Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 9667506, 11 pages
https://doi.org/10.1155/2022/9667506

Research Article

A Novel Vehicle Detection Framework Based on Parallel Vision

Ying Zhuo,[1] **Lan Yan**,[2] **Wenbo Zheng**,[2] **Yutian Zhang**,[1] **and Chao Gou**[1]

[1]*Shenzhen Campus of Sun Yat-sen University, Sun Yat-sen University, Shenzhen 518107, China*
[2]*Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Chao Gou; gouchao@mail.sysu.edu.cn

Received 15 October 2021; Accepted 17 December 2021; Published 12 January 2022

Academic Editor: Ming Yan

Autonomous driving has become a prevalent research topic in recent years, arousing the attention of many academic universities and commercial companies. As human drivers rely on visual information to discern road conditions and make driving decisions, autonomous driving calls for vision systems such as vehicle detection models. These vision models require a large amount of labeled data while collecting and annotating the real traffic data are time-consuming and costly. Therefore, we present a novel vehicle detection framework based on the parallel vision to tackle the above issue, using the specially designed virtual data to help train the vehicle detection model. We also propose a method to construct large-scale artificial scenes and generate the virtual data for the vision-based autonomous driving schemes. Experimental results verify the effectiveness of our proposed framework, demonstrating that the combination of virtual and real data has better performance for training the vehicle detection model than the only use of real data.

1. Introduction

Recently, with the breakthrough progress of artificial intelligence technology, intelligent vehicles with the advanced driving assistance system (ADAS) are vigorously launched on the market [1, 2]. The ADAS of the intelligent vehicle collects the surrounding data from the sensors like radars and cameras and then performs road object detection and so on. With the detection results and the map data, the control system of the intelligent vehicle proposes autonomous driving schemes. If necessary, the ADAS can even take over the vehicle to avoid accidents.

As the important information sources of intelligent vehicles, vision systems play crucial roles in safe and efficient automatic driving. It can not only obtain the information accurately but also help to conduct the analysis promptly [3]. With the rapid development of intelligent visual perception technology, the deep learning methods for the vision-based intelligent driving models far exceed the traditional methods. These deep learning-based methods require a large amount of labeled data for training. These data should be diverse enough to ensure the model validity under complex conditions. However, the collection and annotation of data are time-consuming and costly, especially under complex driving scenes. What is more, annotating images by hand is often accompanied by subjective judgment, making mistakes in the labeling process. Artificial scenes, which simulate the actual traffic scenes, can be utilized to solve the above problems of real data. They can also help to collect data under various complex situations more easily. What is more, the detailed and accurate annotation information is automatically generated during the collection of virtual data.

With the development of game engines [4, 5] and virtual reality [6–8], the construction of vivid artistic scenes has made great progress. Therefore, parallel vision technology [9–11], which combines real and virtual data to conduct the experiments, obtains better research value and broader application prospects. In the computer vision field, parallel vision [9] is the popularization and application of the ACP (artificial societies, computational experiments, and parallel execution), a theory of complex system modeling and control [12–14]. It firstly simulates and represents complex actual scenes through artificial scenes. Then, computational experiments are used to train, verify, and evaluate various visual models. Finally, the parallel execution of virtual-real interaction is applied to optimize the visual model online,

completing the intelligent perception and understanding of the complex environment. Figure 1 presents the basic framework and architecture of parallel vision [10].

In this work, we propose a novel vehicle detection framework based on the parallel vision theory, using specially designed virtual datasets to help train the vehicle detection model. Concretely, a method to construct artificial scenes and generate virtual data is presented at first. This method can generate a large number of artificial scenes to simulate complex real scenes. These generated scenes meet the requirements of quantity, diversity, scale, and specific occlusion levels. What is more, accurate annotation is automatically annotated during the image generation process. Subsequently, we choose DPM (Deformable Parts Model) [15] and Faster R-CNN [16] as the vehicle detection model. Experiments demonstrate the effectiveness of our framework, which are conducted on the combination of the KITTI [17], PASCAL VOC [18], and MS COCO [19] datasets and the constructed virtual datasets.

Motivated by the above analysis, the main contributions of our paper are threefold. First, a virtual dataset of traffic scenes is generated. By configuring the camera parameters and environmental conditions, the proposed method can flexibly construct artificial scenes with different map data. Second, we apply the parallel vision theory to vehicle detection. Combined with the virtual data, the parallel vision addresses the difficulty of collecting and labeling a large amount of diverse data. Finally, experimental results on the combination of the KITTI, PASCAL VOC, and MS COCO datasets and the constructed virtual dataset verify the effectiveness of the proposed framework.

The remainder of this paper is organized as follows. Some related work is introduced in Section 2. Section 3 presents our method to construct artificial scenes and generate virtual datasets with annotation. Experiments are conducted in Section 4 to validate the utility of our vehicle detection framework based on parallel vision. Section 5 provides the conclusions in the end.

2. Related Work

There are numerous researches about the artificial scene construction and some of them have achieved great results. Qureshi and Terzopoulos [20] first use OpenGL to construct virtual train stations and generate virtual pedestrians based on real railway station scenes. Using 2D virtual image data generated from 3D models in the Google 3D Warehouse, Sun and Saenko [21] train a 2D object detector based on the discriminative decorrelation method. And Hattori et al. [22] use virtual data to train the pedestrian detector under the specific scenario. Chao et al. [23] take the spatiotemporal information of traffic flows as a 2D texture and then formula the generation of new traffic flows as a texture synthesis process. Then, the synthesized vehicle trajectory data is placed to virtual road networks through a cage-based registration scheme so that they can populate virtual road networks with realistic traffic flows. By using the stochastic grammar, Jiang et al. [24] propose a learning-based pipeline to automatically generate and render indoor scenes with the physics-based rendering. With the foreground and background elements of traffic scenes, Yuan et al. [25] construct the traffic scenes by the scene model.

With the Unity3D game engine, Gaidon et al. [26] generate a virtual KITTI dataset based on the real KITTI dataset. The virtual KITTI dataset contains automatically generated annotation information of object detection, semantic segmentation, image depth, and optical flow. It is allowed to set different weather like rainy and foggy days, different periods like morning and dusk, and different camera orientations for the artificial scenes, thus obtaining a more diverse virtual dataset than the real KITTI dataset. Based on the Unity3D game engine, Ros et al. [27] establish a virtual city image dataset SYNTHIA with automatically generated pixel-level semantic annotation, including sidewalks, traffic signs, buildings, trees, street lights, pedestrians, vehicles, and other elements. Recently, generative adversarial networks have also been applied to images synthesis [28–30]. Brock et al. [30] use the trunacation trick to control of the tradeoff between sample variety and fidelity. But these synthetic images lack corresponding annotation information and there are artifacts sometimes. Liao et al. [31] consider that the image generation process should be modeled in 3D space. They generate the abstract 3D representation at first, which is then projected to and refined in the 2D image domain. Therefore, the image synthesis process can be modeled jointly in 3D and 2D spaces.

In this work, we propose a method to construct large-scale artificial scenes flexibly. By changing lighting conditions (from sunrise to sunset) and weather (sunny, cloudy, rainy, foggy, etc.), the proposed method can construct artificial scenes with different map data. And we can obtain the diverse contents of artificial scenes by adjusting the position, height, or orientation of the cameras allows.

3. Artificial Scene Construction and Virtual Dataset Generation

In this section, numerous artificial traffic scenes are constructed at first. We then generate various virtual datasets by tuning different parameters, like the configuration of virtual cameras and the placement of vehicles on the roadsides.

3.1. Artificial Scene Construction. In this section, we introduce our method of artificial scene construction for automatic driving task, as shown in Figure 2. To increase the authenticity of the constructed scenes, we export geographic data from OpenStreetMap, making the geographic attribute of the virtual world correspond to the real world. The geographic data is then imported into the CityEngine software. By editing the CGA (Computer Generated Architecture) rules, we can utilize CityEngine to design a realistic artificial scene, which includes roads, buildings, shops, gardens, trees, and sidewalks. Imported into Unity3D, the artificial scene can be refined by adding the models like vehicles and pedestrians as well as weather systems. And we can adjust the final result of the artificial scene through scripts and coloring programs.

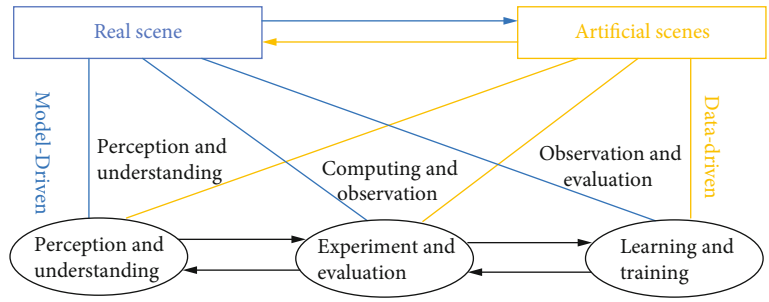

Figure 1: Framework of parallel vision.

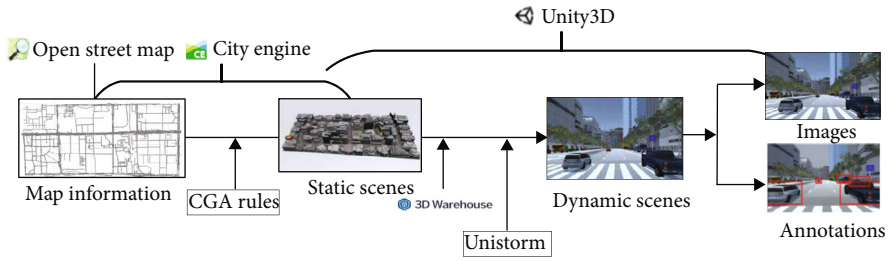

Figure 2: Artificial scene construction pipeline for automatic driving tasks.

3.2. Static Artificial Scene Construction. CityEngine is mainly used to construct a static artificial scene. Firstly, an urban road network is extracted, which is centered on Zhongguancun, Beijing, with a length of 3 kilometers and a width of 2 kilometers. Then, through the OpenStreetMap (OSM), we export the above geographic data as OSM format files, which is subsequently imported into CityEngine. To construct the static city scene in CityEngine, the basic data, texture data, and model component data are needed. Concretely, basic data refers to the vector data needed for modeling and the geographic data needed to construct the scenes. And texture data can help us use rules more easily and make the model more beautiful. And model component data can present complex details and highly reusable models, such as landmarks and tree models. CityEngine is a programmatic modeling application, and we can utilize CGA rules to define how the 3D models are constructed in CityEngine. Since that CGA rules are based on Python, numerous functions are provided to edit CGA rules conveniently. It is critical to extract the common between different objects when we edit CGA rules.

3.3. Dynamic Artificial Scene Construction. Following the construction of the static artificial scene, dynamic models are added dynamic models to construct the dynamic artificial scene in Unity3D, such as vehicles, pedestrians, and the weather system. The static artificial scene in CityEngine is first exported as fbx format models and then imported into Unity3D. Similarly, traffic models like cars, trucks, and buses from Google 3D Warehouse are imported into Unity3D. And we can adjust these traffic models to different sizes and place them in different locations.

Unistorm is applied as the weather system during this process. Unistorm is a powerful Unity3D weather plug-in that can simulate realistic weather such as sunny, partly cloudy, fog, heavy snow, light snow, heavy rain, light rain, and thunderstorm. What is more, Unistorm has an advanced time system and interactive interface, which allows us to adjust the time and weather conveniently.

3.4. Annotation Information Generation. Ground truth annotation information is crucial for the design and evaluation of vision algorithms. Images are traditionally labeled by hand, which is time-consuming and error-prone. Taking semantic segmentation as an example, it usually takes 30 to 60 minutes to label a traffic image containing 10-20 object categories [32]. In addition, manual annotation is subjective. For example, different annotators label different semantic tags for the same image, especially near the object boundary. To solve the above problems, we propose to automatically generate accurate ground truth labels by unity3D. Figure 3 presents some examples of ground truth annotation, including depth, optical flow, object tracking, object detection, instance segmentation, and semantic segmentation.

It is accurate and effective to generate ground truth annotation through Unity3D. The depth ground truth is generated by the built-in depth buffer information, which offers the depth data of the coordinates on the screen. The depth range has a nonlinear distribution from 0 to 1, and 1 means infinitely distant. As for optical flow, the instantaneous speed of a moving object is calculated on the imaging plane at first. Then, according to the changes of pixels in the image sequence, the correspondence between the previous frame and the current frame is found out and the optical flow ground truth is generated. The bounding boxes of object tracking and object detection are generated by MeshFilter and Shaders, the components in Unity3D. The ground truth of semantic segmentation is directly generated by unlit

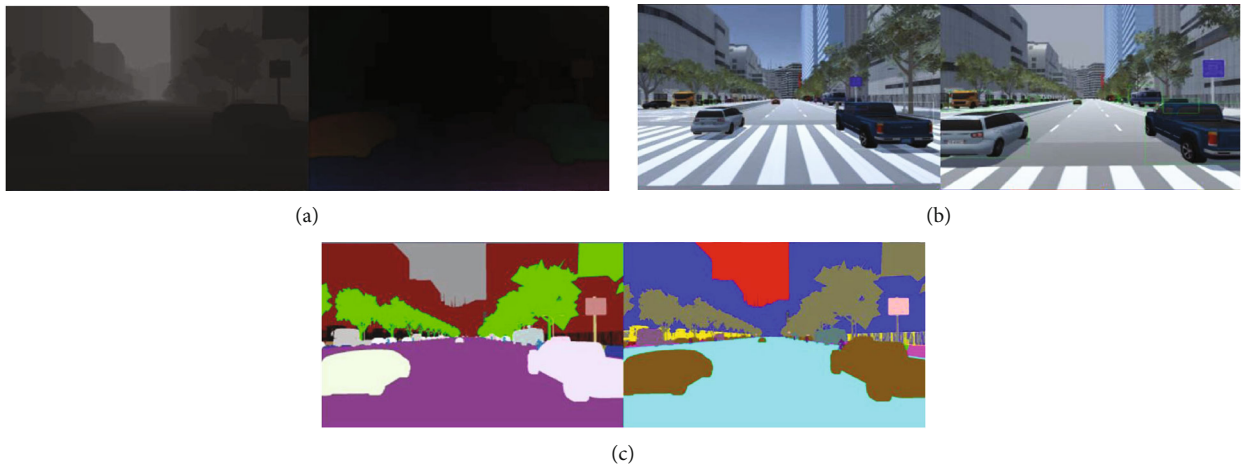

FIGURE 3: Examples of ground truth annotation automatically generated by Unity3D: (a) depth (left) and optical flow (right); (b) object tracking (left) and object detection (right); (c) instance segmentation (left) and semantic segmentation (right).

shades on the material of different objects, and each category corresponds to a unique color. The generation of instance segmentation ground truth is similar. Each instance is assigned a unique color annotation and the color outputted from the modified shader is not affected by light and shadow conditions.

3.5. Virtual Dataset Generation. With the constructed artificial scenes for automatic driving tasks, we generated three different sub-datasets of the ParallelEye [33] dataset, namely, ParallelEye_A, ParallelEye_B, and ParallelEye_C datasets, which contain 5313, 5781, and 4887 images, respectively. The images in ParallelEye_A are the traffic scenes under different weather and shooting angles, used to study the impact of these factors on the vehicle detection models. As for ParallelEye_B, it can help improve the model performance of detecting vehicles at the roadside. There are enough objects with occlusion on each image in ParallelEye_C dataset, which help to study the model performance to detect occluded vehicles. Sample pictures of the three sub-datasets are shown in Figure 4.

For the reason that different weather and shooting angles have a significant impact on the performance of vehicle detection models [34–36], we take weather conditions and shooting angles into consideration at first when generating the virtual datasets. The weather in the artificial scene is adjusted to cloudy at first. And the car camera is set at the height of 2 meters, imitating the camera on the roof. Next, we rotate the camera to five directions, that is, -30°, -15°, 0°, 15°, and 30° relative to the lane deviation. The data is collected once in each direction. These configurations contribute to obvious changes in the appearance of objects. Then, we adjust the camera angle back to 0° and collect data once on rainy and foggy days, respectively. The data collected above consist of the ParallelEye_A dataset. The image resolution of these data is 500×375. There are about 9 to 12 objects on each image, with less occlusion. The size of different objects is small because the road is wide and many vehicles are parked on the side of the road far from the lane where the camera is located.

Since that the virtual scenes in ParallelEye_A dataset simulate the camera on the vehicle driving on the lane, the images in ParallelEye_A dataset present the scenes facing the lane. What is more, the vehicle mentioned above moves on straightly when we generate ParallelEye_A dataset. Therefore, the objects in the images of ParallelEye_A dataset only have a small range of angle changes. To address the above problem, we first adjust the angle of the virtual camera to 90°, so that it faces the side of the road. Then, the camera is set to move up and down when the vehicle is moving. At the same time, we use the script to control vehicles at the roadside to park at numerous angles. Thus, ParallelEye_B dataset is generated. The image resolution of this dataset is 500×375. With less occlusion, there are only 1 or 2 objects on each image, which have large sizes and numerous angles.

Obviously, there are few images with objects occluded in ParallelEye_A and ParallelEye_B dataset and the image resolution is low. Therefore, we further generate ParallelEye_C dataset. In order to increase the number of objects and make more occlusion, we place a lot of vehicles at the adjacent lanes and the roadside. The image resolution of ParallelEye_C dataset has been refined to 1000×3000. What is more, the color of objects varies randomly during the generation of ParallelEye_C dataset. As shown in Figure 5, the color of the same vehicle is different between the adjacent two frame images. Therefore, there are enough objects with occlusion on each image in ParallelEye_C dataset. These objects not only have large sizes but also are diverse enough to ensure the model validity under complex conditions.

4. Experimental Results

In this section, we conduct the experiments to validate the effectiveness of our vehicle detection framework. We first introduce the real dataset used in the experiment. And we choose DPM and Faster R-CNN as the vehicle detection model in this work. Then, experiments are conducted on the combination of real and virtual datasets.

Finally, we analyze and summarize the results of the experiment.

(a) (b)

Figure 4: Example images of three sub-datasets: (a) ParallelEye_A (left) and ParallelEye_B (right); (b) ParallelEye_C.

Figure 5: Two adjacent frame examples in the ParallelEye_C dataset.

4.1. Model Description. Deformable part model [15] is a classic object detection model, called DPM for short. DPM models the object as a combination of several components. A DPM model includes a root filter and several part filters. The root filter focuses on the entire object, and the part filters focus on different components. There are deformation parameters to indicate the deformation cost of different components so that the relative position of each component and the main part is relatively stable. For a certain position, DPM uses a score function to determine whether there is an object.

As for object detection models based on deep learning, they can be divided into two categories: one-stage methods and two-stage methods. The former directly produces the category probability and coordinate position of the object. The latter divides the detection problem into two stages, first generating candidate regions and then classifying the candidate regions. Faster R-CNN [16] model is the most classic object detection model in two-stage methods. After obtaining the feature map of the input image, the Region Proposal Network is used to generate region proposals. This layer uses softmax to determine whether the anchor is foreground or background and then uses border regression to correct the anchor to obtain an accurate proposal. RoI pooling integrates feature maps and proposals to extract proposal feature maps. The proposal feature map is classified into specific categories by the next classifier through softmax. At the same time, the classifier uses L1 Loss to complete bounding box regression and obtain the precise position of the object.

4.2. Data Preparation. The proposed framework is based on the parallel vision theory, and therefore, we should prepare real datasets apart from the virtual dataset constructed above. We use VOC_COCO and KITTI datasets as real datasets in this work.

And the VOC_COCO dataset consists of PASCAL VOC and MS COCO dataset.

PASCAL VOC dataset is provided by PASCAL VOC Challenge, which is a benchmark test of visual object classification, recognition, and detection [18]. Apart from the dataset with standard annotation, PASCAL VOC Challenge also provides the standard evaluation system of object detection algorithms and learning performance. Nowadays, both PASCAL VOC Challenge and PASCAL VOC dataset have become a widely accepted standard in the field of object detection. The proposed framework is applied to vehicle detection under complex driving conditions. Therefore, during the experiment, we focus on the vehicle objects in the traffic scene, that is, the cars, buses, and trucks. For the reason that there are no trunk objects in PASCAL VOC dataset, we add MS COCO dataset [19] including truck objects. To construct the real datasets, we select 1997 images of car and bus objects from PASCAL VOC dataset and 1000 images of truck objects from MS

COCO dataset. After the extraction and fusion of the above data, we normalize the image annotation format of MS COCO dataset to the VOC format, as the annotation format of PASCAL VOC dataset. This real dataset is called VOC_COCO dataset.

Using PASCAL VOC and MS COCO datasets, we can only validate the vehicle detection performance in general scenarios of the proposed framework. To demonstrate the performance of the framework under the autonomous driving scene, KITTI [17] dataset is used to conduct another experiment. KITTI is currently the largest dataset for computer vision algorithm evaluation in automatic driving scenarios throughout the world. This dataset is often used to evaluate the performance of computer vision algorithms such as vehicle detection, in the complex real environment.

4.3. Experiments on VOC_COCO Dataset. In this section, the models are trained with the real data and the combination of real and virtual dataset, respectively, and then tested the models on the VOC_COCO dataset. The VOC_COCO dataset is divided into training set and testing set with the proportion 3 : 1. We first train DPM and Faster R-CNN models on the training set of VOC_COCO. Subsequently, images are selected from the virtual datasets of ParallelEye_A, ParallelEye_B, and ParallelEye_C. We combine this virtual data with the real data from the training set of VOC_COCO with the proportion 3 : 2. Then, the combination of virtual and real datasets is used to train DPM and Faster R-CNN models.

All the models are evaluated on the testing set of VOC_COCO.

When training DPM, we train the model for car, bus, and truck objects separately. The number of components is set as 3 and the maximum number of components is 8. Ensure that the ratio of positive to negative samples is 1 : 3 in the training set and testing set. DPM trained for the car objects on VOC_COCO is shown in Figure 6.

As for the Faster R-CNN model, there is no need to train three models for car, bus, and truck objects separately.

We use the classic VGG16 network structure and set the learning rate to 0.001 with the intersection over union (IoU) set to 0.7. The model is pretrained on ImageNet for initialization. The experimental results are shown in Table 1.

It can be seen from Table 1 that both DPM and Faster R-CNN model trained by the combination of real and virtual data obtain higher average precision (AP) than the models trained by the real data. Especially for the Faster R-CNN model of bus objects, the model based on the combination of real and virtual data increases average precision by 5.1%, compared to the model trained by the real data. The average precision is only improved by 0.6% for the DPM of bus objects. It is because that the AP of the DPM model trained by the real dataset is as high as 71.5%, and the performance is hard to be improved. This experiment verifies the effectiveness of our framework.

In addition, it is noticed that DPM achieves higher AP than the Faster R-CNN model for vehicle detection of bus objects. It may be because that there are fewer bus images in VOC_COCO and these bus objects are easy to be detected. Since DPM is based on HOG features, it focuses on the features related to object detection difficulty, i.e., the clarity and occlusion rate of the edge, while Faster R-CNN is a deep learning algorithm, requiring a large amount of data to train the model.

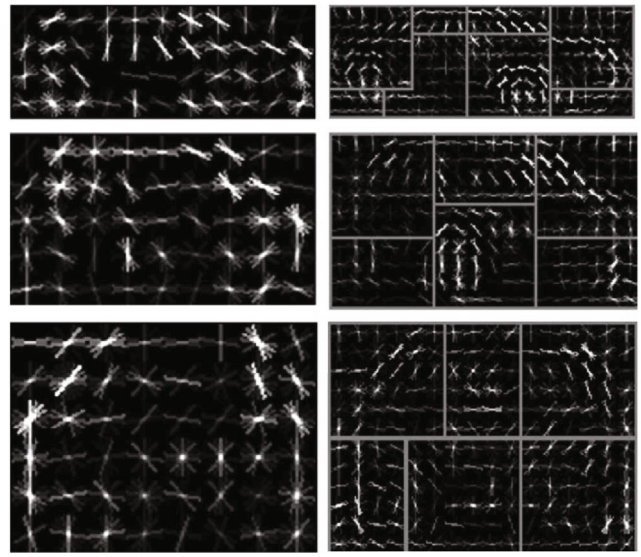

Figure 6: The visual DPM model diagram of car objects trained on VOC_COCO dataset.

The car detection result of the models trained by the VOC_COCO and ParallelEye dataset is shown in Figure 7. It is obvious that DPM performs poorly on occluded objects. While the Faster R-CNN model still has a relatively good performance under such conditions. DPM performs object detection according to the final detection score, and the final score is obtained from the root filter and component filter scores. For partially occluded objects, on the one hand, the features of objects may be affected by the features of the occlusion when the model calculates the object feature. This also affects the score of the filter at the corresponding position, thereby affecting the position with the highest calculated score and resulting in a deviation in the detection position. On the other hand, the occlusion may lower the score of the correct component position, and then, the final score is too low to detect the correct object. The correct object is missed, thereby reducing the detection accuracy. What is more, the object features of DPM are mainly in the edge area of the object because the model is based on HOG features. When there are occlusions in the edge area of the object, DPM has the worst performance of object detection. To sum up, the DPM algorithm cannot deal with the occlusion problem properly. As for the objects without occlusion, there is no significant difference in performance between the two algorithms.

4.4. Experiments on KITTI Dataset. The dataset used in the above experiments is the general dataset, rather than a

Table 1: Experimental results on the VOC_COCO and ParallelEye datasets.

Algorithm	Training dataset	Average precision		
		Car	Bus	Truck
DPM	VOC_COCO	0.482	0.715	0.482
	VOC_COCO and ParallelEye	0.496	0.721	0.499
Faster R-CNN	VOC_COCO	0.542	0.641	0.563
	VOC_COCO and ParallelEye	0.558	0.692	0.594

(a) (b)

Figure 7: Example of the car class detection results of models trained on VOC_COCO and ParallelEye dataset: (a) DPM and (b) Faster R-CNN.

specific dataset for autonomous driving. To validate the effectiveness of our framework in autonomous driving, we use KITTI and ParallelEye datasets to conduct the experiments in this section. Because there are no bus objects and few trunk objects in KITTI dataset, we only use the car data in KITTI and ParallelEye dataset for the experiments. 6684 pictures containing cars are selected from KITTI and we divide them randomly into the training set and testing set with the proportion 1 : 1.

DPM is trained for car objects with the same setting. We set the number of components as 3 and the maximum number of components as 8. The ratio of positive to negative samples is kept as 1 : 3 in both training and testing sets. We first train DPM on the KITTI training set and test the model on the testing set of KITTI. Next, some data is randomly chosen from ParallelEye_A, ParallelEye_B, and ParallelEye_C datasets. These virtual data are then combined with the real data from the training set of KITTI with the proportion 3 : 2. We use these data to train DPM and evaluate the model on the KITTI testing set.

Table 2: Evaluation results of car object detectors based on KITTI and ParallelEye datasets.

Algorithm	Training dataset	Average precision
DPM	KITTI	0.516
	KITTI and ParallelEye	0.527
Faster R-CNN	KITTI	0.741
	KITTI and ParallelEye	0.757

When training Faster R-CNN model, we use the model pretrained on ImageNet for initialization. The learning rate is set as 0.0008 and IOU is 0.7. After the algorithm converges, the model is tested on KITTI testing set. Subsequently, we use 4000 images of cars in the above randomly selected ParallelEye dataset to train Faster RCNN model with the learning rate of 0.001. Then, KITTI dataset is utilized to fine-tune the trained model and we set the learning rate as 0.0002. Finally, the model was evaluated on the

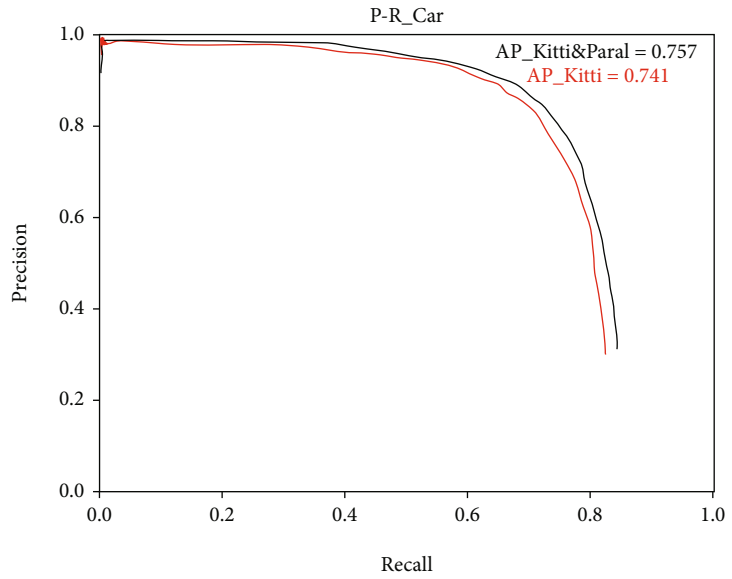

Figure 8: Precision-recall curve of Faster R-CNN on the test set of KITTI dataset. The red line represents model training using only the KITTI dataset, while the black line indicates training using the virtual dataset first and then fine-tuning using the KITTI dataset.

(a) (b)

Figure 9: Example of detection results of a virtual-real training model on a KITTI dataset. (b) Faster R-CNN.

KITTI dataset. The experimental results are shown in Table 2, and the obtained precision-recall (P-R) curve is presented in Figure 8.

According to Table 2, using the combination of virtual and real datasets to train the algorithm achieves better performance than using only the real dataset for visual model training. These experiments all verify the effectiveness of our vehicle detection framework based on the parallel vision theory in automatic driving scenarios. As mentioned before, artificial scenes can help quickly generate the labeled data. It makes up for the shortcomings of real datasets and thereby improves the performance of the model. Figure 9 presents detection results on KITTI testing set of the model trained by the virtual and real dataset.

In addition to the above experiments, we also conducted a comparative experiment on KITTI testing set. We first train detection models for car objects on ParallelEye and VOC_COCO training set separately. Then, both models are evaluated on KITTI testing set. The experimental results are listed in Table 3, and the P-R curve of Faster R-CNN is shown in Figure 10. According to Table 3 and Figure 10, the models trained by ParallelEye achieve better performance under real traffic conditions than the models trained by VOC_COCO. These results further demonstrate the effectiveness of our framework.

Table 3: Evaluation results of car object detectors on KITTI testing set.

Training dataset	Average precision	
	DPM	Faster R-CNN
ParallelEye	0.478	0.574
VOC_COCO	0.419	0.523

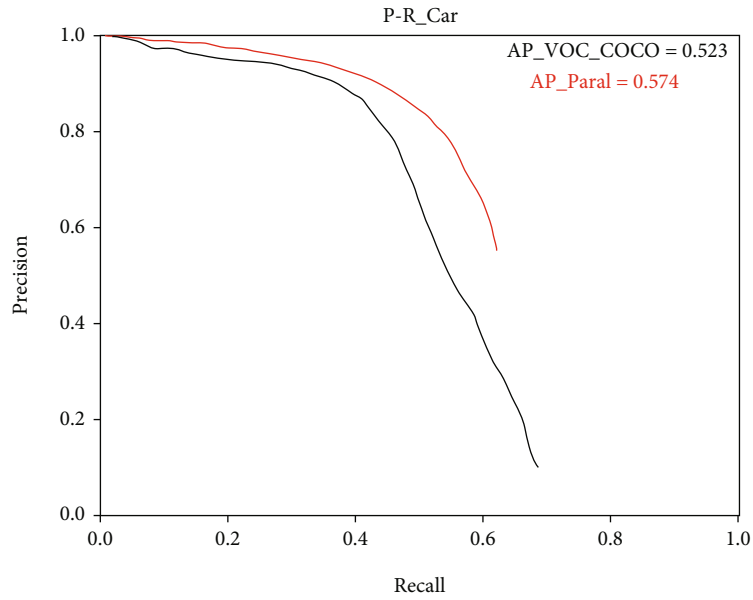

Figure 10: Precision-recall curve of Faster R-CNN model trained on ParallelEye or VOC_COCO dataset.

5. Conclusion

In this paper, we propose a novel vehicle detection framework based on parallel vision, using the specially designed virtual dataset to help train the vehicle detection model. Specially, a method is proposed to construct artificial scenes and generate virtual datasets. This method can not only simulate complex real scenes with diverse environmental changes but also generate ground-truth labels automatically. Extensive experiments have demonstrated that the combination of virtual and real datasets has better performance for training the vehicle detection model than the only use of real datasets, verifying the effectiveness of our proposed framework. Finally, it should be mentioned that we combine real and virtual data with the given proportion for model training, in order to verify the effectiveness of the proposed framework based on parallel vision. Different proportions of virtual and real data in training will be the direction of further development of the proposed framework.

Data Availability

The data used to support the findings of this study were collected from the Pascal VOC, MS COCO, and KITTI public databases as well as ParallelEye dataset. The ParallelEye dataset is also available now.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Authors' Contributions

Chao Gou proposed that we could solve the problem of traffic data shortage based on the parallel vision theory. As supervisor and funding recipient, he oversees and leads the planning and execution of research activities. Ying Zhuo and Lan Yan are the main contributors to this study. They designed the construction pipeline of artificial scenes and generated three different sub-datasets of the ParallelEye dataset. With the help of other authors, they completed vehicle detection experiments based on real and virtual data and analyzed the results. In addition, the initial manuscript is mainly finished by Ying Zhuo, who is also responsible for the visualization and revision based on the review of the manuscript. Wenbo Zheng puts forward some suggestions for vehicle detection and further improvement of virtual data generation. Yutian Zhang has done an in-depth investigation for the research and helped adjust the parameters involved in the virtual data generation. Ying Zhuo and Lan Yan contribute equally to this study.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (2020YFB1600400), in part by the Key Research and Development Program of Guangzhou (202007050002), in part by the National Natural Science Foundation of China (61806198), and in part by the Shenzhen Science and Technology Program (Grant No. RCBS20200714114920272).

References

[1] M. Yan, S. Li, C. A. Chan, Y. Shen, and Y. Yu, "Mobility prediction using a weighted Markov model based on mobile user classification," *Sensors*, vol. 21, no. 5, p. 1740, 2021.

[2] M. Yan, Z. Li, X. Yu, and C. Jin, "An end-to-end deep learning network for 3D object detection from RGB-D data based on Hough voting," *IEEE Access*, vol. 8, pp. 138810–138822, 2020.

[3] A. S. Matveev, H. Teimoori, and A. V. Savkin, "A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance," *Automatica*, vol. 47, no. 3, pp. 515–524, 2011.

[4] W. S. Bainbridge, "The scientific research potential of virtual worlds," *Science*, vol. 317, no. 5837, pp. 472–476, 2007.

[5] Q. Miao, F. Zhu, Y. Lv, C. Cheng, C. Chen, and X. Qiu, "A game-engine-based platform for modeling and computing artificial transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 343–353, 2011.

[6] J. Sewall, J. van den Berg, M. Lin, and D. Manocha, "Virtualized traffic: reconstructing traffic flows from discrete spatio-temporal data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 1, pp. 26–37, 2011.

[7] H. Prendinger, K. Gajananan, A. B. Zaki et al., "Tokyo virtual living lab: designing smart cities based on the 3D Internet," *IEEE Internet Computing*, vol. 17, no. 6, pp. 30–38, 2013.

[8] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 394–406, 2012.

[9] K. Wang, C. Gou, and F. Wang, "Parallel vision: an ACP-based approach to intelligent vision computing," *Acta Automatica Sinica*, vol. 42, no. 10, pp. 1490–1500, 2016.

[10] K. Wang, C. Gou, N. Zheng, J. M. Rehg, and F.-Y. Wang, "Parallel vision for perception and understanding of complex scenes: methods, framework, and perspectives," *Artificial Intelligence Review*, vol. 48, no. 3, pp. 299–329, 2017.

[11] K. Wang, Y. Lu, Y. Wang, Z. Xiong, and F. Wang, "Parallel imaging: a new theoretical framework for image generation," *Pattern Recognition and Artificial Intelligence*, vol. 30, no. 7, pp. 577–587, 2017.

[12] W. Fei-Yue, "Parallel system methods for management and control of complex systems," *Control and Decision*, vol. 19, no. 5, pp. 485–489, 2004.

[13] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: concepts, architectures, and applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630–638, 2010.

[14] F. Wang, "Parallel control: a method for data-driven and computational control," *Acta Automatica Sinica*, vol. 39, no. 4, pp. 293–302, 2013.

[15] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we Ready for Autonomous Driving? The Kitti Vision Benchmark Suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, IEEE, Providence, RI, USA, 2012.

[18] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[19] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft coco: common objects in context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[20] F. Qureshi and D. Terzopoulos, "Smart camera networks in virtual reality," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1640–1656, 2008.

[21] B. Sun and K. Saenko, "From virtual to reality: fast adaptation of virtual object detectors to real domains," *BMVC*, vol. 1, p. 3, 2014.

[22] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade, "Learning scene-specific pedestrian detectors without real data," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3819–3827, Boston, MA, USA, 2015.

[23] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 2, pp. 1167–1178, 2018.

[24] C. Jiang, S. Qi, Y. Zhu et al., "Configurable 3D scene synthesis and 2D image rendering with per-pixel ground truth using stochastic grammars," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 920–941, 2018.

[25] S. Yuan, Y. Chen, H. Huo, and L. Zhu, "Analysis and synthesis of traffic scenes from road image sequences," *Sensors*, vol. 20, no. 23, p. 6939, 2020.

[26] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, Las Vegas, NV, USA, 2016.

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, Las Vegas, NV, USA, 2016.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, pp. 2672–2680, 2014.

[29] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *2018 IEEE International Conference on Big Data (Big Data),*, Seattle, WA, USA, 2018.

[30] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*, pp. 7354–7363, PMLR, 2019.

[31] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger, "Towards unsupervised learning of generative models for 3D controllable image synthesis," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5871–5880, Seattle, WA, USA, 2020.

[32] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, "Joint semantic segmentation and 3D reconstruction from monocular video,," in *Computer Vision – ECCV 2014*, pp. 703–718, Springer, 2014.

[33] X. Li, K. Wang, Y. Tian, L. Yan, F. Deng, and F.-Y. Wang, "The ParallelEye dataset: a large collection of virtual images for traffic vision research," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2072–2084, 2018.

[34] K. Wang, W. Huang, B. Tian, and D. Wen, "Measuring driving behaviors from live video," *IEEE Intelligent Systems*, vol. 27, no. 5, pp. 75–80, 2012.

[35] K. Wang, Y. Liu, C. Gou, and F.-Y. Wang, "A multi-view learning approach to foreground detection for traffic surveillance applications," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4144–4158, 2015.

[36] H. Zhang, K. F. Wang, and F. Y. Wang, "Advances and perspectives on applications of deep learning in visual object detection," *Acta Automatica Sinica*, vol. 43, no. 8, pp. 1289–1305, 2017.