Simulated RGB and LiDAR Image based Training of Object Detection Models in the Context of Autonomous Driving

Durga Prasad Bavirisetti[1][0000−0001−9743−1701], Eskild Brobak[1], Peder Espen[1], Gabriel Kiss[1], and Frank Lindseth[1][0000−0002−4979−9218]

Department of Computer Science, Norwegian University of Science and Technology, 7034 Trondheim, Norway
{durga.bavirisetti, gabriel.kiss, frankl}@ntnu.no
{eskildbr, phespen}@stud.ntnu.no

Abstract. The topic of object detection, which involves giving cars the ability to perceive their environment has drawn greater attention. For better performance, object detection algorithms often need huge datasets, which are frequently manually labeled. This procedure is expensive and time-consuming. Instead, a simulated environment provides full control over all parameters and enables automated image annotation. Carla, an open-source project created exclusively for the study of autonomous driving, is one such simulator. This study examines if object detection models that can recognize actual traffic items can be trained using automatically annotated simulator data from Carla. The findings of the experiments demonstrate that optimizing a trained model using Carla's data, along with some real data, is encouraging. The Yolov5 model, trained using pretrained Carla weights, exhibited improvements across all performance metrics compared to one trained exclusively on 2000 Kitti images. While it didn't reach the performance level of the 6000-image Kitti model, the enhancements were indeed substantial. The mAP0.5:0.95 score saw an approximate 10% boost, with the most significant improvement occurring in the Pedestrian class. Furthermore, it is demonstrated that a substantial performance boost can be achieved by training a base model with Carla data and fine-tuning it with a smaller portion of the Kitti dataset. Moreover, the potential utility of Carla LiDAR images in reducing the volume of real images required while maintaining respectable model performance becomes evident. Our code is available at: https://tinyurl.com/3fdjd9xb.

Keywords: Obtection detection · Synthetic data · RGB · LiDAR · Autonomous driving · Deep learning · Computer vision.

1 Introduction

According to a research by the U.S. Department of Transportation [1], between 94 and 96 percent of all automobile accidents, were the result of human error.

A human driver must be able to perceive and comprehend a wide range of various things, including vehicles, pedestrians, and other road users like cyclists. Therefore, making a vehicle capable of precisely detecting these items is crucial to the sustainability of autonomous cars. This research focuses on object detection of dynamic traffic objects such as vehicles and people which is a subset of this broad and developing field of autonomous vehicles.

According to Zou et al. [2], requirement of having a large enough dataset is one of the major problems in object detection model training. For instance, the COCO dataset [3] has approximately 328,000 images of day to day objects and humans that are categorized into 80 different categories. To train an object detection model, images have to be manually labeled, which is a tedious and time-consuming task. In this study, we examined how effectively an object detection model, trained on simulated RGB and LiDAR image data gathered from the Carla (Car Learning to Act) simulator [1] can identify actual traffic objects.

The structure of the remaining paper is as follows. Section II introduces the existing literature in the research direction. Section III presents the methods and tools used for the current study. Section IV describes the different experiments that were done, along with their results. Finally, concluding remarks and future research directions are discussed in section V.

2 Literature Review

An approach to automatically acquire annotated image data using the Carla simulator is described in a paper by Jang et al [4]. Semantic segmentation is also used to correct the collected bounding boxes. The paper aimed to enhance the reliability of synthetic data collected from Carla by simplifying the process. It was examined by Dworak et al. [5] that if deep learning object detection models could be trained using LiDAR data obtained from the Carla simulator. They discovered that while combining the real and synthetic data for training had no positive effect on the models, it may still be employed for fine-tuning. It was investigated in a report written by Niranjan et al. [6] if data obtained from the Carla simulator might be used to train object detection algorithms. They stated that Carla's open-source nature, compatibility, and environment modeling give it an edge over rival simulators. Bu et al. [7] published a new paper that made use of Carla simulator data. They wanted to find out if object detection algorithms could be trained using data taken from Carla. They concluded that models can be successfully trained to recognize low-quality real-world objects using data from the Carla simulator.

Whether artificial data could be used to train deep networks was examined in research by Tremblay et al. [8]. In order to reduce the amount of annotation required, the research claims that portions of real datasets might be replaced with synthetic data. The hardest part of this research is making backgrounds that are natural and photorealistic, which takes a lot of time and effort. Abu

[1] https://carla.org/

Alhaija et al. [9] tried a hybrid approach to fix this by employing real-life images as backgrounds and realistically integrating photo-realistic 3D objects into the scenes. They discovered that the model trained on their augmented dataset outperformed both the real-world and synthetic datasets in terms of generalization. Tsirikoglou et al. [10] created a comparable method in which very realistic synthetic data with pixel-accurate annotations were produced to support CV tasks including semantic segmentation.

Richter et al. [11] investigated the possibility of creating pixel-accurate semantic label maps for images derived from the realistic open-world video game GTA V. They created a database of 25,000 images and trained a detection model using this dataset. Their findings demonstrated that a model enhanced with synthetic data not only produced noticeably better improvements but also allowed for a reduction in the amount of hand-labeled data. The performance of a model improved when it was trained using both their dataset and only a third of the real-world dataset.

The research goal for this paper is to investigate whether automatically annotated images from the Carla simulator can be used to train object detection models for real traffic scenarios. To achieve this goal, two research questions (RQs) were posed:

• RQ 1: Is it possible to reduce the number of real images needed by fine-tuning an object detection model trained mainly on the simulated RGB image data gathered from the Carla simulator? • RQ 2: To what extent can the fine-tuning of a model that was previously trained on simulated LiDAR image data be utilized to minimize the number of real LiDAR images needed?

3 Methodology

Building on earlier research by Jang et al. [4], we address these RQs by gathering automatically annotated images using the open-source autonomous driving simulator Carla. We have collected 3D bounding boxes and RGB images from the simulation. Then, in order to correct them, these 3D boundary boxes were translated into 2D. The bounding boxes acquired using the paper Jang et al. [4] demonstrated that they were frequently inaccurate since they were larger than the boundaries of the objects. Research by Kervadec et al. [12] demonstrated the significance of tightness of bounding boxes in an ML scenario. Additionally, the very first research only classified objects into two categories: pedestrians and vehicles. Due to this, the diversity of the dataset was constrained as Carla has several types of other types of spawnable objects, like trucks and cyclists. In this project, we are particularly interested to build datasets that were more stable and balanced on groundwork laid by the paper [12]. This study investigates if the created datasets can be used to train a model to detect traffic items in real-life images in order to determine whether the gathered data is useful for object detection tasks. Even though experiments were conducted on both Yolov5 and Faster R-CNN algorithms, due to space constraint, experimental analysis related to single stage simple and efficient Yolov5 object detector is presented with differ-

ent parameter settings. The Kitti, a well known dataset for autonomous driving tasks, was used to validate the models.

In detail discussion on the methodology is explained in following subsections. The subsection 3.1 details the collection of simulator data in order to create a dataset. The next subsection 3.2 describes how this dataset was used to train object detection models to verify the research objectives.

3.1 Data collection pipeline

A data collection pipeline is developed to collect data and create the training datasets. First, the Carla server is started, and the actors are spawned after selecting the desired map and weather conditions. The number of actors depends on the size of the map that data was collected. If not enough actors were spawned in relation to the map size, many of the collected images would have few or no objects in view most of the time due to them being spread out. To ensure a consistent number of labels in the collected images, maps with relatively similar sizes were chosen. The chosen maps were Town01, Town02, and Town11. After setting up the Carla simulator with actors, map, and weather conditions, we have extracted both RGB and LiDAR images in autopilot mode and saved instance segmentation image, RGB image, and bounding box information for all the actors present in the current image view of the simulator for every five seconds. Tightened the bounding boxes further and saved them for the object detection process. In detail explanation of the pipeline can be seen in the following subsections.

Dynamic Objects creation Performing object detection requires the selection of object classes. As this project focused on detecting dynamic objects in a traffic situation, four classes were chosen. They are 0 - Car (regular passenger cars), 1 - Truck (bigger cars like vans or ambulances), 2 – Cyclist, and 3 – Pedestrians. The classes were partly chosen based on what objects were available in the Carla simulator and partly based on what classes were typically found in available real-life datasets for comparison.

Sensor Data (RGB/LiDAR) Acquisition There are primarily two ways one could go about gathering data from the Carla simulator: Manual and automatic controlling. The player vehicle would be fitted with selected sensors, which are accessible through the Python API. For example, RGB or LiDAR sensors attached to the vehicle can gather data for every game update.

RGB data Extraction from Carla The first step of the data collection process was to gather raw data from the simulator which was done using the script from the CarFree project [4]. For most of the data collection, the autopilot and automatic capture of images were used. The images were captured with a resolution of 960x540. The player vehicle was attached with the RGB camera

sensor and Instance Segmentation camera. In addition to saving the RGB and instance segmentation images, bounding boxes with 8 corners in (x, y, z) format were also collected and saved. A transformation has been applied to transform bounding boxes with respect to the camera view. This was done in two steps: first by transforming the actor coordinates to the world coordinates, and then from the world coordinates to the sensor coordinates. After this, the coordinates are projected from 3D into 2D to achieve bounding boxes on images using camera calibration [6]. As mentioned before CarFree only used two labels whereas this project aimed at 4 labels for the dynamic objects. A different approach has been taken to add additional labels aimed at converting 3D bounding boxes to 2D for creating correct labels which include extended classes.

Extracting LiDAR data from Carla The LiDAR extraction process from the Carla simulator is also similar to the extraction of the RGB data as discussed in section 3.1 which can collect Point cloud files at a rate of one every 5 seconds. The Carla LiDAR sensor has several attributes that need to be tweaked before collecting data. LiDAR data collection in this research is conducted using the NTNU research platform, which is a Kia e-Niro (fully electric) equipped with three LiDARs (one 360 degrees 128-channel at the top, one 180 degrees 16 channel in the front and one close to 270 degrees looking at the rear right side), other sensors and software. We try to simulate LiDAR images with the top mounted LiDAR configuration as in Table 1. Using these settings, 20.ply files were gener-

LiDAR Specification	Value
Channels	128
Range	240
Upper and lower FOV	-11.25 to 11.25
Rotation frequency	20 FPS
Points per second	2,621,440

Table 1. Simulated LiDAR specifications.

ated and LiDAR point cloud scans were collected at a similar pace to the RGB images (one image every 5 seconds). This would allow for a greater variance in the images, as 20 images each second would result in many very similar images.

Generating LiDAR annotated data *Projecting 3D points to 2D image space*: In order to create an image, the 3D point cloud needed to be projected to a 2D image space. Wu et al. [13] described one method to achieve this. This is done by calculating where the 3D points will fit as pixel coordinates, as well as what the pixel value should be. Figure 1 shows an example of the generated range and intensity images by 3D to 2D image projection on our one LiDAR scan.

Automatic annotation for Lidar images: After projecting 3D points to 2D image space, the next step is to the annotation of LiDAR images automatically.

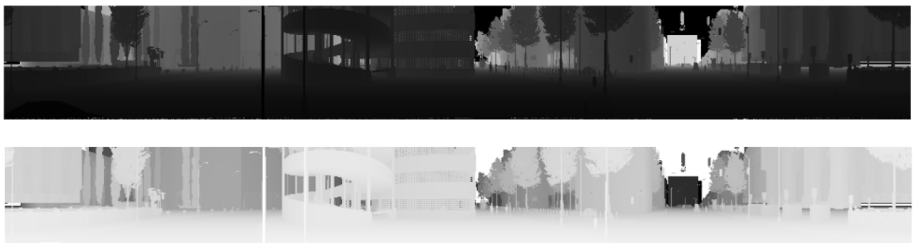

Fig. 1. Example range image (top) and intensity image (bottom).

A similar method to the RGB images was applied when generating the LiDAR bounding boxes. While the semantic LiDAR sensor did not provide instance colors for each object, it did provide the actor ID directly. As such, a slightly different approach had to be used compared to the RGB images.

A case was found where certain objects could appear on both sides of the image. Since the image was 360 degrees, an actor could appear split on both ends of the image horizontally. This would cause issues with the object's bounding box, as the bounding box would stretch across the entire image. To avoid this, any object where the difference between max and min X above 900 pixels was ignored. Another option would be to treat these two parts as separate objects, but this seemed like a sub-optimal solution from an annotation perspective.

As with the RGB images, bounding boxes that were too small were removed. However, unlike for the RGB images, where a percentage of the x and y values was chosen as a threshold for removal, the area of the bounding box was calculated and used as a threshold. Compared to the RGB images, the LiDAR images had more slim and tall objects, particularly pedestrians. Using an area as the threshold preserved these bounding boxes while still removing small boxes that could be a detriment to model training. Figure 2 shows the effect of using the threshold on a range image. Note the smaller objects being removed in the bottom image.

Fig. 2. LiDAR range image without bounding box threshold (top) vs with threshold (bottom).

Real RGB images for testing: The team needed a dataset with properly annotated real images to test the models. A popular dataset used for 2D traffic object detection is the Kitti dataset[2]. It contains roughly 7500 annotated images. This dataset was chosen due to the quality of the annotations, as well as it having annotations for all the relevant label types used in this project. For the Carla dataset, the team initially collected 500 test images to review the dataset before collecting a larger sample. The images were all collected from Town11.

Real LiDAR images for testing: Similar to RGB images, a test dataset was needed to test the generated LiDAR images. As mentioned previously, LiDAR data collected from our research platform and nearly 5000 images were hand-annotated for the Object detection task in Darknet and COCO formats. Mainly four classes (car, person, rider (cyclist), and bus) were considered. One issue that had to be addressed was the mismatching labels between the simulated and real data. The data provided for the project did not contain any truck class, however, it did contain a bus class. We have decided to simply change the class name in the project dataset, with the reasoning that some of the trucks in Carla like the fire truck or ambulance look fairly similar to buses.

Dataset format conversion: Yolov5 uses the Darknet Pytorch dataset format while Carla simulator uses the regular Darknet format for bounding box information. To convert between these formats, a website called Roboflow[3] was used.

Dataset imbalance: We have to make sure that the collected datasets did not suffer from significant class imbalance. According to Oksuz et al. [14], an imbalanced dataset can have adverse effects on the final detection performance of object detection models. As mentioned previously, a test dataset consisting of 500 images was initially collected. It was discovered that the 500-image test dataset suffered from a significant class imbalance, which is illustrated in Figure 3 (left).

Fig. 3. Class imbalance for 500 image test dataset Before (left) vs After (right) spawn balancing.

As the figure shows, the cyclist class was severely underrepresented whereas the car was over represented in the dataset. After investigation, it is found that

[2] https://www.cvlibs.net/datasets/kitti/
[3] https://roboflow.com/

the class imbalance is arising while spawning actors during the simulation due to the random selection of vehicles from the list of vehicle blueprints, where the Carla blueprint library had significantly more car blueprints than trucks or cyclists (Car:25, Truck:7, Cyclists:3). To fix this issue and balance the spawning, we update the existing code to spawn a certain amount of each type of vehicle. We have decided that 20 of each vehicle type was enough for the size of the selected maps, for a total of 60 vehicles. There was still some class imbalance. However, it was an improvement over the original dataset as shown Figure 3 (right). By following the approach class balance for LiDAR images was also addressed. After the imbalance was addressed on 500 images initially, 7500 images from the Carla simulator were gathered to match the Kitti dataset size. The images were equally split between the three maps, with each map having three separate weather conditions: clear, night and rain. Both datasets were split, with 80% reserved for training, 10% for testing and 10% for validation.

For the data collection of LiDAR images from the Carla simulator, the team opted for a total of 5000 images to match the testing dataset size, with 2500 range and 2500 intensity images. This was done by collecting an equal amount of .ply files from the same three towns as the RGB images. The LiDAR images would not be affected by world conditions like rain and lighting, so these settings were not considered. The same 80/10/10 split was used for this dataset.

3.2 Training and testing object detection models

One of the reasons for picking Yolov5 for this research was its ease of use in everything from training on custom data to testing the trained models. There are several versions of Yolov5 available; Yolov5n (nano), Yolov5s (small), Yolov5m (medium), Yolov5L (large) and Yolov5X (XL). The primary difference is a trade-off in speed vs. accuracy. The smaller models like nano and small could run inference (detect objects in images) faster, but are not as accurate as the larger models. We have decided that a balance between speed and accuracy was the best choice, resulting in Yolov5m being chosen. For more information about the model architecture, one could refer to github repo [4]. In an interview [5], Glenn Jocher (the lead developer on the Yolov5 project), shared an insight on why YoloV5 is an easy and efficient method. According to him, the new contribution in Yolov5 was the introduction of genetic anchors. Instead of using pre-existing anchors, the model uses genetic learning algorithms to create new anchors based on the bounding boxes of the custom dataset. This enables the model to train using custom datasets much more easily than previous versions, with no modification of the model needed.

Testing Yolov5 produced a variety of information and metrics about the models. In this research, we considered a most common performance measure by [15] in the context of object detection called Mean Average Precision (mAP) score as the main metric to compare the results from the experiments. Average

[4] https://github.com/ultralytics/yolov5/issues/6998
[5] https://blog.roboflow.com/yolov5-improvements-and-evaluation/

Precision (AP) for each class was also measured, as it could show discrepancies among different classes and how they could affect the mAP value. The mAP training values for a model trained exclusively on Carla would likely not be very useful when the research objective is to check how well a model trained on simulator data can perform on a real dataset. Hence, the models were tested using the test portions of the Kitti and Ouster LiDAR datasets.

4 Experiments & Results

This section details the experiments that were carried out in an attempt to answer the research questions posed in Section 2. First, baseline models were created which are used for comparison. A total of three experiments were conducted. Experiments 1 and 2 focus on RQ1. RQ2 is addressed in experiment 3, respectively. In this regard, experiments (Exp 1-2) pertaining to RGB camera are shown in the Table 2 whereas experiment (Exp 3) with regard to the LiDAR sensor are shown in the Table 3.

	Model	Train data		Test data	Epochs	mAP0.5	mAP 0.5:0.95	Car	Truck	Cyclist	Pedestrian
		Kitti	Carla RGB	Kitti							
	Baseline 1 (Yolov5)	6000	–	1500	300	0.936	0.703	0.812	0.822	0.680	0.497
Exp 1	Kitti 2k	2000	–	1500	300	0.863	0.592	0.748	0.711	0.537	0.374
	Kitti Transfer	2000	6000	1500	300	0.898	0.644	0.776	0.764	0.591	0.447
Exp 2	Carla RGB	–	6000	1500	300	0.168	0.078	0.221	0.052	0.005	0.037
	COCO weights+ Carla RGB	–	6000	1500	300	0.283	0.144	0.352	0.13	0.005	0.088

Table 2. Results of Experiments 1 and 2.

	Model	Train data		Test data	Epochs	mAP0.5	mAP 0.5:0.95	Car	Truck	Cyclist	Pedestrian
		Ouster	Carla LiDAR	Ouster							
	Baseline 2 (Yolov5)	4000	–	1000	300	0.877	0.522	0.691	0.498	0.475	0.425
Exp 3	Ouster 2k	2000	–	1000	300	0.631	0.346	0.442	0.469	0.245	0.202
	Ouster Transfer	2000	4000	1000	300	0.670	0.402	0.476	0.569	0.305	0.250

Table 3. Results of Experiment 3.

Training baseline models: Before any experiments were conducted, As shown in Table 2 and Table 3, two baseline models Baseline 1 and Baseline 2 were created; one for RGB and another LiDAR data respectively from the hand-annotated datasets. As mentioned previously, the Kitti dataset contained roughly 7500 annotated images. As per the 80/10/10 split, 6000 of the images were used

to train the baseline models, with the remaining 1500 images divided equally into the validation and testing sets. For the Ouster LiDAR data, the dataset was hand-annotated by researchers at NAPLab of NTNU. A total of 5000 LiDAR images, of which 4000 were used for training, and 500 each for test/valid. All trained models were tested on the test portion of their respective hand-annotated datasets. As mentioned, these scores represented the baseline against which the other experiments were compared.

Exp 1: Fine-tuning with Kitti images: It was conducted to investigate whether the Carla data could be used to train a baseline model, which could then be fine-tuned using a smaller subset of the Kitti data. 2000 images were selected randomly from the Kitti training dataset, and transfer learning was done on top of pre-trained Carla weights (6000 images, 300 epochs). Models were also trained from scratch using the same 2000 Kitti images to check if the pre-trained weights had any effect at all. The Yolov5 model trained using the pre-trained Carla weights showed an improvement across all metrics compared to the one trained solely on 2000 Kitti images. While not reaching the performance of the 6000 image Kitti model, the improvement was significant. The mAP0.5:0.95 score improved by roughly 10%, while the Pedestrian class showed the biggest improvement overall.

Exp 2: Fine-tuning on pre-trained COCO models: It was conducted to investigate whether models pre-trained on the COCO dataset could be used to enhance the data collected from Carla. A substantial increase in mAP can be seen, even doubling the value for the mAP0.5:0.95 metric. The only class which did not benefit from the pre-trained weights was the cyclist.

Exp 3: Fine-tuning with LiDAR images: It was conducted to investigate whether the Carla LiDAR images could be used to reduce the amount of real images required when training with LiDAR images. A subset of the Ouster training dataset consisting of 2000 randomly chosen images were used to fine-tune the models from exp 6 by doing transfer learning. Model trained using the 2000 images was also used to check whether the transfer learning had any effect. It is observed that the fine-tuned model saw substantial gains in its metrics compared to the model trained only on the 2k Ouster images. It can observed that all scores increased by between 5-14%, with the truck class seeing the largest increase. However, the scores did not reach the baseline.

Looking at the Exp 1, a model that was trained using pre-trained Carla weights demonstrated improvements across all metrics when compared to a model trained exclusively on Kitti images. The Exp 2 also demonstrates a significant improvement when training a base model on Carla data and fine-tuning with a small portion of the Kitti dataset which could be an application in training the building blocks of Yolov5 object detection models used in traffic environments. The Exp 3 indicates that the Carla LiDAR images could have a use in reducing the amount of real images required while still achieving decent model performance.

5 Conclusion and Future work

In conclusion, this study has highlighted the significance of object detection in the context of autonomous driving and the challenges associated with acquiring large labeled datasets. The use of a simulated environment, such as Carla, for automated image annotation has shown promise in mitigating these challenges. Our experiments have demonstrated that training object detection models with automatically annotated data from Carla can yield encouraging results.

Specifically, the Yolov5 model, optimized using pretrained Carla weights, exhibited significant improvements across various performance metrics when compared to a model trained solely on a limited set of 2000 Kitti images. While it did not match the performance of the 6000-image Kitti model, the enhancements were substantial, particularly in the Pedestrian class where the most significant improvement was observed, leading to an approximate 10% increase in mAP0.5:0.95 score.

Furthermore, our findings suggest that a substantial performance boost can be achieved by initially training a base model with Carla data and then fine-tuning it with a smaller portion of the Kitti dataset. This approach offers a potential solution to reduce the dependency on a large volume of real images while maintaining respectable model performance.

Additionally, the study highlights the valuable role of Carla LiDAR images in augmenting the dataset, further reducing the need for a massive collection of real images. In essence, our research demonstrates the feasibility and effectiveness of leveraging simulated environments like Carla for training object detection models, which can be instrumental in advancing autonomous driving technology.

The possibility of extracting 2D bounding boxes directly from Carla rather than having to project the 3D bounding boxes into the 2D is one of the future directions we think is worth exploring. Other potential directions include the effect of image scale variation on object detection performance, and the study of specific aspects of the data collection process.

References

1. US Department of Transportation. 2016 fatal motor vehicle crashes: Overview. `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812456`, 2017. [Online; Last accessed 27 September 2022].
2. Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.
3. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
4. Jaesung Jang, Hyeongyu Lee, and Jong-Chan Kim. Carfree: Hassle-free object detection dataset generation using carla autonomous driving simulator. *Applied Sciences*, 12(1):281, 2021.

5. Daniel Dworak, Filip Ciepiela, Jakub Derbisz, Izzat Izzat, Mateusz Komorkiewicz, and Mateusz Wójcik. Performance of lidar object detection deep learning architectures based on artificially generated point cloud data from carla simulator. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 600–605. IEEE, 2019.
6. DR Niranjan, BC VinayKarthik, et al. Deep learning based object detection model for autonomous driving research using carla simulator. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 1251–1258. IEEE, 2021.
7. Tom Bu, Xinhe Zhang, Christoph Mertz, and John M Dolan. Carla simulated data for rare road object detection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2794–2801. IEEE, 2021.
8. Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.
9. Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9):961–972, 2018.
10. Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. Procedural modeling and physically based rendering for synthetic data generation in automotive applications. *arXiv preprint arXiv:1710.06270*, 2017.
11. Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.
12. Hoel Kervadec, Jose Dolz, Shanshan Wang, Eric Granger, and Ismail Ben Ayed. Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision. In *Medical Imaging with Deep Learning*, pages 365–381. PMLR, 2020.
13. Tao Wu, Hao Fu, Bokai Liu, Hanzhang Xue, Ruike Ren, and Zhiming Tu. Detailed analysis on generating the range image for lidar point cloud processing. *Electronics*, 10(11):1224, 2021.
14. Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3388–3415, 2020.
15. Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016.