Improve Bounding Box in Carla Simulator

Mohamad Mofeed Chaar[a], Jamal Raiyn[b] and Galia Weidl[c]
Connected Urban Mobility, Faculty of Engineering,
University of Applied Sciences, Aschaffenburg, Germany

Keywords: Bounding Box, Carla Simulator, Object Detection, Deep Learning, Yolo.

Abstract: The CARLA simulator (Car Learning to Act) serves as a robust platform for testing algorithms and generating datasets in the field of Autonomous Driving (AD). It provides control over various environmental parameters, enabling thorough evaluation. Development bounding boxes are commonly utilized tools in deep learning and play a crucial role in AD applications. The predominant method for data generation in the CARLA Simulator involves identifying and delineating objects of interest, such as vehicles, using bounding boxes. The operation in CARLA entails capturing the coordinates of all objects on the map, which are subsequently aligned with the sensor's coordinate system at the ego vehicle and then enclosed within bounding boxes relative to the ego vehicle's perspective. However, this primary approach encounters challenges associated with object detection and bounding box annotation, such as ghost boxes. Although these procedures are generally effective at detecting vehicles and other objects within their direct line of sight, they may also produce false positives by identifying objects that are obscured by obstructions. We have enhanced the primary approach with the objective of filtering out unwanted boxes. Performance analysis indicates that the improved approach has achieved high accuracy.

1 INTRODUCTION

The procedure of bounding box in Carla simulation (Dosovitskiy et al., 2017) is implemented by invoking the Bounding Box object. This returns a 3D object box, which we then transform to align with the sensor's coordinate system on the ego vehicle, using vertex transformations as the following formula (Correll et al., 2022).

$$\grave{P} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} P \tag{1}$$

Where $P = (x, y, z, 1)$ is the map of coordinates and $\grave{P} = (\grave{x}, \grave{y}, \grave{z}, 1)$ represents the new coordinates with reference point R to the sensor on the vehicle:

$$R = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

and Translation T is 3x1 matrix (or vector) that translates a point in 3D space. Subsequently, we identify all vehicles that fall within the sensor's field of view, accounting for its height and width parameters. Finally, we project the 3D boxes to 2D because we work with images. In Carla, the bounding box accounts for the sensor's coordinates and field of view. However, it does not consider objects that are obscured by other structures, such as buildings. To address this limitation, our methodology involves refining the detection process. We synchronize the sensor data with a Semantic Segmentation camera (Dosovitskiy et al., 2017) By conducting a pixel-by-pixel comparison between the sensor and the Semantic Segmentation camera, we are able to filter out unwanted bounding boxes effectively (Figure1).

(a) (b)

Figure 1: Comparison between the image before (a) and after (b) the filter.

[a] https://orcid.org/0000-0001-9637-5832
[b] https://orcid.org/0000-0002-8609-3935
[c] https://orcid.org/0000-0002-6934-6347

Chaar, M., Raiyn, J. and Weidl, G.
Improve Bounding Box in Carla Simulator.
DOI: 10.5220/0012600500003702

In *Proceedings of the 10th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2024)*, pages 267-275
ISBN: 978-989-758-703-0; ISSN: 2184-495X

2 RELATED WORKS

The authors (Niranjan et al., 2021) Operated the Single Shot Detector (SSD) to train a data generated by the Carla simulator, which was created by capturing 1028 images from RGB sensor with size (640x380) pixels from different maps and different weather conditions. Because of the encountered problems in Carla with the auto generation of a bounding box from the simulator, the authors annotated the labels using Python GUI called labelimg to identify 5 object classes (Vehicles, Bike, Motorbike, Traffic light, and Traffic Sign). The paper provides a valuable contribution to the field of autonomous driving research by demonstrating the effectiveness of using the CARLA simulator to train and test deep learning-based object detection models. On the other hand, (Muller, 2022) generate a new methodology to collect a datasets automatically, which they filtered the objects by taking the semantic LiDAR data and remove all objects which are not in the point cloud of the Lidar sensor. The issue with the Lidar sensor is its lack of precision when dealing with distant objects. The sensor's accuracy diminishes proportionally with the distance to the objects. Additionally, it is highly affected by weather conditions. For instance, when operating in foggy conditions, it negatively impacts the sensor's accuracy. Therefore, we have implemented flexible criteria that align with this requirement. Further elaboration on these points will be presented in this paper. The work of (Mukhlas, 2020) utilized the depth camera to remove the non-visible vehicles by comparing the distance between the ego-vehicle and the object, i.e. if there is an obstacle, as a building, the real distance will be bigger than the measurement of the depth image, then such "ghost" vehicles will be removed. As we see in the Figure 2 this work is not accurate enough. Here we find that some objects are wrongly filtered. The reason for this issue, is that the distance is dependent on the center of the object and it does not take into account all object dimensions, e.g. such as the center of object is covered by obstacle but the object is partially visible (Figure 2). In addition, this work does not solve the problem of unwanted box, which covers a significant portion of an image. In comparison, our work is checking each pixel in the bounding box, leaves any existing object (even if only partially visible) and filters any object which is completely not visible. The authors (Rong et al., 2020) introduced the LGSVL Simulator, which is a high-fidelity simulator designed for developing and testing autonomous driving systems. The simulator provides a comprehensive environment that replicates real-world driving conditions, enabling developers to train and evaluate their autonomous vehicle algorithms under various scenarios. The LGSVL Simulator is a powerful tool for developing and testing autonomous driving systems. Its high-fidelity representation of the real world, end-to-end simulation flow, customization capabilities, and support for ROS and ROS2 make it an essential tool for researchers, developers, and companies working in the autonomous driving field. In datasets, we could use simulator data or synthetic data, the paper (Burdorf et al., 2022) investigates the use of synthetic data to reduce the reliance on real-world data for training object detection models. Object detection is a crucial task in many computer vision applications, including autonomous driving, robotics, and surveillance. However, collecting large amounts of high-quality real-world data can be expensive and time-consuming. Their paper conducts experiment to evaluate the effectiveness of using synthetic data for object detector training. The results show that synthetic data can significantly reduce the requirement for real-world data, with mixed datasets of up to 20% real-world data achieving comparable detection performance to datasets with 100% real-world data. Finally, as we are aware there is a data collected by real images, The Canadian Adverse Driving Conditions (CADC) dataset (Pitropov et al., 2021) is a publicly available dataset of annotated driving sequences collected in winter conditions in the Waterloo region of Ontario, Canada. It consists of over 20 kilometers of driving data, including over 7,000 annotated frames from eight cameras, a lidar sensor, and a GNSS+INS system. The dataset is annotated with a variety of objects, including vehi-

Figure 2: In the work of (Muller, 2022) the filter is using semantic LiDAR. One can see that two objects are filtered while they should be included in bounding boxed (see both down images). The significant box (only its red bottom line is visible (see Up left image)) was not filtered although it should be. In the filter using depth camera (Up right) one can still see that the object is existing and visible, although it has been wrongly filtered (no bounding box was put around it).

Figure 3: Semantic segmentation camera in Carla Simulator.

cles, pedestrians, cyclists, and traffic signs, as well as weather conditions, such as snow, rain, and fog. The CADC dataset is a valuable resource for researchers and developers working on autonomous driving systems. Its large size, diverse collection of driving sequences, high-quality annotations, and multiple sensor modalities make it a powerful tool for improving the performance and robustness of autonomous driving systems in winter conditions.

3 CONTRIBUTIONS

In this work, we have developed a new technique to collect the necessary data, just by generating it with Carla simulation (Chaar et al., 2023). Furthermore, we generated a new package to develop the Carla environments where we could control on the number of vehicles, pedestrians, environments, weather, etc by yaml files. The package, developed by us, is called carlasimu, and to simplify the work of other developers, we uploaded it to the GitHub link: (https://github.com/Mofeed-Chaar/Improving-bouning-box-in-Carla-simulator). The sensors that we have used for our developments in the Carla simulator are RGB Camera sensor and Semantic segmentation camera. You can find our code of the filter at the GitHub link above. As we are aware, the Semantic segmentation camera gives a special pixels colour for each object (Figure 3), in Carla simulator there are 27 classes of objects (Table 1). To improve the bounding boxes, we applied our work to filter the "ghost" object, which are occluded by e.g, buildings and which should not be present in the simulation. Simultaneously, we put bounding boxed around existing objects, which are partially covered by other objects. The task involves six distinct categories: cars, buses, trucks, vans, pedestrians, and traffic lights. It revolves the "ghost" bounding box around invisible cars, extracting a 2D bounding box, generated by the Carla simulator and conducting a pixel-wise comparison within the box using a semantic segmentation camera. The objective is to determine if the pixels within the box correspond to the object identified by the image Semantic Segmentation sensor. By this operator, we have got highly accurate results by using for training the data which has been collected by our filter (precision = 0. 968, Recall = 0.926, and mAP50 = 0.965) in YOLOv5s. We generated this training data form 8 maps under clear weather conditions.

Table 1: The colour of objects for Semantic segmentation camera in Carla simulator.

Class Name	Colour (R,G,B)
Unlabelled	(0, 0, 0)
Car and Truck	(0,0,142)
Bus	(0,60,100)
Van	(0,0,70)
Bicycle	(119,11,32)
Motorcycle	(0,0,230)
Building	(70, 70, 70)
Fence	(100, 40, 40)
Other	(55, 90, 80)
Pedestrian	(220, 20, 60)
Pole	(153, 153, 153)
Road Line	(157, 234, 50)
Road	(128, 64, 128)
Sidewalk	(244, 35, 232)
Vegetation	(107, 142, 35)
Wall	(102, 102, 156)
Traffic Sign	(220, 220, 0)
Sky	(70, 130, 180)
Ground	(81, 0, 81)
Bridge	(150, 100, 100)
Rail Track	(230, 150, 140)
Guardrail	(180, 165, 180)
Traffic Light	(250, 170, 30)
Static	(110, 190, 160)
Dynamic	(170, 120, 50)
Water	(45, 60, 150)
Terrain	(145, 170, 100)

4 BOUNDIG BOX IN CARLA

In Carla simulator, the developers use a method which can generate a bounding box (3D, and 2D) for all objects in the environment such as vehicles. For this purpose is used the python class in Carla package (carla.BoundingBox) - see the code in (Team, 2020).In this section we will discuss the issues in Auto Bounding box which is generated by Carla simulator.

The biggest problem appears during the genera-

tion of a bounding box in the negative values of the annotation and for objects, which are occluded by obstacles (Figure 4).

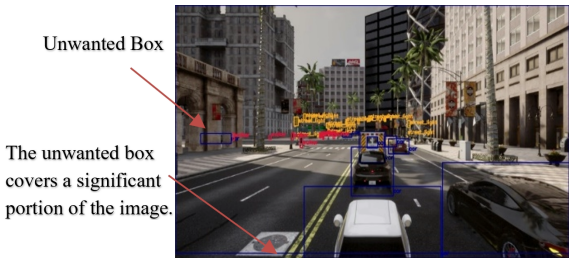

Figure 4: In Carla, as we've found out, objects are detected in bounding box even when they are obscured behind buildings. This affects data generation, where some labels of bounding boxes are false positives, impacting the accuracy of training data.

4.1 Negative Value in Annotations

The procedure of the bounding box in the Carla simulator: The bounding box is taken from all the environment, and we choose all the objects in the front of the sensor with respect to the FOV (Field of View) and vertical to the image in the sensor. The issue in this case is that some objects appear partially in the range of the sensor. This makes a negative value for the bonding box and makes the box appear behind the actual object (as in Figure 4) as a beam for 3D boxes and significant box for 2D bounding box. Thus, it does not fix the "ghost" box problem, when we correct the negative boxes to zero boxes. The previous works for filtering the bounding box (Section 2) did not fix this issue where we operate a special filter for negative notation. Therefore, we could generate a sample of "ghost"-problem typical data in our GitHub link by running the file (boundingBox.py).

4.2 Ghost Bounding Box

The bounding box which is generated by Carla simulator appears for all objects in the coordinate of the view of the camera (FOV and vertical view), and it does not take into account the occluded object by obstacles such as a wall or buildings (Figure 4). The previous works solved this problem, but it is not resolved accurately in some special cases (Section 2). Whereas we solved this issue with high accuracy i.e. if an object is present in the image by just one pixel, we can detect it.

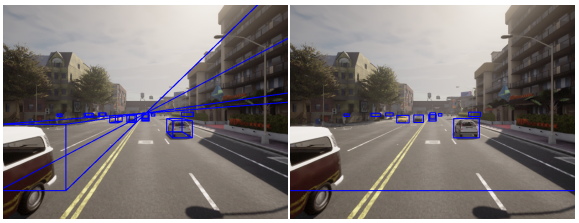

Figure 5: The negative notation effects on the bounding boxes for the objects. For 3D bounding box, the box appears as a beam (left). For 2D bounding box (see the right image), the box should have included the vehicle (in the corner of the image at the right) inside, but instead, it is covering a significant portion of the entire image.

5 METHODOLOGIES

5.1 Objects in Carla Simulator

Carla simulator includes four vehicle classes (car, truck, van, and bus), as well as vulnerable road users (motorcycles, bicycles, and pedestrians). We excluded motorcycles and bicycles from our work because their 2D bounding boxes appear as lines instead of boxes. For static objects, we included traffic lights in our work. Extending the work to include other static objects is possible with our algorithm. The classes we included in our work is (car, bus, truck, van, walker, and traffic light).

5.2 Sensors in Carla Simulator

- RGB Image Sensor. This sensor is installed in the ego vehicle in Carla simulator and the output as an RGB (Red, Green, and Blue) image, the parameters of this sensor in our work have the following FOV=90, iso=100, gamma=2.2, and image size= (x:1280, y:720).
- Semantic Segmentation Camera. We also installed Semantic segmentation Camera on the ego vehicle in the simulation. In the context of a camera, it refers to a computer vision technique used in image analysis and computer vision systems. It involves the process of classifying each pixel in an image into a specific category or class, such as identifying objects, regions, or areas in the image and labelling them accordingly. This technique is commonly used in various applications, including autonomous vehicles, surveillance, image editing, and medical imaging, among others. Applications of semantic segmentation with cameras include Autonomous Vehicles (Sistu et al., 2019) where Semantic segmentation is crucial for self-driving

cars to identify and understand their surroundings, including detecting pedestrians, other vehicles, road signs, and road boundaries. In Carla simulator we used automatic labelling of the object (Table 1) which plays a major role in our work, and we take the same parameters as the RGB camera sensor (ROV and Image Size) (Chresten et al., 2018).

- Radar. The RADAR sensor in the CARLA simulator is a placeholder model that is not based on raytracing. It casts rays to objects and computes distance and velocity using a simplified model. The sensor can detect obstacles in front of it and can be used for applications such as adaptive cruise control and obstacle detection. The output of radar is altitude, azimuth, depth, velocity.
- LiDAR. The CARLA LiDAR sensor is a 3D point cloud sensor that uses laser light to measure distance to objects in its surroundings.
- Depth Image. CARLA depth images are grayscale images where each pixel represents the distance to the object at that pixel. There are two kinds of depth image: colour image and gray scale image, which turned the distance sorted as an RGB channels into a [0,1] float. In our generated Data, we generated the Depth image as grayscale.

Note: The parameters in this sensors are editable where we could control in the parameters in the file (sensors.yaml) – see the package in our GitHub project, where we generated RGB images, Semantic segmentation images, radar data, lidar data, depth images, and filtered bounding box notations as an text file.

5.3 Correct the Coordinates of the Boxing Object

The 2D box notation is defined in two ways: the centre box and the corner box. The corner box is represented as (x min, y min, x max, y max), with the reference point located at the upper-left corner of the image. The numerical values within this notation correspond to the coordinates of the pixels at that specific point (Figure 6). In addition, the coordinates should be normalized (Albumentations, 2023) according to the following formula:

$$\begin{aligned} x_{min} &= x_{up\_left}/width \\ y_{min} &= y_{up\_left}/hight \\ x_{max} &= x_{down\_right}/width \\ y_{max} &= y_{down\_right}/hight \end{aligned} \tag{3}$$

Where the width and hight describe the dimension of image by pixels. The centre coordinates describe the centre, width and hight of the box (x centre, y centre, width, hight) (Figure 7). It's important to note that

Figure 6: The coordinates of the box of an object are shown. We defined the corner coordinates of the box on the car as (x min, y min, x max, y max) = (83, 410, 466, 640) respectively.

the minimum value for a coordinate is always zero. However, the maximum value varies depending on whether the coordinate has been normalized or not. In the case of normalized coordinates, the maximum value is set to one. On the other hand, when dealing with non-normalized coordinates, the upper limit is determined by the dimensions of the image pixel. It's worth mentioning that in the context of the coordinate system used in the Carla simulator, it's possible for a generated corner coordinate to fall below zero or exceed the dimensions of the image pixel. To correct this issue, we filtered the value of coordinates as follows:

Data: Normalized corner box (x centre, y centre, width, hight)
Result: Correct the coordinate of the box
if $x_{min} > 1$ **then**
  delete the box and stop;
end
$x_{min} = max(0, x_{min})$;
if $x_{max} < 0$ **then**
  delete the box and stop;
end
$x_{max} = min(x_{max}, 1)$;
Repeat same steps for y_{min} and y_{max};
if *area of the box = 0* **then**
  delete the box;
end

Algorithm 1: Correct the coordinate.

The correction of this issue of coordinates will resolve the incorrect bounding box value. However, it's important to note that this correction won't address the box covering a significant portion of the image (Figure 4). We will delve into how we addressed this challenge and how we did solve it later in this paper.

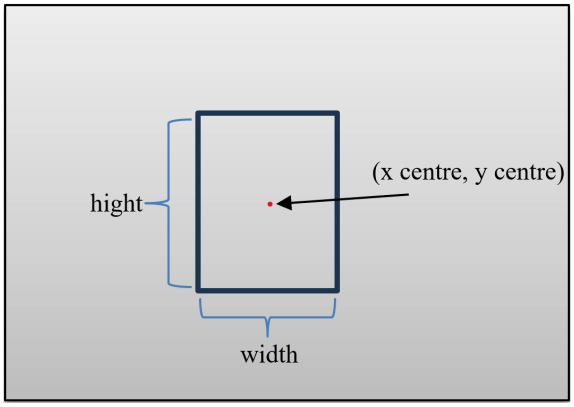

Figure 7: Centre coordinates for bonding box objects.

5.4 Filter Unwanted Boxes

As discussed earlier, another issue related to bounding boxes in Carla simulator is the presence of "ghost boxes." These are boxes that are occluded by other objects (and not visible in the simulation) but they are still labelled in Carla (Figure 1). To address this challenge, we utilize semantic segmentation in the Carla simulator to filter out unwanted boxes. In the Carla simulator, there are 28 distinct colours associated with various objects (as shown in Table 1). Our approach involves systematically examining each bounding box using semantic segmentation cameras. We verify whether the colour within the box corresponds to the expected object colour. For instance, if the box represents a car, we check if there are pixels with the colour (0,0,142) inside the box, aligning with the designated area in the image segmentation camera. To enhance the precision of the bounding box, we implemented a filtering criterion to refine object detection. Specifically, we retained bounding boxes that occupied at least 10% of the total area. In other words, if at least 10% of the bounding box of the car area contained pixels with a value of (0, 0, 142) (Table 1), the box was preserved, otherwise, it was deleted (Figure 8).

The threshold of 10% is a flexible value in our work. We can change the boundingBox.yaml file in the GitHub project by changing the threshold_small_box to the appropriate present. This allows us to adjust the size of the bounding boxes that are detecting. For example, if we want to detect smaller objects, we can lower the threshold value. The threshold_small_box parameter is located in the boundingBox.yaml file. We can change the value of this parameter by editing the file and saving it. The new value will be used the next time we run the object detection code.

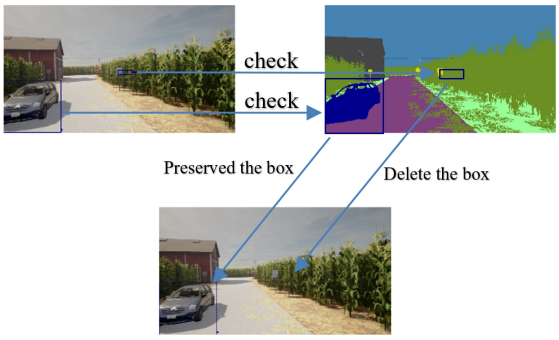

Figure 8: In this filter, we check each box in image segmentation camera if it fulfils 10% of the colour corresponding to the object in image segmentation. In our example the colour of the car in image segmentation in Carla is (0, 0, 142), the small box behind the field is hidden and there is no pixel with value (0, 0, 142) inside of it, as compared to the bounding box for the car in the large box.

Here is an example of how to change the threshold_small_box parameter:

```
threshold_small_box: 5
```

This will change the threshold value to 5%. This means that only bounding boxes, that are at least 5% of the area, which contains the colour corresponding to this object on image segmentation, will be detected. This procedure allows us to check whether the object exists at each pixel of the bounding box, so we can detect the object even if the box has only one pixel.

5.5 Filter the Large Boxes

The previous method (Section 5.4) suffers one limitation in the case of large bounding boxes. This is because a bounding box that identifies an object may contain other objects, if the bounding box is large enough to cover them. Therefore, if we use a 10% match threshold, then the bounding box may be incorrectly retained, even if it does not accurately identify the object (Figure 9).

To solve this challenge, we defined two thresholds for the large boxes.

- The ratio of the bounding box area to the image area.
- A special threshold for filtering large bounding boxes.

In our work, we used a criterion of 70% to determine whether a bounding box is large. In other words, if the ratio of the bounding box area to the image area is greater than 70%, then we considered the box to be large. Otherwise, the box was considered to be normal or small. The special threshold that we used in our GitHub project to filter the boxes is 50%. This

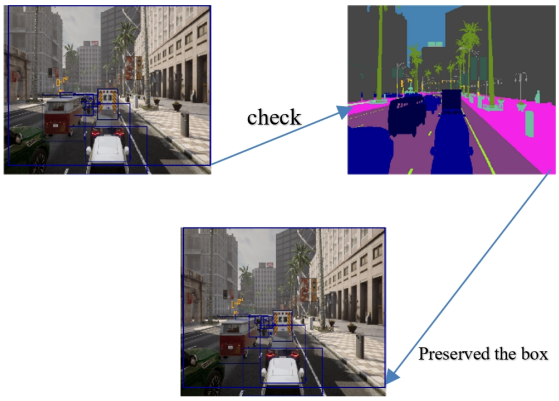

Figure 9: The large box that covers a significant portion of the image and contains multiple cars that exceed the threshold (10%) of the area of the box, is a ghost box for a car. This leads to preserving the bounding box, even though it does not accurately identify a car.

means that if a large bounding box contains more than 50% of pixels in the image segmentation that belong to the same class as the detected object, then the box is preserved. Otherwise, the box is deleted. This algorithm describes the filtering of the box:

Data: Normalized corner box (x centre, y centre, width, hight)
Result: Filter the ghost boxes.
initialization;
Count the pixels = 0;
for *all the pixels in the box in image segmentation* **do**
  if *the pixel is the same class of the box detected* **then**
    Count the pixels +=1;
  end
end
if *the box is large* **then**
  if *the ratio of Count the pixels to the number of pixels of the box is less than the large box threshold filter* **then**
    delete the box;
  end
else
  if *the ratio of Count the pixels to the number of pixels of the box is less than the normal box threshold filter* **then**
    delete the box;
  end
end

Algorithm 2: Filter the boxes.

5.6 Filter 3D Bounding Box

In our work, we propose a novel filtering approach for 3D bounding boxes by converting the boxes to 2D bounding boxes and applying a filtering criterion to the 2D representations. This approach enables efficient and effective filtering of 3D bounding boxes (Figure 10). Filtering 3D bounding boxes by identifying ghost boxes, which are bounding boxes that do not correspond to real objects. This is a crucial step in object detection and tracking tasks. In our work, we propose a novel filtering method that leverages the concept of 2D bounding boxes to efficiently identify and remove ghost boxes from 3D bounding box sets. By converting 3D bounding boxes to their corresponding 2D representations, we can effectively check for ghost boxes based on 2D geometric criteria. This approach significantly improves the accuracy and efficiency of object detection and tracking tasks, particularly in scenes with complex backgrounds or occlusions. In the GitHub we implemented an example to filter the 3D bounding box in the file name 3D_Bounding_Box.py.

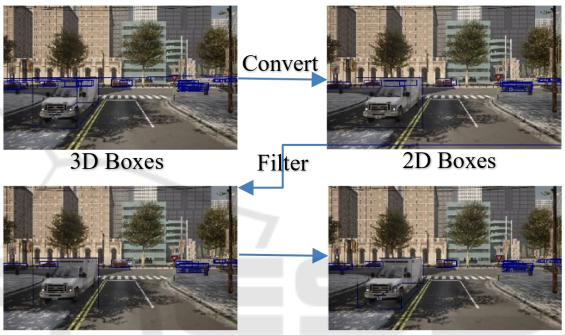

Figure 10: Filtering the 3D bounding box by check if the box is ghost box or not in 2D bounding box.

6 OUR DATA VALIDATION

To create a diverse and comprehensive dataset for object detection and localization, we employed the Carla simulator 9.14 (Dosovitskiy et al., 2017) and utilized eight distinct maps (5000 images for each map). Our filtering approach yielded a dataset encompassing six object classes: car, bus, truck, van, walker, and traffic light (see section 5.4). To enhance the dataset's variety, we incorporated data from five sensors: RGB camera, semantic segmentation camera, depth camera, radar sensor, and LIDAR sensor. For all objects within a range of 50 meters, 100 meters, 150 meters, and 200 meters (Figure 11), we applied our bounding box filter to the RGB images (1280,720). To standardize our results, we trained our dataset using a pretrained YOLOv8 model (Jocher et al., 2023) for bounding boxes with all objects up to 100 meters. The training parameters included SGD (lr=0.01, momentum=0.9), image size for training 640, epoch 50, and iou 0.7 Using 20% of images for validation (i.e. 1000

images for each map) we got a result for data labelled up to 100 meter as the in Table 2, and results for the same data labelled up to 50 m (Table 3).

Figure 11: Labelling of bounding boxes up to 50m (left) and up to 100 m (right).

Table 2: A summary of results from the model which is trained using our filtered data in base of bounding box, which is labeling all objects up to 100 meters, where the image size for training has 640x640 pixels.

Class	Precision	Recall	mAP50
All	0.915	0.664	0.75
Car	0.895	0.788	0.859
Bus	0.925	0.949	0.961
Truck	0.909	0.433	0.536
Van	0.874	0.795	0.865
Walker	0.914	0.485	0.585
Traffic light	0.972	0.535	0.696

Table 3: A summary of results from the model which is trained using our filtered data in base of bounding box, which is labeling all objects up to 50 meters, where the image size for training has 640x640 pixels.

Class	Precision	Recall	mAP50
All	0.947	0.805	0.882
Car	0.941	0.908	0.964
Bus	0.917	0.981	0.983
Truck	0.965	0.524	0.639
Van	0.927	0.918	0.965
Walker	0.954	0.757	0.865
Traffic light	0.977	0.739	0.876

Table 4: A summary of results from the model which is trained using our filtered data in base of bounding box, which is labeling all objects up to 100 meters, where the image size for training has 1280x1280 pixels.

Class	Precision	Recall	mAP50
All	0.939	0.787	0.876
Car	0.927	0.852	0.919
Bus	0.918	0.953	0.977
Truck	0.969	0.615	0.787
Van	0.906	0.866	0.924
Walker	0.944	0.687	0.8
Traffic light	0.971	0.75	0.851

Table 5: A summary of results from the model which is trained using our filtered data in base of bounding box, which is labeling all objects up to 50 meters, where the image size for training has 1280x1280 pixels.

Class	Precision	Recall	mAP50
All	0.955	0.908	0.966
Car	0.962	0.932	0.979
Bus	0.916	0.982	0.989
Truck	0.948	0.734	0.909
Van	0.949	0.967	0.987
Walker	0.975	0.895	0.961
Traffic light	0.98	0.935	0.97

Form the previous two tables, we can conclude that the accuracy of object detection decreases as the distance between the object and the camera increases. This is because smaller objects are more difficult to detect due to their lower resolution. This effect is particularly pronounced when annotating objects for distant scenes. To improve the accuracy of object detection for distant objects, it is important to use high-resolution images and to train the object detection model on a dataset that includes a large number of distant objects.

As evident in Table 4 and Table 5, training our data using an image size of 1280 resulted in enhanced object detection accuracy, particularly for objects located at greater distances. This improvement can be attributed to the increased resolution of the images, which allows the object detection model to capture finer details and distinguish objects more effectively, especially when dealing with smaller objects at a distance. Interestingly, we observed that traffic lights, despite being small objects, achieved high detection accuracy. This can be attributed to the distinctive shape of traffic lights, which resembles traffic light poles (Figure 12). The object detection model is likely to learn these distinctive features, enabling it to accurately identify traffic lights even when they are small or hight distant.

Figure 12: Sample of our data where we see that the traffic light has figures such as a traffic light poles.

7 CONCLUSIONS

The Carla simulator is an effective tool for generating datasets for object detection tasks. Its flexibility and controllability over the environment and artificial scenarios, such as accidents, congestion, and severe weather conditions, make it a valuable tool for creating realistic and challenging datasets. Our proposed filter has been shown to be highly effective in improving the accuracy of object detection models trained by Carla datasets generated using our filter. We believe that our work represents a significant step forward in the development of high-quality datasets for object detection tasks. In addition, data generation through the CARLA simulator has become more reliable. The project we developed to create bounding boxes is now fully available on GitHub. In addition, we have made it more flexible to select parameters in CARLA through YAML files. This allows for the generation of data related to weather conditions, such as fog, rain, and the number of cars, as well as the ability to control car lights. This flexibility makes it easy to develop self-driving cars in severe weather conditions in CARLA. In addition, we have included many sensors, such as radar, lidar, and depth image, which allow for the integration and synchronization of multiple sensors and the use of the bounding boxes that we developed. Using YOLOv5 and YOLOv8, we achieved good results and high accuracy in the data that was collected using our algorithm to filter bounding boxes. The accuracy exceeded 90%. This confirms the success of the filter we developed in generating data through the CARLA simulation program.

REFERENCES

Albumentations (2023). Bounding boxes. Boundingboxe saugmentationforobjectdetection. [Online; accessed 28-December-2023].

Burdorf, S., Plum, K., and Hasenklever, D. (2022). Reducing the amount of real world data for object detector training with synthetic data. *arXiv preprint arXiv:2202.00632*.

Chaar, M. M., Weidl, G., and Raiyn, J. (2023). Analyse the effect of fog on the perception. In *10th International Symposium on Transportation Data & Modelling (ISTDM2023)*, page 332.

Chresten, L., Lund-Hansen, Juul, T., Eskildsen, T. D., Hawes, I., Sorrell, B., Melvad, C., and Hancke, K. (2018). A low-cost remotely operated vehicle (rov) with an optical positioning system for under-ice measurements and sampling. *Cold Regions Science and Technology*, 151:148–155.

Correll, N., Hayes, B., Heckman, C., and Roncone, A. (2022). *Introduction to autonomous robots: mechanisms, sensors, actuators, and algorithms*. Mit Press.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Jocher, G., Stoken, A., Borovec, J., Changyu, L., Hogan, A., Diaconu, L., Poznanski, J., Yu, L., Rai, P., Ferriday, R., et al. (2023). Yolov8 docs. https://docs.ultralytics.com/#where-to-start. [Online; accessed 28-December-2023].

Mukhlas, A. (2020). Carla 2d bounding box annotation module. https://mukhlasadib.github.io/CARLA-2DBBox. [Online; accessed 28-December-2023].

Muller, R. (2022). Drivetruth: Automated autonomous driving dataset generation for security applications. In *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*.

Niranjan, D., VinayKarthik, B., et al. (2021). 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.

Pitropov, M., Garcia, D. E., Rebello, J., Smart, M., Wang, C., Czarnecki, K., and Waslander, S. (2021). Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690.

Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al. (2020). Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE.

Sistu, G., Chennupati, S., and Yogamani, S. (2019). Multi-stream cnn based video semantic segmentation for automated driving. *arXiv preprint arXiv:1901.02511*.

Team (2020). Bounding boxes. https://carla.readthedocs.io/en/latest/tuto_G_bounding_boxes/. [Online; accessed 28-December-2023].