COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 04/06 pp 45–54
Volume 1, Number 1, March 2021
DOI: 10.23919/CSMS.2021.0004

3D Environmental Perception Modeling in the Simulated Autonomous-Driving Systems

Chunmian Lin, Daxin Tian, Xuting Duan*, and Jianshan Zhou

Abstract: Self-driving vehicles require a number of tests to prevent fatal accidents and ensure their appropriate operation in the physical world. However, conducting vehicle tests on the road is difficult because such tests are expensive and labor intensive. In this study, we used an autonomous-driving simulator, and investigated the three-dimensional environmental perception problem of the simulated system. Using the open-source CARLA simulator, we generated a CarlaSim from unreal traffic scenarios, comprising 15 000 camera-LiDAR (Light Detection and Ranging) samples with annotations and calibration files. Then, we developed Multi-Sensor Fusion Perception (MSFP) model for consuming two-modal data and detecting objects in the scenes. Furthermore, we conducted experiments on the KITTI and CarlaSim datasets; the results demonstrated the effectiveness of our proposed methods in terms of perception accuracy, inference efficiency, and generalization performance. The results of this study will faciliate the future development of autonomous-driving simulated tests.

Key words: autonomous-driving system; environmental perception; simulated test; deep-learning model

1 Introduction

An autonomous-driving system comprises a comprehensive software and hardware platform that provides a wide spectrum of safety-related functions and requires extensive testing to ensure safety prior to large-scale production. However, research on urban autonomous-driving involves large infrastructure costs and poses challenges in real-world system tesing. Specifically, the construction and operation of automated vehicles require huge investments of funds and manpower. In addition, conducting vehicle tests in urban environments is dangerous, cumbersome, and time-consuming. For instance, model tesing and validation of environmental perception require extensive labeling of urban scene data covering as many traffic scenarios as possible, which is costly and labor-intensive. These challenges severely hinder the practical application and development of automated vehicles in the physical world.

An alternative is to use simulation software to train and validate driving strategies, which can facilitate autonomous-driving research. Simulation is also useful for system testing as certain traffic scenarios are too dangerous to stage in the real world, e.g., an elderly person walking onto the road in front of a vehicle. With the recent advancements in computer graphics and rendering technology, several simulation frameworks[1–5] and racing game engines[6–9] have become available for use in autonomous-driving research. These simulators mimic large-scale traffic scenarios and real-world situations and can be used to evaluate new approaches to autonomous driving. Specifically, given sufficient computational resources, a large-scale urban dataset featuring diverse conditions can be quickly simulated. Data labeling can also

• Chunmian Lin, Daxin Tian, Xuting Duan, and Jianshan Zhou are with the School of Transportation Science and Engineering, Beihang University, Beijing 100191, China. E-mail: cmlin@buaa.edu.cn; dtian@buaa.edu.cn; duanxuting@buaa.edu.cn; jianshanzhou@foxmail.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2021-02-16; revised: 2021-02-27; accepted: 2021-03-01

© The author(s) 2021. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

be fully automated, eliminating the need for manual supervision[10]. Overall, these advantages enable efficient and effective system validation, and simulation is currently regarded as an important method for automated vehicle testing in unseen environments.

As the basis of the autonomous-driving system, the environmental perception module must provide accurate scene information to guide path planning and vehicle control. The majority of relevant studies have focused on the design of deep-learning based[11] perception models and the adoption of existing autonomous-driving datasets[12–14] for model training and testing. There are several limitations in the current research. First, although autonomous-driving datasets have been obtained from real-world scenarios, the limited amount of data leads to poor performance accuracy and robustness due to the lack of scene diversity. There have also been difficulties in developing an accurate perception model that can precisely detect objects in traffic scenarios. Therefore, future studies on autonomous-driving systems must focus on acquiring a diverse range of data from different urban scenes and developing a more powerful perception model.

To the end, in this study, we focused on autonomous-driving simulated systems and investigated the performance of an environmental perception model on the basis of simulated data. Specifically, we synthesized and generated data for large-scale traffic scenarios from various urban scenes using the CARLA simulator[15], which is an open source autonomous-driving simulator that can export high-quality and synchronized sensor data with annotations. The CARLA simulated dataset, i.e., CarlaSim, comprises 15 000 camera-LiDAR (Light Detection and Ranging) samples with accurate annotation and calibration files. We designed a Multi-Sensor Fusion Perception (MSFP) network that has a two-stream architecture with a feature fusion module for aggregating camera and LiDAR data. We conducted extensive experiments on the KITTI[12] and CarlaSim datasets; the results revealed the effectiveness of the simulated data and perception model. The contributions of this study can be summarized as follows:

(1) We constructed a large-scale simulated dataset named CarlaSim, which contains 15 000 images and point-cloud data generated from a diverse range of traffic scenes using the CARLA simulator.

(2) We developed an MFSP model, which consumes camera and LiDAR data separately, and then uses a feature fusion module to combine multiscale feature representations.

(3) We conducted validation experiments on the KITTI and CarlaSim datasets, the outstanding performance of which confirms the effectiveness of the proposed methods.

2 Related Work

2.1 Autonomous-driving simulator

Typically, a versatile autonomous-driving simulator includes a wide variety of traffic scenarios comprising fine-grained details and high-level scenes. Such simulators must satisfy the following requirements[16]: (1) simulate large-scale environments and guarantee both low-level details and high-level scenes; (2) provide simple integration and comparison of different sensors, actuators, and controllers; and (3) perform automated regression testing to enable agile and customized development.

Currently, a wide variety of simulation frameworks are available, each of which has pros and cons. PTV Vissim[1], Mathworks Simulink[2], and SUMO[3] provide data-import functionality from OpenStreetMap and can analyze traffic densities and weather conditions on multiplatform machines. However, they cannot simulate different cars, sensors, or controller models due to their purely high-level nature. Gazebo is a popular simulation framework[4] that enables the design of complex maps, but it cannot import OpenStreetMap and provides limited support for large-scale scenario simulations. Although the CarSim[5] simulator has its own physics engine, it runs only on the Windows system and does not support cross-platform machines. With the recent developments in computer graphics and rendering technology, several game engines have been developed and applied in autonomous-driving research. DeepDriving[6], a perception-based method for predicting affordance indicators for driving vehicles, directly maps an input image to several key indicators related to the cost of driving. These indicators also provide complete descriptions of urban scenes to enable the control of vehicle behavior. References [7, 8] emphasize the importance of creating large urban datasets with a pixel-wise label map. Reference [7] produced a pixel-level semantic map with a computer game and graphics hardware, whereas Ref. [8] created different high-resolution video frames annotated with ground-truth data for high-level and low-level vision tasks. TORCS[9] is a highly portable open-racing car simulator that supports cross-platform development and deployment (i.e., Linux, Windows, and FreeBSD). Originally developed as a car racing game, it has subsequently been used for autonomous-driving research.

The open-source CARLA[15] simulator has been developed from the ground up to support the development, training, and validation of autonomous-driving systems. It supports flexible specification and setup of sensor suites, environmental conditions, and full control of all the static and dynamic actors in autonomous-driving research. Moreover, different versions of CARLA have been released on various platforms, with functional APIs for system validation. In CARLA, three general approaches are used to evaluate autonomous-driving performance. First, the classic modular pipeline comprises a vision perception module, a rule-based planner, and a maneuver controller. Second, an end-to-end imitation learning strategy, which uses sensory input for training driving commands, is adopted. Third, deep reinforcement learning is used to explore and analyze end-to-end driving behavior. Due to its generalizability and practicality, we used the CARLA simulator to conduct our autonomous-driving system research.

2.2 Environmental perception

In an autonomous-driving system, environmental perception[17] includes object localization, offline obstacle and road mapping, moving-obstacle tracking, and traffic-signalization detection and recognition. In this study, we mainly focused on the multi-object three-dimensional (3D) perception problem in autonomous-driving systems. Generally, the role of environmental perception is to detect objects and derive their localization information from sensor data in the urban scene. Previous research on multiobject perception for autonomous driving can be roughly divided into the handcrafted feature algorithm and deep-learning[11] methods. Handcrafted feature algorithms depend heavily on professional knowledge and skills to extract representative features; thus, it is difficult to achieve satisfactory performance in complex environments using this algorithm. With the recent advent of the deep-learning method, a variety of 3D perception models have been designed for detecting attended objects in a scene using data from different sensors. This method can be further classified into camera-based, LiDAR-based, and multisensor fusion methods.

Camera-based methods perform 3D object detection directly from two-dimensional (2D) RGB images by first regressing the proposed 2D object and then predicting the 3D bounding-box result on the basis of its geometric relationship or constraint in 3D space[18–21]. The Mono3D[18] method infers a monocular 3D bounding box with semantic segmentation and contextual features (e.g., size, location, and shape). In Ref. [19], geometric constraints are imposed on a 2D bounding box and a 3D bounding box with the object pose which is produced from a single image. To mimic a LiDAR signal from images, pseudo-LiDAR[20,21] was defined, whereby a set of pseudo points is generated from an RGB image using a depth-estimation algorithm.

LiDAR-based detection is the mainstream method used for 3D object perception in autonomous-driving systems. This method uses a PointNet architecture to directly consume raw point-cloud data[22–24], or it transforms points into a regular voxel representation[25] and adopts a convolutional operation for 3D object detection[26,27]. PointNet architectures[22,23] take point-cloud data as input and apply a max-pooling operation to retain the permutation invariance of unordered points. On this basis, the TANet method[24] designs triple attention modules and a coarse-to-fine regression strategy for robust object localization. Reference [25] proposed VoxelNet, which uses a point-voxel encoding module that converts point-cloud data into a regular grid format for feature extraction. Subsequently, Ref. [26] reported the development of a 3D sparse convolutional operation to accelerate voxel feature encoding and 3D object detection.

The multisensor fusion method has emerged as a promising approach that uses multimodal input data (i.e., image and point-cloud data) and combines their respective feature maps using various fusion strategies to achieve a more accurate and robust detection[28–31]. MV3D[28] encodes sparse 3D point-cloud data with compact multiview representation and combines region features from multiple views for object detection in 3D space. The frustum PointNet[29] framework leverages a mature 2D object detector to generate region proposals and precisely estimates 3D bounding boxes under the constraint of the geometric relationship. AVOD[30] aggregates views for object detection in autonomous-driving scenarios. In the proposed network, a shared feature map is generated from point-cloud and RGB image data, and the multimodal feature maps are then fused for reliable 3D object proposals via a region proposal network. This method is effective for object detection in urban scenes. In this study, we developed an MSFP network based on AVOD architecture to realize 3D object detection in an autonomous-driving system.

3 Methodology

In this section, we introduce the simulated CarlaSim dataset and the proposed MSFP model, separately.

3.1 CarlaSim

To reduce the data-acquisition cost and facilitate autonomous-driving simulation testing, we use the CARLA simulator for generating high-quality urban scene data. CARLA includes various functionalities, i.e., LiDAR-to-camera projection, 2D and 3D bounding box labeling, and sensor calibration matrices. For simplicity, the formats of the annotation and calibration files are the same as those used in the KITTI benchmark; this makes them compatible with several existing object-detection models. Here, we set LiDAR and camera sensor suites in the CARLA simulated environment; these suites are placed and synchronized such that a full LiDAR rotation is possible in each image. In other words, this setting provides an approximate correspondence relationship between the points obtained from the LiDAR and camera sensors. Subsequently, the raw data are projected to a unified coordinate space; then, the visible object is found in the urban scene, and its coordinates are determined relative to those used in KITTI.

Free content in the CARLA simulator includes different urban layouts, various vehicle models, pedestrians, buildings, and traffic infrastructure. A variety of environments can also be specified, including weather conditions, illumination, and time of day. The simulation platform also provides signals, such as the GPS coordinates, speed and acceleration information, density of vehicles, collision data, and data regarding other infractions, which can be used to train driving behavior. The CARLA simulator also provides server-client environment simulation and a functional interface between the world and an agent. The server performs the simulation and renders the scenarios, and the client implements interactions between the server and agent via socket communication.

In our data generation research, we chose Town$_{01}$ and Town$_{02}$ in the CARLA simulator as the scenes. In total, the CarlaSim dataset comprises 15 000 image-point pairs, which are used to simulate real-world traffic scenarios as shown in Fig. 1. In comparison with the mainstream autonomous-driving dataset, i.e., KITTI, our CarlaSim dataset has approximately 18 624 cars and 6872 pedestrians, which is less than the KITTI benchmark. We note that the KITTI data were acquired from busy traffic scenarios in urban environments, whereas our CarlaSim dataset was generated from a suburban environment, in which the distribution of objects is relatively sparse and simple. Furthermore, the instances generated in the CarlaSim dataset have smaller object sizes and scales, which make them harder to detect. The sparsity and variety of the CarlaSim dataset make it useful for perception model validation in autonomous-driving research.

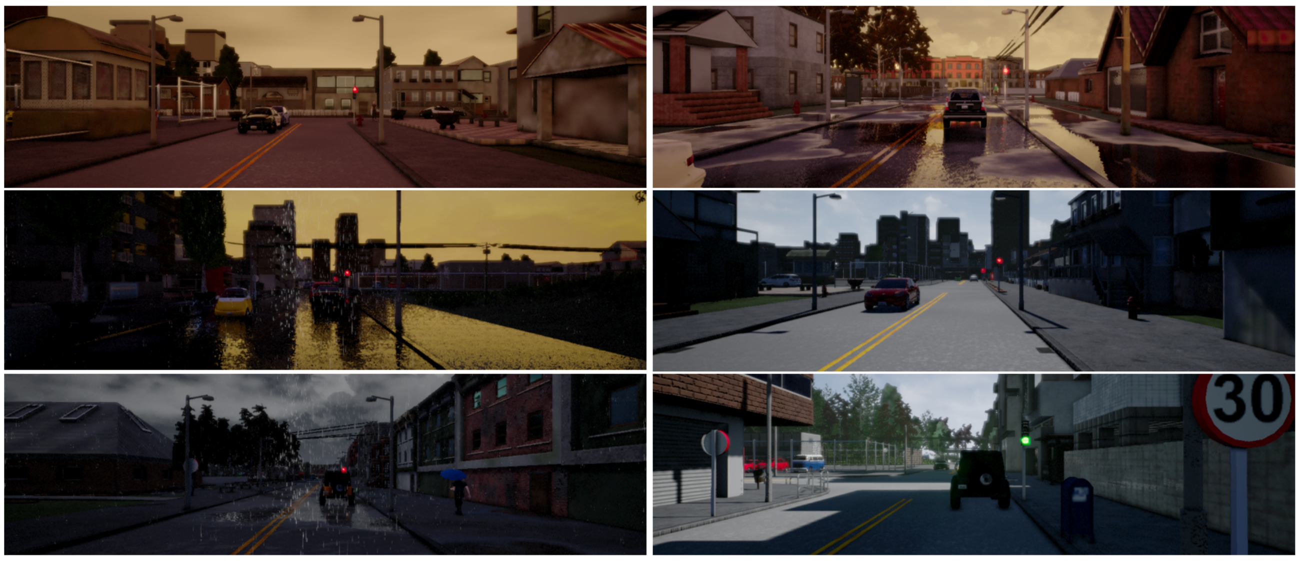

Fig. 1 Generated samples in the CarlaSim dataset. Noted that images in the left and right columns are generated from Town$_{01}$ and Town$_{02}$ scenes in the CARLA simulator, respectively.

3.2 MSFP

Based on the AVOD[30] model, we designed an MSFP network for camera-LiDAR object detection. Figure 2 shows the overall model architecture, which mainly comprises camera and LiDAR streams, feature fusion module, and Region Proposal Network (RPN).

3.2.1 Camera and LiDAR streams

Camera and LiDAR streams are used for processing image and point-cloud data, respectively. Prior to feature extraction, we perform preprocessing operations on the point cloud. Specifically, the point cloud is projected to six-channel Bird's-Eye View (BEV) maps from a voxel grid representation at 0.1 m resolution and is then cropped at [–40 m, 40 m]×[0 m, 70 m] to points within the camera's field of view. The first five channel BEV maps are encoded using the maximum heights of points in each grid cell, and the remaining BEV map contains density information of the points in each cell.

Next, we adopt the same feature extractors for the image and point-cloud data, which have an encoder-decoder architecture for aggregating multiscale feature representations. The encoder has four VGG[32]-style convolutional layers, with half the number of channels. Taking an $M \times N \times D$ image from the BEV map as example, the size of outputting feature map is eight-time lower than that of input in the encoder. For the decoder, we developed a bottom-up architecture[33] that hierarchically upsamples the feature map to the original input size. The decoder takes the output of the encoder as an input and generates a new $M \times N \times D$ feature map via convolutional-transpose and feature-concatenation operations. The final feature map has high resolution and representational power and is shared by the subsequent RPN.

3.2.2 Feature fusion module

To combine the multimodal feature representations, we introduce a feature fusion module that performs crop and resize operations to calibrate and fuse intermediate feature maps. Given an anchor box in 3D, we adopt a 1×1 convolution for processing the image and point feature maps, which significantly reduces the dimensionality and computational cost. To some extent, the 1×1 convolution also learns to select discriminative features that contribute significantly to the generation of region proposals. Subsequently, regions of interest are obtained by projecting the anchor onto the image and BEV feature maps. Then, corresponding regions are adopted for extracting feature-map crops from each view. The bilinear method is then used to resize feature vectors of equal length, and the cropped feature map follows the aspect ratio of the projected anchor box and provides a more reliable feature-crop result.

3.2.3 RPN

The RPN consumes the fused feature map and generates 3D region proposals for object detection. Similar to the 2D object detector[34, 35], we introduce the RPN architecture to regress the differences between the anchor and ground-truth boxes. These anchor boxes are parameterized as their center points $(x_a, y_a, z_a, l_a, w_a, h_a, \theta_a)$, and the corresponding ground-truth points are denoted as $(x_g, y_g, z_g, l_g, w_g, h_g, \theta_g)$, where (x_*, y_*, z_*) is the centroid coordinates of bounding boxes in 3D space, (l_*, w_*, h_*) defines the size of the box, and θ_* is the yaw angle along the z-axis. As mentioned above, the output feature crops are equal-size vectors from both streams, whose element-wise means are fused. We used a fully connected layer with 256 channels

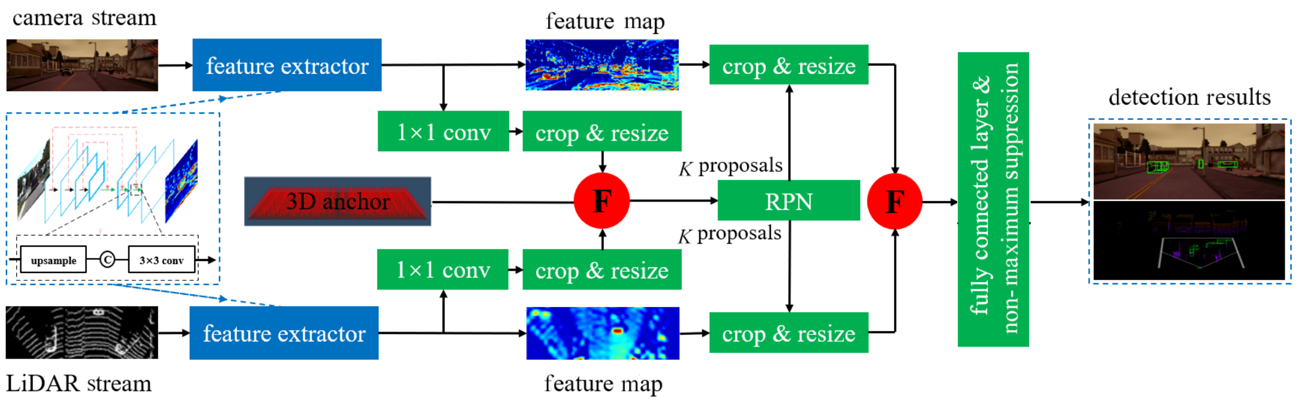

Fig. 2 Architecture of MSFP model. It is noted that the capital "F" denotes element-wise fusion operation between image and point feature maps.

for performing the 3D bounding-box regression and obtaining the objectness score. The coordinates of the box are regressed by computing the difference, $\Delta r = (\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta\theta)$, between the anchor and ground-truth boxes, as demonstrated in the following:

$$\Delta x = \frac{x_g - x_a}{d_a}, \Delta y = \frac{y_g - y_a}{d_a}, \Delta z = \frac{z_g - z_a}{d_a},$$
$$\Delta l = \log\left(\frac{l_g}{l_a}\right), \Delta w = \log(\frac{w_g}{w_a}), \Delta h = \log\left(\frac{h_g}{h_a}\right),$$
$$\Delta\theta = \theta_g - \theta_a \tag{1}$$

The objectness score prediction distinguishes the output belonging to the targeted object or background. We use Non-Maximum Suppression (NMS) method to filter out redundant proposals and select the top 1024 proposals during model training and inference.

4 Experiment

In this section, we conducted case studies to test the model perception performance using the real-world and simulated datasets. Then, we performed a comprehensive analysis of the results to determine the effectiveness of the CarlaSim dataset and MSFP model.

4.1 Training details

We adopted the KITTI and CarlaSim datasets for model training and evaluation. The KITTI dataset, which is a widely used autonomous-driving benchmark, contains 7481 training samples and 7518 test samples. We assigned 3712 samples for training and 3769 samples for validation, as commonly done in the work[28]. We then randomly halved the CarlaSim dataset for model training and validation. For evaluation, we considered three difficulty levels (i.e., easy, moderate, and difficult), as proposed by KITTI, and simultaneously computed the Average Precision (AP) values for the car and pedestrian classes. Specifically, AP_{3D} denotes the mean 3D detection precision value of all classes, and AP_{BEV} denotes the average precision value of all classes in the BEV. To ensure a fair comparison, we used 40 recall points instead of 11, as demonstrated in the study of Ref. [12], and we set the Intersection-of-Uion (IoU) threshold to 0.7 and 0.5 for the car and pedestrian classes, respectively.

We trained the MSFP model for 120 000 epochs using the ADAM[36] optimizer with an initial learning rate of 0.0001, which decays exponentially every 30 000 iterations with a decay factor of 0.8. We used the cosine annealing strategy to realize more stable training and smooth optimization. Considering the difference in modality between a camera image and a LiDAR point, it is difficult to guarantee correct correspondence between a pixel and a point after spatial augmentation transformation. Therefore, we performed no data augmentation during model training and tested the perception performance from scratch.

For the model training loss, we employ multi-task loss function, that comprises of Smooth-L1 loss for 3D bounding-box regression and cross-entropy loss for objectness score prediction, respectively. The respective equations are shown in the following:

$$L = L_{reg} + L_{cls} \tag{2}$$

$$L_{reg} = \frac{1}{N_{pos}}\sum_i smooth_{L_1}(r, \Delta r) \tag{3}$$

$$L_{cls} = \frac{1}{N_{box}}\sum_i -\alpha(1-p_i)^{\gamma}\log(p_i) \tag{4}$$

where L_{reg} and L_{cls} are box regression and classification loss, respeclively, N_{pos} is the number of positive boxes, N_{box} is the total number of boxes, p_i denotes the objectness score for the i-th box, and hyperparameter $\alpha = 0.25$ and $\gamma = 2$.

4.2 Experimental results

First, we considered the performance of the MSFP model on the KITTI benchmark. As shown in Table 1, the MSFP model achieved outstanding performance on AP_{3D} and AP_{BEV} metrics. Specifically, the car AP_{3D} values are 83.48%, 73.91%, and 67.88% at the easy, moderate, and difficult levels, respectively, and the corresponding car AP_{BEV} values are 89.30%, 86.25%, and 78.58%, respectively. For the pedestrian class, the AP_{3D} values are 41.04%, 37.11%, and 32.02%, respectively, and the AP_{BEV} values are 44.16%, 39.57%, and 38.13% at the easy, moderate, and difficult levels, respectively. These results indicate that the MSFP model can consistently perceive objects accurately. With respect to the running speed, we

Table 1 Performance evaluation results of MSFP model trained and validated both on KITTI dataset.

Class	AP_{3D}(%)			AP_{BEV}(%)			Inference time (ms)
	Easy	Moderate	Hard	Easy	Moderate	Hard	
Car	83.48	73.91	67.88	89.30	86.25	78.58	117
Pedestrian	41.04	37.11	32.02	44.16	39.57	38.13	122

tested the inference time, starting from when the sample data were fed into the model until the detection result was generated, and then took the average time of all the validation samples. It took 117 ms to detect the car and 122 ms to detect the pedestrian in the scene, which demonstrates the efficiency of the perception performance of the proposed model.

Next, we trained and tested the MSFP network using the CarlaSim dataset to evaluate its performance in the simulated environment. As shown in Table 2, the MSFP model still demonstrates excellent detection performance on both classes, even though it was trained using only simulated data, yielding car AP_{3D} values of 81.76%, 72.54%, and 67.25%; car AP_{BEV} values of 88.63%, 85.14%, and 77.69%; pedestrian AP_{3D} values of 40.12%, 35.98%, and 31.20%; and pedestrian AP_{BEV} values of 43.09%, 38.33%, and 36.87% at the easy, moderate, and difficult levels, respectively. The inference time required to detect objects was 119 ms for the car and 121 ms for the pedestrian. The AP values obtained by the MSFP model trained using the CarlaSim dataset were slightly lower than those obtained by the MSFP model trained using the KITTI dataset, but the inference time was similar for the two datasets, thus confirming the accurate and efficient perception performance of the proposed model on the simulated dataset.

Last, to explore the generalizability of the MSFP model from simulated to real-world scenes, we trained the MSFP model on the CarlaSim dataset and used the KITTI sample for validation. For a fair comparison, we randomly selected the same number of samples from the training split of the CarlaSim dataset for model training and then validated the model performance on the KITTI validation split, as presented in Table 3. Quantitatively, the model achieved car AP_{3D} values of 83.32%, 73.81%, and 68.06%; car AP_{BEV} values of 89.29%, 86.13%, and 78.66%; pedestrian AP_{3D} values of 41.25%, 37.07%, and 32.14%; and pedestrian AP_{BEV} values of 44.28%, 40.22%, and 37.99% at the easy, moderate, and difficult levels, respectively. A comparison of the MSFP model performance results which are obtained when trained and validated using the KITTI dataset shows that they are comparable or even slightly more accurate in the car and pedestrian classes, indicating that our MSFP model can generalize well from the unreal to real-world scenes. Compared to the MSFP model trained and evaluated using the CarlaSim dataset, it obtained better perception performance at all three levels of difficulty. To some extent, these results indicate that the simulated CarlaSim data can be substituted for real-world data in autonomous-driving simulation tests, thereby greatly reducing the data-acquisition cost.

For better visualization, Fig. 3 lists several perception results obtained using our MSFP model on the CarlaSim dataset, where we can clearly see that the MSFP model can perceive object localizations and distinguish object classes accurately, even under different weather or lighting conditions.

5 Conclusion

In this study, we investigated the 3D environmental perception ability of a simulated autonomous-driving system. Using the CARLA simulator, we generated unreal traffic scenarios to obtain the CarlaSim dataset, which contains 15 000 camera-LiDAR samples with annotations and calibration matrices. We then developed an MFSP model for object detection in the scene. We performed a wide variety of experiments on the KITTI and CarlaSim datasets; the results demonstrated the effectiveness and efficiency of the CarlaSim dataset and the outstanding perception performance of the MSFP model. The good generalizability of the MSFP model from simulated to real-world data demonstrates that autonomous-driving simulation research can greatly reduce the data-acquisition cost and promote the use

Table 2 Performance evaluation results of MSFP model trained and validated both on CarlaSim dataset.

Class	AP_{3D} (%)			AP_{BEV} (%)			Inference time (ms)
	Easy	Moderate	Hard	Easy	Moderate	Hard	
Car	81.76	72.54	67.25	88.63	85.14	77.69	119
Pedestrian	40.12	35.98	31.20	43.09	38.33	36.87	121

Table 3 Performance evaluation results of MSFP model trained on CarSim and validated on KITTI dataset.

Class	AP_{3D} (%)			AP_{BEV} (%)			Inference time (ms)
	Easy	Moderate	Hard	Easy	Moderate	Hard	
Car	83.32	73.81	68.06	89.29	86.13	78.66	117
Pedestrian	41.25	37.07	32.14	44.28	40.22	37.99	119

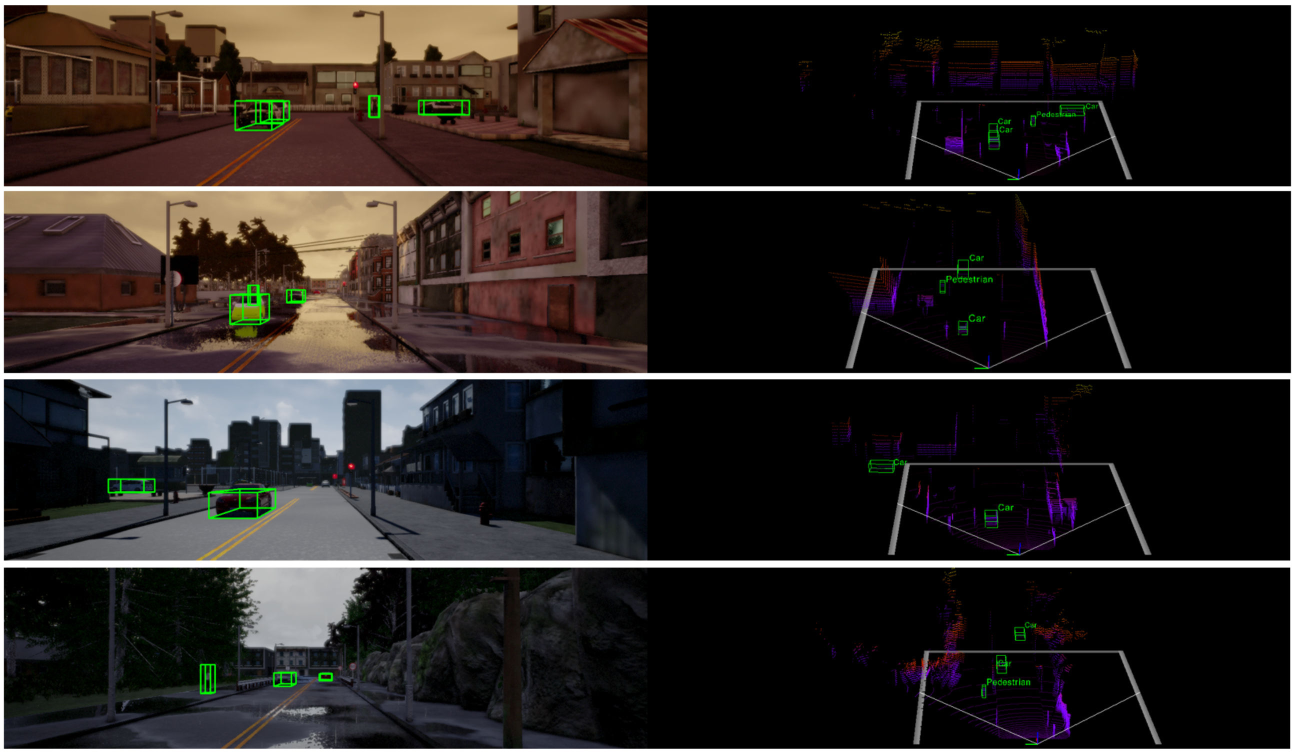

Fig. 3 Visualization results achieved by MSFP model on CarlaSim dataset.

of autonomous-driving system testing.

In the future, three main ideas are worthy of deeper exploration. First, we are interested in obtaining more data from rare scenes, the quality of which must be quantitatively measured to evaluate their effect on model performance. Second, a powerful perception model should be developed, especially one that is capable of multimodal fusion. The development of ways to fully exploit multiple feature maps to obtain complementary information is also highly desirable. Finally, to ensure the integrity of autonomous-driving systems, it is becoming increasingly important to develop methods for harmonizing different modules to enable functionality testing and system validation from simulated to real-world environments, which is a long-term goal of autonomous driving.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61822101 and 62061130221), the Beijing Municipal Key Research and Development Program (No. Z181100004618006), the Beijing Municipal Natural Science Foundation (No. L191001), the Zhuoyue Program of Beihang University (Postdoctoral Fellowship) (No. 262716), and the China Postdoctoral Science Foundation (No. 2020M680299).

References

[1] G. Gome, A. May, and R. Horowitz, Congested freeway microsimulation model using vissim, *Transp. Res. Rec. J. Transp. Res. Board*, vol. 1876, pp. 71–81, 2004.

[2] K. S. Patil, V. Jagtap, S. Jadhav, A. Bhosale, and B. Kedar, Performance evaluation of active suspension for passenger cars using MATLAB, *IOSR Journal of Mechanical and Civil Engineering*, 2013, pp. 6–14.

[3] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, SUMO–simulation of urban mobility: An overview, in *Proc. 3^{rd} Int. Conf. Advances in System Simulation*, Barcelona, Spain, 2011, pp. 55–60.

[4] N. Koenig and A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in *Proc. 2004 IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (*IROS*), Sendai, Japan, 2004, pp. 2149–2154.

[5] I. M. Abuhadrous, F. Nashashibi, C. Laurgeau, and M. Chinchole, Multi-sensor data fusion for land vehicle localization using RTMAPS, in *Proc. IEEE Intelligent Vehicles Symposium*, Columbus, OH, USA, 2003, pp. 339–344.

[6] C. Y. Chen, A. Seff, A. Kornhauser, and J. X. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in *Proc. 2015 Int. Conf. Computer Vision* (*ICCV*), Santiago, Chile, 2015, pp. 2722–2730.

[7] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, Playing for data: Ground TRUTH from computer games, in *Proc. 14^{th}*

European Conf. Computer Vision (*ECCV*), Amsterdam, the Netherlands, 2016, pp. 102–118.

[8] S. R. Richter, Z. Hayder, and V. Koltun, Playing for benchmarks, in *Proc. 2017 IEEE Int. Conf. Computer Vision* (*ICCV*), Venice, Italy, 2017, pp. 2232–2241.

[9] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, TORCS: The open racing car simulator, http://torcs.sourceforge.net/, 2021.

[10] A. Brekke, F. Vatsendvik, and F. Lindseth, Multimodal 3D object detection from simulated pretraining, arXiv preprint arXiv: 1905.07754, 2019.

[11] Y. Lecun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, Vision meets robotics: The KITTI dataset, *Int.J. Robot. Res.*, vol. 32, no. 11, pp: 1231–1237, 2013.

[13] F. Yu, H. F. Chen, X. Wang, W. Q. Xian, Y. Y. Chen, F. C. Liu, V. Madhavan, and T. Darrell, BDD100K: A diverse driving dataset for heterogeneous multitask learning, arXiv preprint arXiv: 1805.04687, 2018.

[14] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, nuScenes: A multimodal dataset for autonomous driving, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Seattle, WA, USA, 2020, pp. 11618–11628.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. Lopez, and V. Koltun, Carla: An open urban driving simulator, in *Proc. 1st Ann. Conf. Robot Learning*, Mountain View, CA, USA, 2017, pp. 1–16.

[16] F. Grazioli, E. Kusmenko, A. Roth, B. Rumpe, and M. Von Wenckstern, Simulation framework for executing component and connector models of self-driving vehicles, in *Proc. 3rd Int. Workshop on Executable Modeling*, Austin, TX, USA, 2017, pp. 105–115.

[17] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, et al., Self-driving car: A survey, expert systems with applications, *Expert Syst. Appl.*, vol. 165, p. 113816, 2020.

[18] X. Z. Chen, K. Kundu, Z. Y. Zhang, H. M. Ma, S. Fidler, and R. Urtasun, Monocular 3D object detection for autonomous driving, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Las Vegas, NV, USA, 2016, pp. 2147–2156.

[19] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, 3D bounding box estimation using deep learning and geometry, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Hawaii, HI, USA, 2017, pp. 7074–7082.

[20] Y. Wang, W. L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, CA, USA, 2019, pp. 8437–8445.

[21] X. S. Weng and K. Kitani, Monocular 3D object detection with pseudo-LiDAR point cloud, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision Workshop*, Seoul, Republic of Korea, 2019, pp. 857–866.

[22] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Hawaii, HI, USA, 2017, pp. 652–660.

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in *Proc. 31st Int. Conf. Neural Information Processing Systems* (*NeurIPS*), Long Beach, CA, USA, 2017, pp. 5099–5108.

[24] Z. Liu, X. Zhao, T. T. Huang, R. L. Hu, Y. Zhou, and X. Bai, Tanet: Robust 3D object detection from point clouds with triple attention, in *Proc. 34th AAAI Conf. Artificial Intelligence, AAAI 2020, the 32nd Innovative Applications of Artificial Intelligence Conf., IAAI 2020, the 10th AAAI Symp. Educational Advances in Artificial Intelligence*, New York, NY, USA, 2020, pp. 11677–11685.

[25] Y. Zhou and O. Tuzel, VoxelNet: End-to-end learning for point cloud based 3d object detection, in *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Salt Lake City, UT, USA, 2018, pp. 4490–4499.

[26] Y. Yan, Y. X. Mao, and B. Li, SECOND: Sparsely embedded convolutional detection, *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[27] A. H. Lang, S. Vora, H. Caesar, L. B. Zhou, J. Yang, and O. Beijbom, PointPillars: Fast encoders for object detection from point clouds, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, CA, USA, 2019, pp. 12698–12705.

[28] X. Z. Chen, H. M. Ma, J. Wan, B. Li, and T. Xia, Multi-view 3D object detection network for autonomous driving, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Hawaii, HI, USA, 2017, pp. 6526–6534.

[29] C. R. Qi, W. Liu, C. X. Wu, H. Su, and L. J. Guibas, Frustum PointNets for 3D object detection from RGB-D data, in *Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Salt Lake City, UT, USA, 2018, pp. 918–927.

[30] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, Joint 3D proposal generation and object detection from view aggregation, in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (*IROS*), Madrid, Spain, 2018, pp. 1–8.

[31] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, Multi-task multi-sensor fusion for 3D object detection, in *Proc. 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), Long Beach, CA, USA, 2019, pp. 7345–7353.

[32] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv: 1409.1556, 2014.

[33] T. Y. Lin, P. Dollár, R. Girshick, K. M. He, B. Hariharan, and S. Belongie, Feature pyramid networks for object detection, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Hawaii, HI, USA, 2017, pp. 936–944.

[34] R. Girshick, Fast R-CNN, in *Proc. 2015 IEEE Int. Conf. Computer Vision*, Santiago, Chile, 2015, pp. 1440–1448.
[35] S. Q. Ren, K. M. He, R. Girshick, and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[36] D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization, in *Proc. of International Conference on Learning Representation (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.

Chunmian Lin received the BE and MS degrees from Fujian Agriculture and Forestry University in 2016 and 2019, respectively. He is currently a PhD candidate at the School of Transportation Science and Engineering, Beihang University, Beijing, China. His current research interests include autonomous driving, computer vision, deep learning, and the applications in intelligent transportation systems.

Daxin Tian received the BEng, MEng, and PhD degrees in computer application technology from Jilin University, Changchun, China, in 2002, 2005, and 2007, respectively. He is currently a professor at the School of Transportation Science and Engineering, Beihang University, Beijing, China. His research is focused on intelligent transportation systems, autonomous connected vehicles, swarm intelligent, and mobile computing. He was awarded Changjiang Scholars Program (Young Scholar) of Ministry of Education of China in 2017, the National Science Fund for Distinguished Young Scholars in 2018, and the Distinguished Young Investigator of China Frontiers of Engineering in 2018. He is also a senior member of IEEE and served as the technical program committee member/chair/co-chair for several international conferences, including EAI 2018, ICTIS 2019, IEEE ICUS 2019, IEEE HMWC 2020, GRAPH-HOC 2020, etc.

Xuting Duan reveived the PhD degree in traffic informaton engineering and control from Beihang University in 2018. Currently, he is an assistant professor at the School of Transportation Science and Engineering, Beihang University, Beijing, China. His current research interests are focused on vehicular ad hoc networks.

Jianshan Zhou received the PhD degree in traffic information engineering and control from Beihang University, Beijing, China in 2020, where he is currently a postdoctoral research fellow. His research interests include vehicular communication network and intelligent transportation systems.