A Large-scale Simulation Dataset: Boost the Detection Accuracy for Special Weather Conditions

Dongfang Liu[1]*, Yiming Cui[2]*, Zhiwen Cao[1], and Yingjie Chen[1]
[1]Department of Computer Graphics Technology, Purdue University, West Lafayette, 47907, USA
Email: {liu2538, cao270, victorchen}@purdue.edu
[2]Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL, 32611, USA
Email: cuiyiming@ufl.edu
* authors contribute equally on this work.

***Abstract*—Object detection is a fundamental task for autonomous driving systems. One bottleneck hindering detection accuracy is a shortage of well-annotated image data. Virtual reality has provided a feasible low-cost way to facilitate computer vision related developments. In autonomous driving area, existing public datasets from real world generally have data biases and cannot represent a wide range of weather conditions, such as rainy or snowy roads. To address this challenge, we introduce a new large-scale simulation dataset which is generated by an automated pipeline from a high realism video game. Our dataset focuses on weather conditions, which can be adopted to train networks to effectively detect objects under such conditions. We use extensive experiments to evaluate our dataset by comparing it with public datasets. The experiment results show that networks trained with our dataset outperform the networks trained by other public datasets. Our work demonstrates the effectiveness of using simulation data to address real-world challenges in the practice of object detection.**

***Index Terms*—Large-scale dataset for autonomous driving, simulation data, automated data generation, special weather autonomous driving, object detection accuracy**

I. Introduction

Computer vision-based object detection is considered one of the fundamental challenges in autonomous driving, which deals with the problem of locating cars and pedestrians in the region of interest [1] [2]. The progress of deep learning networks has drastically improved the detection accuracy for autonomous driving system in recent years. However, for any object detection network to gain a high detection accuracy, well-annotated datasets are essential—especially within current network structures [3] [4].

Without changing the architecture of a network, researchers can leverage improved datasets for better detection results. The recent progress in dataset research has brought us larger and better annotated public datasets. In the domain of autonomous driving, KITTI [3] is the widely recognized dataset. However, KITTI has notable drawbacks: 1) it contains relatively small amounts of data; 2) the data were collected in specific regions and nations which lack diversity with respect to car types and road conditions; and 3) the datasets lack representation of different weather for driving, such as snowy or rainy days. Due to the datasets' limitations, networks trained by this type of dataset may encounter difficulties in generalization [5].

However, collecting data on such weather conditions in order to train an object detection network can be arduous and dangerous. Also, such data collection will take a much longer time since we would have to wait for each different level of rain or snow intensity. To our best knowledge, no currently available public datasets specifically focus on this challenge. Berkeley Deep Driving (BDD) includes special weather conditions such as rain and snow but is not sufficient as the size of the weather data is limited [6]. Virtual reality has provided a feasible low cost way to facilitate computer vision related developments [7]. With the help of virtual reality and computer graphics, simulation data can be applied to augment existing dataset. However, most of the existing works [8], which uses games like Grant Theft Auto V (GTA V) to augment dataset, only take limited weather conditions into account. In contrast, we care more about autonomous driving in different special weather conditions since it is challenging and all-consuming to get real data. Inspired by [9] [10] who used simulations to address the problem of data hunger, we present a large dataset for autonomous driving research that simulates special weather conditions in this paper.

The major contributions of our work are as follows: 1. We create a new large-scale simulation dataset for autonomous driving which focuses on special weather conditions. The results from our work indicate that the object detection networks trained by our simulation dataset outperform their counterparts in terms of detection accuracy under both rainy and snowy weather conditions; 2. We implement an automated pipeline for data collection and annotation from a high realism video game. Our data approach reduces the cost of data collection and annotation; 3. We help similar research in our field better understand the simulation data generation and investigate its contribution to object detection for autonomous driving.

II. Relevant Research

Object detection has developed rapidly from an obscure target to a key emphasis in the most up-to-date studies on computer vision [11] [12]. Notwithstanding significant progress, object detection is still subpar, especially when compared to human performance. The development of deep-learning approaches has profoundly influenced most advanced, contemporary techniques in the field of computer vision [13]. Currently, many fruitful methods in various practices have been set up using deep neural networks [14] [15] [16] [17]. One of the primary reasons for their achievements is due to their easy access to extensive datasets like ImageNet [18], PASCAL VOC [19], PASCAL-Context [20], and Microsoft COCO [21] that encourage deep neutral networks to play their due roles.

For autonomous driving, the autonomous-driving agent has special and high requirements for system performance and dependability. As a result, much attention has been given to studies on the latest object detection technologies for architecture design for deep-learning networks [22]. On the other hand, research progress towards image recognition for autonomous driving systems has direct association with the existence of datasets such as the KITTI [23], CamVid [24], Leuven [25], Cityscapes [26] and Daimler Urban Segmentation [27] datasets. However, the above-mentioned scene datasets collected in cities are usually much too small to handle all possible driving circumstances in more general surroundings. Up to now, this deficiency has restricted the further development of image recognition of road conditions. To achieve the goal of human-level recognition, deep learning networks should be trained with datasets that include good annotations, a large capacity, and extensive contents. In future efforts, attention should be given to working out an automated annotator to replace human labors in data labeling and classification [10]. It will get researchers nowhere to set up high-quality and large-capacity datasets without the means to implement automated annotation of that data [10]. Despite part of current achievements probing into the possibilities of automated annotation, dealing with such large datasets still requires the engagement of manpower for managing specific labeling procedures and qualifying annotation outcomes. To address this challenge, our work endeavors to reach a fully automated annotation process and accurately label image data.

Based on the discussion above, it is insufficient for the current, relatively limited (only a few thousand of images in contemporary datasets), manually-labeled datasets to enable an autonomous car to operate safely in the real world. The collection of real-world data requires significant amounts of time, money and manpower [10]. Current prestigious datasets with emphasis on real-world data have required the support of substantial monetary appropriations [26] [23] [28]. As a reply to this issue, simulation data is put forward as an alternative to limited, manually-labeled datasets. In the past three decades, computer graphics have become more sophisticated and there has been an increasing market for virtual reality entertainment, stimulating great innovation in the production of high realism imagery at high speeds. If it is possible for networks trained with this kind of simulation data to work as well as those trained with real-world data to handle practical missions, then we can start to increase the amount of training data provided from simulations and validate whether larger datasets are adequate for understanding generic models.

Up to now, several attempts have been made to produce simulation data to better the performance of deep-learning networks. [29] applied 3D renderings of objects to enrich existing training data for the assignment of pedestrian recognition. [30] introduced synthetic modeling of pedestrian movement from different detectors. [29] applied 3D rendering to generate a synthetic image, which is conducive to the setting of training data for a convolutional neural network (CNN) based method in viewpoint estimation. [10] employed high-realism video games to produce training data for vehicle detection. [10] collected and annotated data from video games using a semi-automated labeling program. [10] demonstrated that the upper limit of training examples that can be applied without changing the architecture of a deep-learning network has not yet been reached [10].

Our work is inspired by [10]. We also collect simulation data for training object detection network. Unlike [10] whose data generation process is semi-automated, we implement a fully automated pipeline for data collection and annotation to create a new large-scale simulation dataset. We use our simulation dataset to train a network that achieved state-of-the-art performance to a network trained on real-world data.

III. Method and Experiments

In this section, we first present the data collection process which relies on a self-driving agent in the video game. Once we collect the raw data, we implement an automated annotation tool to label our data. At the end, we evaluate the collected dataset using several modern object detection networks.

A. Autonomous Data Collection

1) Self-Driving Agent: To collect data, we implement a self-driving agent based on [5] [31] in the computer game Grant Theft Auto V (GTA V). GTA V creates highly photo-realistic image data to simulate city and scenery roads under a variety of terrain and weather conditions.

Although it is possible to simply have a human play the game and capture scenes, acquiring a large enough dataset in this way will cost too much in terms of human resources. Therefore, we first hire several players to play the game and capture their driving behavior, and then develop a self-driving agent based on the collected human driving data. The driving commands for the self-driving agent are generated by a CNN model which takes the driving images as inputs and predicts a proper steering decision (as shown in Fig. 1).

To avoid potential collisions, we utilize a customized object detection [32] and depth estimation network [33] to detect vehicles and pedestrians. If vehicles or pedestrians are detected by the object detection network, we calculate the relative distance (RD) of the detected object from the self-driving agent by using the depth estimation network. RD is in percentile scale with a small number being equal to a small distance. We designate $RD>0.5$ to be a safe distance for driving. If $RD\leq 0.5$, we continue to check the localization of the object's centroid point (CP). If the CP falls in the range of 30 to 70 percent of the image's width, it means that the detected object is on our agent's trajectory and a brake is continuously generated until the object moves away. In contrast, no brake is generated

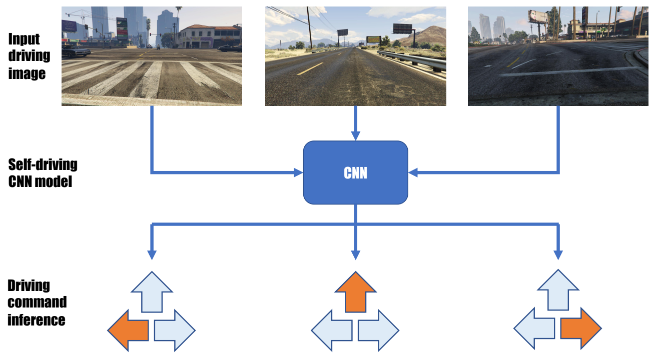


Fig. 1: The CNN model for self-driving agent. It takes the driving image as input and predicts steering commands. For instance, from the left to the right column, the CNN model predicts to turn left, move forward, and turn right.

for the self-driving agent. The brake command generation is demonstrated in Fig. 2.

2) Raw Data Generation: Once the self-driving agent is trained, we use the trained agent to replace manpower to collect raw data. When the self-driving agent drives in the video game, we implement a screen-shot program to prepare to grab the video frame every two seconds. To collect data accurately, we leverage our customized object detection module to detect if an object of interest appears in the frame. If objects of interest (vehicular objects) are detected at the 2-second mark, this frame is grabbed and saved for further stages of the process; on the contrary, no data is collected at that moment. In our work, we define special weather conditions to be either rainy or snowy weather, so we set the game's weather status to one of those options when collecting data. In order to include diverse lighting conditions, the time flow rate is set as one hour per second in the game. The entire data collection process is summarized in Algorithm 1.

B. Automated Data Processing and Labeling

When collecting data, we set the game window size as 1600×900. Once we collect the raw data, we first pre-process the raw image by removing both the headlight of the self-driving agent and the sky. We only keep the region of interest which focuses on the road image, as shown in Fig. 3. The final image size is 1600×600.

After pre-processing, we implement an automated annotator to produce the accurate ground truth of each object in the image, as shown in Fig. 4. In order to obtain high quality annotations, we implement an annotator that follows a strict protocol to label the raw data. The annotation steps are elaborated below:

- Based on the saved ground truth of each object from data collection, our annotator computes the change of the image's size from pre-processing and generates a new rough ground truth for the individual object (as shown in Fig. 3(a)). However, the rough ground truth is not accurate enough and includes pixels that do not belong to the target object.
- Next, the annotator computes the mean depth of the object of interest (as shown in Fig. 3(b)). With this depth estimation information, we are able to identify objects with severe truncation or occlusion.
- Based on the mean depth information and rough ground truth, our annotator iteratively extracts and cuts out the individual object of interest. Each object is then masked with a unique color and its contours are drawn to help discriminate occlusion or truncation (as shown in Fig. 3(c)).
- Once we have the contours of each object, it is easy for the annotator to separate each detected object from the image and produce a more accurate ground truth. We visualize the bounding boxes of each object which illuminate that the ground truth for each is improved effectively (as shown in Fig. 3(d)).

After each labeling process, the annotator automatically saves the ground truth of the individual objects from the image data into a corresponding XML file. There are 300,000 images labeled. Once the annotation process is completed, the entire dataset is randomly split into training, testing, and validation datasets in the proportions of 75%, 15%, and 10% respectively.

C. Dataset Evaluation

We employ four state-of-the-art object detection networks, R-FCN [34], Faster-RCNN [35], SSD [36], and YOLOv3 [37], to evaluate the performance of our dataset. We employ the Keras framework for all network training. We train each of the networks with our dataset, with KITTI, and with BDD (only snowy and rainy data).

Before training, we modify the annotations of the KITTI and BDD datasets to align with Pascal VOC format [3]. In our work, the detected objects are all vehicles, so the annotations of the two datasets are modified accordingly. In KITTI, the original 'car', 'truck', and 'van' classes are changed to 'car'. In BDD, the original 'car', 'bus', 'truck', and 'motor' classes are changed to 'car'. If the annotation is not a vehicle, that object's class is removed from the two datasets used in our training.

During the training, we employ an Adam optimizer with 10^{-4} as the learning rate, 10^{-5} as the decay rate, and 12 as the batch size. For each epoch, the datasets are learned entirely. Since the image sizes and data size of each dataset are different, we do not use a fixed number for the training epoch. When training for each network reaches a plateau status, namely the loss rate and accuracy stop changing, we end the training.

To our best knowledge, no dataset is currently available which exclusively focuses on special weather and road conditions. To test the performance of each trained network in real-world environments, we leverage Google Search API to download 4,000 pictures which focus on traffic of rainy and snowy day. We use the self-prepared testing dataset to evaluate

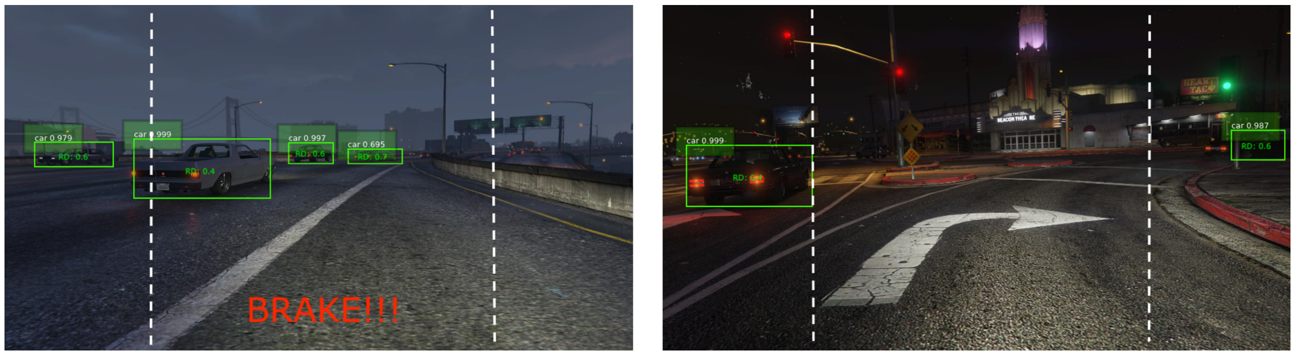


Fig. 2: Brake command generation. The area within the white dotted lines illuminates the trajectory of the self-driving agent. The left image includes a vehicular object which is within the driving trajectory of the self-driving agent and at an unsafe distance, so our model issues a brake command to stop our self-driving agent; the right image shows a vehicular object which is not at a safe distance but not within the driving trajectory of our self-driving agent, so our model issues no brake at this moment.

Algorithm 1 Autonomous Data Collection Process.

```
 1: input: frame$\{I_t\}$                      ◁ Input frames $I_t$ from GTA V game, $t = 1, \ldots, \infty$
 2: $dr\_cmd = CNN(I_t)$                       ◁ Use the CNN model to predict driving commands
 3: $(x^i, y^i, w^i, h^i) = obj\_det(I_t)$     ◁ Detect if there is any object of interest
 4: if $(x^i, y^i, w^i, h^i)$ then             ◁ If there is any object detected
 5:   $data_t = grab(I_t, (x^i, y^i, w^i, h^i))$   ◁ Save the frame and ground truth
 6:   $apx\_dis = depth(I_t)$                  ◁ Predict an approximate distance of the scene
 7:   if $apx\_dis \leq 0.5$ then              ◁ If the distance of an object is closer than 0.5
 8:     $mid\_pt = x + w/2$                    ◁ Compute the middle point of the object
 9:     if $0.3 < mid\_pt < 0.7$ then          ◁ If the object is on the trajectory of our agent
10:       $brake = ''S''$                      ◁ Enter key "S" for brake and stop our agent
11:     end if
12:   end if
13: end if
15: output: Save $\{data_t\}$                  ◁ Save raw all the frames from data collection
```

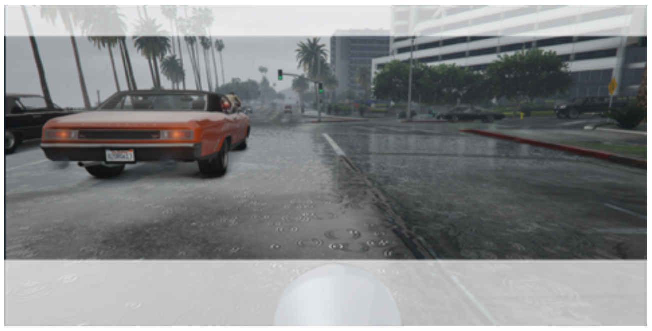


Fig. 3: Image pre-processing. The sky and headlight of the scooter are removed.

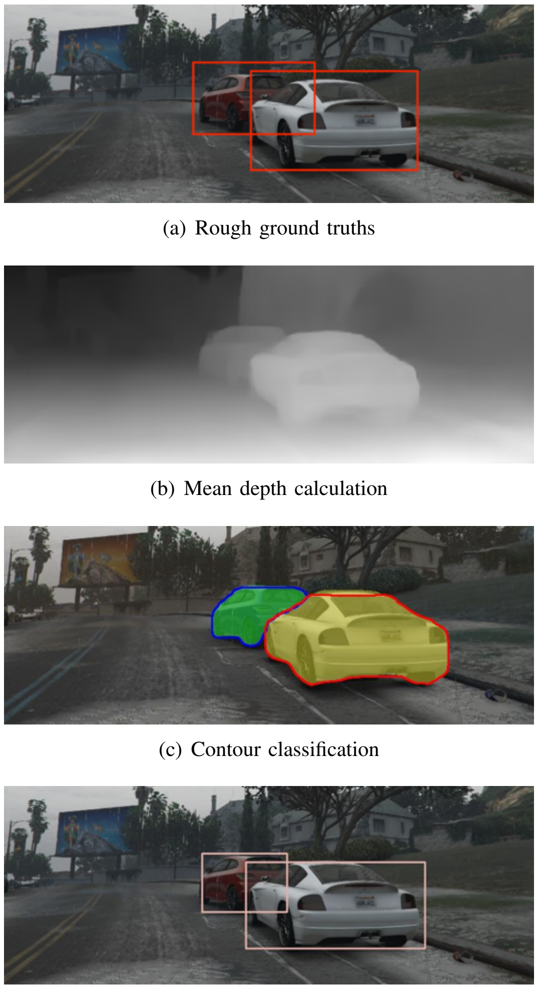
the performance of each dataset.

All the trained object detection networks are assessed on our self-prepared testing dataset. We use the object detection criteria from PASCAL [3] to evaluate the performance of each trained network. Considering the fact that there is only one labeled class (cars), we evaluate the detection accuracy based on average precision (AP). The intersection over union (IoU) between the ground truth and the predicted bounding box are calculated for AP. Following the design from [38], we use 70%, 80%, and 90% IoU thresholds to determine the true positive detection result. Namely, for each detection inference, the IoU needs to meet the designated threshold to be counted as a successful inference.

IV. Results Analytics

A. *Autonomous Data Collection*

The self-driving agent can typically stay within the appropriate driving lane in the game and thus collect data autonomously. If in danger of running off a curved roadway, the self-driving agent is capable of recovering and driving back onto the road. On average, it can drive in the game without human intervention for 22 minutes. Major exceptions occur under three circumstances. First, when the self-driving agent has to make decision at intersections for turning left or right, it may enter the wrong road or run off the road, which can cause collisions with other cars and pedestrians. Second, when the self-driving agent encounters red lights, it cannot read the signals for stopping or continuing. This also causes collisions. Finally, if an object suddenly moves close to our self-driving agent, this can also lead to problems for the self-driving agent.


(a) Rough ground truths

(b) Mean depth calculation

(c) Contour classification

(d) Improved ground truths

Fig. 4: Labeling protocol. In (a), our annotator first produces rough ground truths of each object. In (b), the mean depth of each object's appearance is computed to resolve the issue of occlusion. In (c), each object of interest is masked with different colors which help to draw their distinct contours. In (d), ground truths are improved effectively.

B. Simulation Dataset

We select and label a total of 300,000 rainy and snowy road images from the simulation. In our dataset, we only have a single "car" class which includes buses, trucks, vans, motorcycles, and cars. Since we set the time flow rate in the game to a fast mode, we are also able to collect data under different lighting conditions.

Compared to state-of-the-art human annotated datasets such as KITTI [3] and BDD (only snowy and rainy data) [6], our dataset size is tremendously large (as shown in Table I). With our automated data collection and annotation tools, we can conveniently enlarge the size of the dataset. Our dataset meets the need for data reflecting special weather conditions. In addition, since our simulation data are collected from diverse road types and at a range of times, our dataset includes more diversity in terms of road structures and lighting conditions compared to existing real-world datasets.

To evaluate the object appearance distributions from each dataset, we calculate the bounding box centroids across our dataset, KITTI [3], and BDD (only snowy and rainy data) [6]. We employ heatmaps to visualize the distribution of the object appearance (as shown in Fig. 5). It can be seen that our simulation dataset has a much larger spread of distribution in comparison with the two real-world datasets. In addition, while most of the instances appear in the center of the images, the populous density of instances for our dataset is much higher than Cityscape and KITTI.

TABLE I: The number of images from BDD, KITTI, and our simulation dataset.

Dataset	Weather		
	Rain	Snow	Total
Ours	**145,039**	**154,961**	**300,000**
KITTI	N/A	N/A	5,000
BDD	5,070	5,549	10,619

C. Comparison with State-of-the-art Datasets

1) Quantitative Results: We evaluate the performance of these datasets by comparing the detection accuracy of each network trained by each dataset. Since all the networks are trained in the same experiment settings, the training dataset is the only contributing factor which would differentiate networks' detection accuracy. The evaluation results are reported in Table II. Based on the results, networks trained by our dataset have the best overall performance in detection accuracy. For testing on both rainy and snowy images, our dataset assists networks in achieving higher detection accuracy under different IoU thresholds.

For R-FCN, the network trained by our dataset led the BDD-trained network by 2%, 0.5%, and 0.1% and the KITTI-trained network by by 6.5%, 3.9%, and 7.3% under the rainy condition; the network trained by our dataset led the BDD-trained network by 4.9%, 0.4%, and 2.1% and the KITTI-trained network by 8%, 5.5%, and 10.6% under the snowy condition.

For Faster-RCNN, the network trained by our dataset led the BDD-trained network by 4.2%, 0.5%, and 0.1% and the KITTI-trained network by 6.4%, 6.5%, and 2.4% under the rainy condition; the network trained by our dataset led the BDD-trained network by 4%, 2.6%, and 2.8% and the KITTI-trained network by 6.3%, 3.9%, and 0.3% under the snowy condition.

For YOLOv3, the network trained by our dataset led the BDD-trained network by 3.4%, 3.5%, and 4.3% and the KITTI-trained network by 6.8%, 7.5%, and 8.8% under the rainy condition; the network trained by our dataset led the BDD-trained network by 4.5%, 3.2%, and 2% and the KITTI-trained network by 5.2%, 5.3%, and 3.6% under the snowy condition.

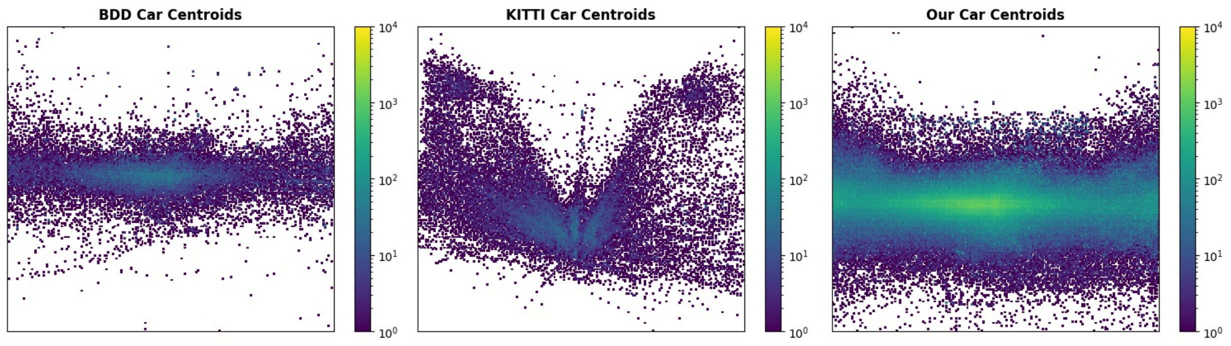


Fig. 5: Heatmaps of the bounding box centroids for BDD, KITTI, and our simluation dataset.

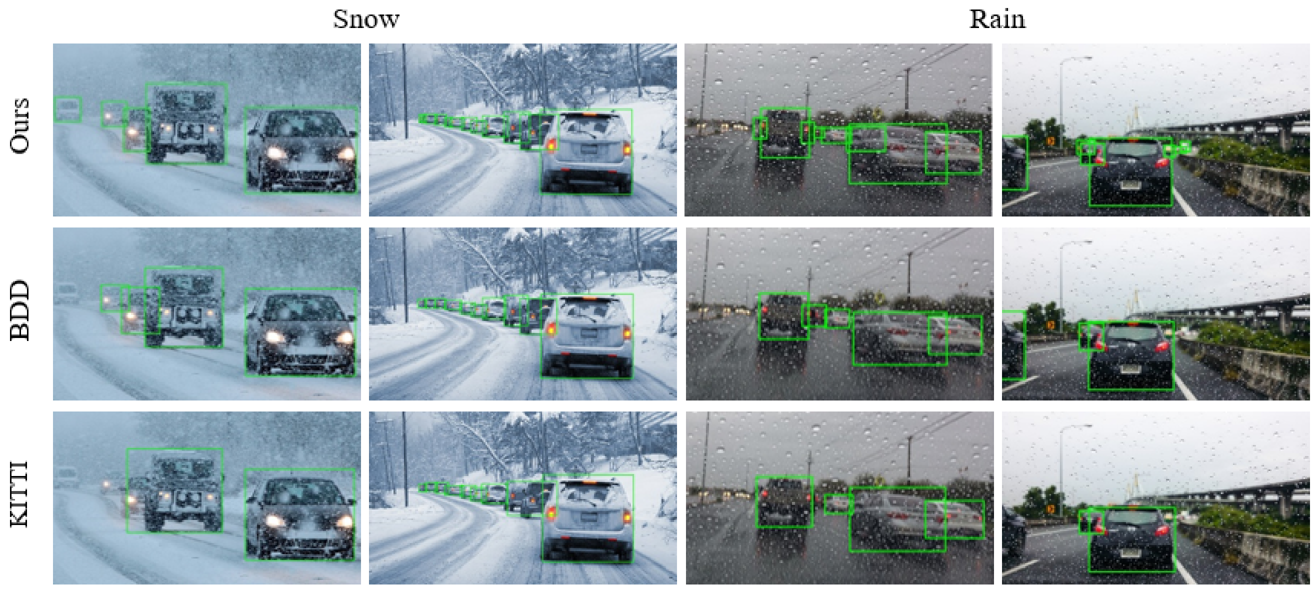


Fig. 6: Demonstration of detection results from networks trained by ours, KITTI and BDD datasett. We use examples from R-FCN. The top row demonstrates examples from a network trained by our simulation dataset; the middle row demonstrates examples from a network trained by KITTI; and the last row demonstrates examples from a network trained by BDD.

For SSD, the network trained by our dataset led the BDD-trained network by 4.4%, 6%, and 5.1% and the KITTI-trained network by 4.8%, 7.6%, and 5.4% under the rainy condition; the network trained by our dataset led the BDD-trained network by 1.8%, 2.7%, and 0.1% and the KITTI-trained network by 3.7%, 6.3%, and 0.8% under the snowy condition. These results demonstrate that the networks trained by our simulation dataset predict a more accurate ground truth compared to that predicted by its counterparts.

2) Qualitative Analysis: We visualize the qualitative results of each trained model for comparisons (as shown in Fig. 6). We notice that networks trained by our large-scale dataset are more capable of detecting smaller objects or objects with severe occlusion or truncation. In contrast, networks trained by KITTI and BDD struggle to handle objects which have severe occlusion or truncation or are a large distance away are normally smaller than 20×20 pixels.

Based on the evaluation results, we argue that simulation datasets can be a powerful technique to train networks and to resolve the limitations of real-world datasets. Our dataset outperforms its two counterparts because its size is much larger than real-world datasets and our dataset includes more diverse imagery information. Specifically, although BDD claims that it includes snowy and rainy days, most of pictures included in its dataset are not representative of the two weather conditions (as shown in Fig. 7). In contrast, our approach addresses the challenge of data diversity and quality effectively. After comparing our simulation dataset with two prestigious real-world datasets, we argue that our dataset can compete against the best existing dataset for object detection in autonomous driving tasks. However, the proposed rule for breaking actions may not work for objects moving toward a car agent itself because a car agent must predict the direction and approaching speed of other car agents. We will introduce extra information

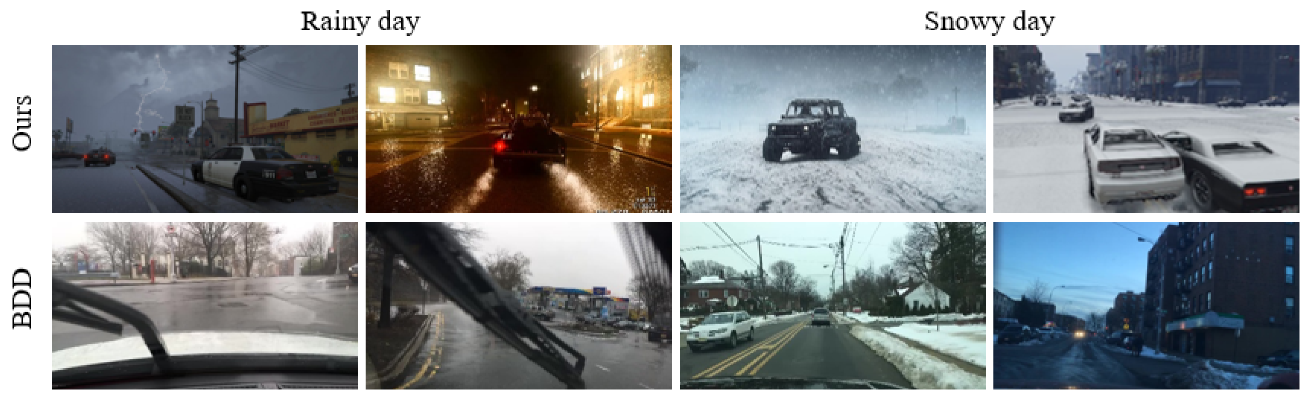


Fig. 7: Comparison of data between our dataset and BDD. The bottom row shows images from BDD which frequently are blocked by the wiper. In general, the BDD images have little representation of rainy and snowy conditions. The top shows images are from our dataset which offer clear objects and high resolution in the two weather conditions.

like optical flow to help predict the direction and approaching speed to improve the performance which will serve as our future works to update our dataset. Also, for the evaluation of our dataset, the cross-validation of three datasets may not emphasize the attributions of the proposed dataset entirely and clearly due to the space limitation. We will update the dataset and add more detailed evaluation experiments about its intrinsical characteristics in our extended future works.

TABLE II: Detection results for networks trained by KITTI, BDD, and our simulation dataset. All reports are in % and best results.

Dataset	Network	Rainy Images			Snowy Images		
		$AP^{70\%}$	$AP^{80\%}$	$AP^{90\%}$	$AP^{70\%}$	$AP^{80\%}$	$AP^{90\%}$
KITTI	R-FCN	83.2	73.9	56.1	79.3	68.7	55.6
BDD	R-FCN	87.7	77.3	63.3	82.4	73.8	64.1
Ours	R-FCN	**89.7**	**77.8**	**63.4**	**87.3**	**74.2**	**66.2**
KITTI	Faster-RCNN	81.9	71.6	63.2	80.6	70.8	65.6
BDD	Faster-RCNN	84.1	76.1	62.5	82.9	72.1	63.1
Ours	Faster-RCNN	**88.3**	**78.1**	**65.6**	**86.9**	**74.7**	**65.9**
KITTI	YOLOv3	79.8	69.8	60.1	80.1	69.8	60.2
BDD	YOLOv3	83.2	73.8	64.6	80.8	71.9	62.6
Ours	YOLOv3	**86.6**	**77.3**	**68.9**	**85.3**	**75.1**	**64.6**
KITTI	SSD	77.8	68.5	60.3	78.2	66.9	59.6
BDD	SSD	78.2	70.1	60.6	80.1	70.5	61.3
Ours	SSD	**82.6**	**76.1**	**65.7**	**81.9**	**73.2**	**61.4**

D. A Mixed Training with Real-world Dataset

With a larger amount of simulation data, we achieve a superior performance compared to networks trained with only real-world images and data. We further evaluate the contribution of our dataset by training object detection networks with both our simulation dataset and a real-world dataset. The results are presented in Table III. Based on the results, we notice that networks trained by a mixture of our dataset and a real-world dataset achieve further improvement in detection accuracy. The mixture training method improves detection accuracy on a recognizable scale. This result indicates that we can use our large-scale dataset to pre-train an object detection network and then use a customized dataset to fine-tune the network and achieve more robust results in a specific context.

TABLE III: A mixed training. All reports are in % and best results.

Dataset	Network	Rainy Images			Snowy Images		
		$AP^{70\%}$	$AP^{80\%}$	$AP^{90\%}$	$AP^{70\%}$	$AP^{80\%}$	$AP^{90\%}$
Ours+KITTI	R-FCN	90.6	79.4	64.2	88.4	75.8	67.0
Ours+BDD	R-FCN	91.5	80.9	65.1	89.3	78.7	69.6
Ours	R-FCN	**89.7**	**77.8**	**63.4**	**87.3**	**74.2**	**66.2**
Ours+KITTI	Faster-RCNN	89.1	79.3	67.5	87.9	75.3	66.2
Ours+BDD	Faster-RCNN	90.8	80.6	69.2	88.6	76.8	68.6
Ours	Faster-RCNN	**88.3**	**78.1**	**65.6**	**86.9**	**74.7**	**65.9**
Ours+KITTI	YOLOv3	87.2	78.4	69.6	86.8	7.9	65.6
Ours+BDD	YOLOv3	89.3	79.7	70.2	90.2	79.2	66.9
Ours	YOLOv3	**86.6**	**77.3**	**68.9**	**85.3**	**75.1**	**64.6**
Ours+KITTI	SSD	83.3	77.2	66.6	82.1	74.5	62.3
Ours+BDD	SSD	87.8	78.8	68.2	84.3	76.9	69.1
Ours	SSD	**82.6**	**76.1**	**65.7**	**81.9**	**73.2**	**61.4**

V. Conclusion and Future Research

In this work, we explore the power of employing simulation data to train networks to effectively detect objects under special weather-related road conditions. We introduce a innovative way to collect a large-scale simulated dataset with a focus on facilitating object detection in autonomous driving under rainy and snowy weather conditions. Considering the fact that special weather conditions are missing from current public datasets, our work feeds a need for data about such road context. We use extensive experiments to validate our simulation dataset by comparing with KITTI and BDD. Results indicate that our dataset is competitive with current state-of-the-art datasets. The experiment results demonstrate the effectiveness of using simulation data to meet real-world challenges in practice.

References

[1] 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, 2010.

[2] D. Liu, Y. Wang, K. E. Ho, Z. Chu, and E. T. Matson, “Virtual world bridges the real challenge: Automated data generation for autonomous driving,” in *2019 IEEE Intelligent Vehicles Symposium Conference*. IEEE, 2019, pp. 159–164.

[3] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[4] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset,” in *CVPR Workshop on the Future of Datasets in Vision*, vol. 1, no. 2, 2015, p. 3.

[5] Y. Wang, D. Liu, H. Jeon, Z. Chu, and E. Matson, “End-to-end learning approach for autonomous driving: A convolutional neural network model,” in *Proceedings of the 11th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, INSTICC. SciTePress, 2019, pp. 833–839.

[6] H. Xu, Y. Gao, F. Yu, and T. Darrell, “End-to-end learning of driving models from large-scale video datasets,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[7] W. Qiu and A. Yuille, “Unrealcv: Connecting computer vision to unreal engine,” in *European Conference on Computer Vision*. Springer, 2016, pp. 909–916.

[8] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in *European Conference on Computer Vision*. Springer, 2016, pp. 102–118.

[9] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[10] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1–8.

[11] D. Hoiem, J. Hays, J. Xiao, and A. Khosla, “Guest editorial: Scene understanding,” *International Journal of Computer Vision*, vol. 112, no. 2, pp. 131–132, 2015.

[12] D. Liu, Y. Wang, and T. Chen, “Application of color filter adjustment and k-means clustering method in lane detection for self-driving cars,” in *2019 IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 153–158.

[13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] Y. Wang, D. Liu, H. Jeon, Z. Chu, and E. T. Matson, “End-to-end learning approach for autonomous driving: A convolutional neural network model,” in *Proceedings of the 11th International Conference on Agents and Artificial Intelligence: ICAART*, 2019.

[16] M. Zhang, Z. Lin, Y. Cui, F. Shen, and Y. Shen, “Multiclassification method for hyperspectral data based on chernoff distance and pairwise decision tree strategy,” in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2016, pp. 505–508.

[17] M. Zhang, W. Guo, Y. Cui, F. Shen, and Y. Shen, “Manifold learning based supervised hyperspectral data classification method using class encoding,” in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2016, pp. 3160–3163.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[19] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[20] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 891–898.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[22] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, and A. M. Lopez, “Vision-based offline-online perception paradigm for autonomous driving,” in *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2015, pp. 231–238.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[24] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[25] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool, “Dynamic 3d scene analysis from a moving vehicle,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[26] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[27] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth, “Efficient multi-cue scene segmentation,” in *German Conference on Pattern Recognition*. Springer, 2013, pp. 435–445.

[28] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, “Bdd100k: A diverse driving video database with scalable annotation tooling,” *arXiv preprint arXiv:1805.04687*, 2018.

[29] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, “Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[30] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade, “Learning scene-specific pedestrian detectors without real data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3819–3827.

[31] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[32] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[33] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *CVPR*, 2017.

[34] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” 2016.

[35] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[37] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[38] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” *arXiv preprint arXiv:1610.01983*, 2016.