow box in the right column of Fig. 7).

Experimental analysis

The scenario map generated by the scenario map generator distributes scenario nodes over the whole map. The positions of the scenario nodes in the scenario map vary because of the randomizing algorithm used at the beginning of scenario generation, which guarantees that different terrains are covered over repeated simulations. The proposed CNN-based scenario agent selector accurately determined whether the agent selection generated a realistic scenario. Furthermore, the trained CNN model is more accurate than the output of the three SVMs, having an average accuracy of 92.67%. From the right column of Fig. 7, we know that the proposed event-centric action dispatcher module allows natural responses from agents close to the corresponding event locations.

Conclusion

In this paper, we have proposed a pipeline that generates various scenarios for autonomous vehicle simulations. The key contributions of the proposed pipeline are the training of a CNN for the selection of appropriate agents and the generation of realistic scenarios involving pedestrians, animals, and vehicles and the application of an event-centric action dispatcher module to generate related actions for agents surrounding the event location in real time. We simulated various scenarios in a virtual environment by not only generating the scenarios but also generating the related actions of the surrounding agents to ensure that the simulation conforms to reality. For real time performance, the scenario simulated only the environment surrounding the autonomous vehicle, and this was achieved by generating a scenario map. The experiments showed that the CNN-based scenario agent selector can achieve a high accuracy of 92.67%. With the assistance of the event-centric action dispatcher module, the scenario generation pipeline successfully generated convincing scenarios in the virtual environment designed for autonomous driving. The experimental results show that this pipeline can be utilized to generate scenarios in a virtual environment for autonomous driving. Ultimately, the actions dispatched by the event-centric action dispatcher are classified based on human experience. However, manual summarization is time-consuming when additional events are needed. In the future, approaches applying an action recognition model via deep learning to video data classified based on event categories to recognize the action set related to corresponding event automatically will be our priority.

Acknowledgements
Not applicable.

Authors' contributions
MW wrote the source codes and the manuscript. JP provided action data for generation of scenario. KC provided full guidance. All authors read and approved the final manuscript.

Funding
This research was supported by a grant from Defense Acquisition Program Administration and Agency for Defense Development, under contract #UE171095RD and the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program)(2019-0-01585) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 28 September 2019 Accepted: 29 April 2020
Published online: 03 June 2020

References

1. Kanade T, Thorpe C and Whittaker W (1986) Autonomous Land Vehicle Project at CMU. Proc. 1986 ACM Computer Conference, February, pp. 71-80
2. Fridman L (2018) Human-centered autonomous vehicle systems: Principles of effective shared autonomy. arXiv preprint arXiv:1810.01835,2018, pp.1-9
3. Luo L, Cai W, Zhou S, Lees M, Yin H (2015) A review of interactive narrative systems and technologies: a training perspective. Simulation 91(2):126–147
4. Luo L, Yin H, Cai W, Zhong J, Lees M (2017) Design and evaluation of a data-driven scenario generation framework for game-based training. IEEE Transact Comput Intell AI Games 9(3):213–226
5. Janssens O, Samyny K, Van de Walle R and Van Hoecke S (2014) Educational virtual game scenario generation for serious games. 2014 IEEE 3nd International Conference on Serious Games and Applications for Health (SeGAH), Rio de Janeiro, 14–16 May, pp. 1-8
6. Zook A, Lee-Urban S, Riedl MO, Holden HK, Sottilare RA, Brawner KW (2012) Automated Scenario Generation: Toward Tailored and Optimized Military Training in Virtual Environments. FDG '12 Proceedings of the International Conference on the Foundations of Digital Games, 30 May-1 June, Raleigh, North Carolina, USA, pp. 164–171
7. Ros G, Sellart L, Materzynska J, Vazquez D and Lopez AM (2016) The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 27-30 June, pp. 3234-3243
8. Dosovitskiy A, Ros G, Codevilla F, Lopez A and Koltun V (2017) CARLA: An Open Urban Driving Simulator. Conference on Robot Learning (CoRL), Mountain View, California, 13-15 November, pp. 1-16
9. Shah S, Dey D, Lovett C and Kapoor A (2018) AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In: Hutter M., Siegwart R. (eds) Field and Service Robotics. Springer Proceedings in Advanced Robotics, vol 5. Springer, Cham, 12–15 September, Zürich, Switzerland, pp. 621–635
10. Wymann B, Espi´e E, Guionneau C, Dimitrakakis C, Coulom R and Sumner A (2015) TORCS, The Open Racing Car Simulator. http://www.torcs.org, pp. 1-5
11. Müller M, Casser V, Lahoud J, Smith N, Ghanem B (2018) Sim4CV: a photo-realistic simulator for computer vision applications. Int J Comput Vision 126(9):902–919
12. Alireza S, Rahil H (2019) A state-of-the-art survey of malware detection approaches using data mining techniques. Hum Centric Comput Inform Sci 8(1):3
13. Hyejin S, Kihoon L, Nammee M (2019) User Modeling using user preference and user life pattern based on personal bio data and SNS data. J Inf Process Syst 15(3):645–654
14. Zhoua L, Pana S, Wanga J, Vasilakosb AV (2017) Machine learning on big data: opportunities and challenges. Neurocomputing 237:350–361
15. Paris S and Donikian S (2009) Activity-Driven Populace: A Cognitive Approach to Crowd Simulation. in IEEE Computer Graphics and Applications, 21 July; 29(4): 34–43
16. Lim CK, Tan KL, Zaidan AA, Zaidan BB (2019) A proposed methodology of bringing past life in digital cultural heritage through crowd simulation: a case study in George Town Malaysia. Mult Tools Appl 19:1–37
17. Hullett K and Hullett K (2009) Scenario generation for emergency rescue training games. FDG '09 Proceedings of the 4th International Conference on Foundations of Digital Games, 26–30 April, Orlando, Florida, pp. 99-106
18. Martin GA, Hughes CE, Schatz S and Nicholson D (2010) The use of functional L-systems for scenario generation in serious games. PCGames '10 Proceedings of the 2010 Workshop on Procedural Content Generation in Games, 18 June, Monterey, California, pp. 1-5
19. Kapadia M, Singh S, Reinman G and Faloutsos P (2011) A Behavior-Authoring Framework for Multiactor Simulations. in IEEE Computer Graphics and Applications, November-December 2011; 31(6): pp. 45-55
20. Liu T, Liu Z, Ma M, Chen T, Liu C and Chai Y (2019) 3D visual simulation of individual and crowd behavior in earthquake evacuation. Simulation: Transactions of the Society for Modeling and Simulation International 2019; 95(1): pp. 65-81
21. Liu Z, Liu T, Ma M, Hsu H H, Ni Z and Chai Y (2018) A perception-based emotion contagion model in crowd emergent evacuation simulation: Computer Animation and Virtual Worlds; 29(3-4): pp. e1817
22. Başak A E, Güdükbay U and Durupınar F (2018) Using real life incidents for creating realistic virtual crowds with data-driven emotion contagion: Computers & Graphics; 72: pp. 70-81.
23. Xu M, Xie X, Lv P, Niu J, Wang H, Li C, Zhu R, Deng Z and Zhou B (2019) Crowd behavior simulation with emotional contagion in unexpected multihazard situations: IEEE Transactions on Systems, Man, and Cybernetics: Systems; PP(99)(2019): pp.1-15
24. Chen L, Jung C R, Musse S R, Moneimne M, Wang C, Fruchter R, Bazjanac V, Chen G and Badler N I (2018) Crowd simulation incorporating thermal environments and responsive behaviors: PRESENCE: Teleoperators and Virtual Environments; 26(4): pp.436-452
25. Lyu, L and Jinling Z (2018) Toward modeling emotional crowds: IEEE Access; 6: pp. 55893-55906
26. Luo L, Yin H, Cai W, Lees M, Zhou S (2013) Interactive scenario generation for mission-based virtual training. Comput Anima Virtual Worlds 24(3–4):345–354

27. Luo L, Yin H, Zhong J, Cai W, Lees M and Zhou S (2013) Mission-based scenario modeling and generation for virtual training. AIIDE'13 Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 14-18 October, Boston, MA, USA, pp. 44-50
28. Luo L, Yin H, Cai W, Lees M, Othman NB, Zhou S (2014) Towards a data-driven approach to scenario generation for serious games. Computer Anim Virtual Worlds 25(3–4):395–404
29. Sung M, Gleicher M, Chenney S (2004) Scalable behaviors for crowd simulation. Computer Graphics Forum 23(3):519–528
30. Maïm J, Haegler S, Yersin B, Mueller P, Thalmann D and Gool LV (2007) Populating ancient pompeii with crowds of virtual romans. VAST'07 Proceedings of the 8th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage, 26-30 November, Brighton, UK, pp. 109-116
31. Busetta P, Robol M, Calanca P and Giorgini P (2017) PRESTO Script: scripting for serious games. AI & Games Symposium at AISB 2017, 18-22 April, Bath, UK, pp. 1-6
32. Puel D (2018) An authoring system for VR-based firefighting commanders training: Electronic Imaging; 2018(3): pp. 469-1
33. Kapadia M, Frey S, Shoulson A, Sumner RW and Gross M (2016) Canvas: Computer-assisted narrative animation synthesis. In Proceedings of the ACM SIG-GRAPH/Eurographics Symposium on Computer Animation, SCA'16, Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. 11-13 July, pp. 199–209
34. Bhatti G, Brémond R, Jessel J-P, Dang N-T, Vienne F, Millet M (2015) Design and evaluation of a user-centered interface to model scenarios on driving simulators. Trans Res Part C Emerg Technol 50:3–12
35. Wong SK, Chou YH, and Yang HY (2018) A framework for simulating agent-based cooperative tasks in crowd simulation. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 15–18 May, Montreal, Quebec, Canada, pp. 1–10
36. Abdelgawad K, Henning S, Biemelt P, Gausemeier S, Trächtler A (2016) Advanced traffic simulation framework for networked driving simulators. IFAC-PapersOnLine 49(11):101–108
37. Krizhevsky A, Sutskever I and Hinton GE (2012) ImageNet classification with deep convolutional neural networks. NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, 3-6 December, Lake Tahoe, Nevada, USA, pp. 1097-1105
38. You SD, Chien-Hung L, Woei-Kae C (2018) Comparative study of singing voice detection based on deep neural networks and ensemble learning. Hum-Centric Comput Inform Sci 8(1):34
39. Min-Ji S, Myung-Ho K (2019) A system for improving data leakage detection based on association relationship between data leakage patterns. J Inf Process Syst 15(3):520–537

Publisher's Note