IJAER/ March-April 2024/Volume-13/Issue-2 ISSN: 2278-9677
International Journal of Arts & Education Research
(Peer-Reviewed, Open Access, Fully Refereed International Journal) Impact Factor:7.06

Object Detection for Simulating Vehicle Driving using CARLA Simulator

1st Sakshi Nadarge
dept. Electronics and Telecommunication
International Institute of Information Technology
Pune, India

2nd Anushree Patil
dept. Electronics and Telecommunication
International Institute of Information Technology
Pune, India

3rd Prof. Prashant Ahire
Dept. Electronics and Telecommunication
International Institute of Information Technology Pune, India

Abstract—

Nowadays, autonomous vehicles are gaining traction due to their numerous potential applications in resolving a variety of other real-world challenges. AV validation on real scenarios involving actual objects such as cars or pedestrians in a wide range of traffic cases would escalate the cost and could generate hazardous situations. Autonomous vehicle technology has witnessed significant advancements in recent years, driven by the need for safer, more efficient transportation solutions. As the development of autonomous systems necessitates rigorous testing and validation, simulation platforms have become indispensable tools. Among these, the CARLA (Car Learning to Act) simulator has emerged as a leading open-source solution, offering a realistic and customized virtual environment for autonomous driving research. CARLA open-source AV simulator is designed to be able to train and validate control and perception algorithms in complex traffic scenarios with hyper-realistic environments. CARLA simulator allows to easily modify on-board sensors such as cameras or LiDAR, weather conditions and also the traffic scene to perform specific traffic cases.

Index Terms—Object Detection, CARLA (Car Learning To, Machine Learning, Deep Learning, Autonomous Vehicles etc.

I. INTRODUCTION

The pursuit of autonomous vehicle technology has revolutionized the landscape of transportation, promising safer and more efficient roads through vehicles capable of navigating and making decisions without human intervention. The development of autonomous systems demands meticulous testing and validation, a process that is inherently complex and often impractical to conduct solely in the physical world. As a result, simulation platforms have become instrumental in advancing autonomous vehicle research, providing a controlled and adaptable environment for testing algorithms, training machine learning models, and validating control strategies.

The increasing autonomy of vehicles necessitates a paradigm shift in testing methodologies. Traditional real-world testing is resource-intensive, time-consuming, and often constrained by safety concerns. Simulation offers a viable alternative, allowing researchers and developers to iterate and experiment rapidly in a controlled virtual environment. CARLA, as an open-source and extensible simulator, provides a valuable platform to address these challenges.

Autonomous driving research is always hampered; by the infrastructure costs, high-performance systems, sensors, communication devices, the logistical difficulties of training and testing the systems, and jeopardizing the safety of people in the real world. In addition to this, equipping and operating a single autonomous vehicle needs a significant budget and manpower. However, this single vehicle is far away enough from collecting the data, pre-processing, training, testing, and validating it. Therefore, one of the ideal solutions to real-time testing is to leverage simulation. The simulation democratizes the autonomous urban driving research and plays a significant role in training and validating the autonomous driving solution approaches.

There are plenty of simulators to test the autonomous vehicle solutions having their pros and cons, such as CarCraft, Udacity, TORCS, and RRADS, etc. However, these simulators lack the dynamics and complexity of environment such as pedestrians, intersections, traffic rules, and other complex dynamics that distinguish urban driving from straightforward track racing. Moreover, some closed-source commercial simulators such as Grand Theft AutoV, PreScan, and ANSYS provides little customization and control over the environment, limited kinematic behavior, restricted scripting and use case scenarios specifications, limited sensor suite specification, and several other impediments due to commercial nature of the simulators.

An ideal simulator should be as realistic as possible. This is to say that it should be detailed in terms of the 3D environment and must be precise with the lower-level vehicle calculations such as the physics of the vehicle. There is always a trade-off between the fidelity of the 3D environment and the simplified vehicular dynamics. CARLA meet this trade-off, making them state-of-the-art simulator. CARLA provides a high-fidelity simulation environment that closely replicates real-world driving conditions. It offers realistic graphics, physics, and sensor models, allowing developers to create immersive and accurate simulations. CARLA is an open-source platform, which means developers can access and modify the source code to suit their specific needs. This flexibility allows for customization and integration of new features, algorithms, and sensor models.

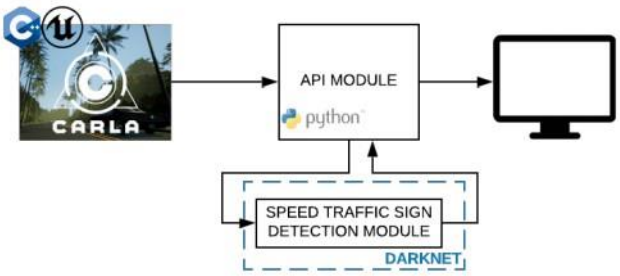

Fig. 1. Project's Pipeline

II. CHALLENGES IN REAL-WORLD VALIDATION AND TESTING OF AUTONOMOUS DRIVING

A. Safety Concerns

Autonomous driving systems must operate safely in complex and dynamic environments. Validating their safety in real-world scenarios involves exposing them to various unpredictable situations, such as adverse weather conditions, construction zones, and unexpected pedestrian or cyclist behavior. Ensuring the safety of autonomous vehicles and their ability to handle these scenarios is a significant challenge.

B. Scale and Diversity of Testing

Validating autonomous driving systems requires extensive testing across a wide range of scenarios, including different road types, traffic conditions, and geographical locations. Achieving comprehensive coverage of all possible scenarios is a daunting task, as it would require an enormous amount of time, resources, and physical test vehicles.

C. Safety Concerns

Cost and Time Constraints: Real-world testing is expensive and time-consuming. It involves deploying test vehicles, collecting data, and analyzing the results. The cost of acquiring and maintaining a fleet of test vehicles, along with the expenses associated

with insurance, permits, and personnel, can be prohibitive. Additionally, conducting tests on public roads may require obtaining regulatory approvals, further adding to the time and cost constraints.

D. Replicating Rare and Dangerous Events

Autonomous driving systems must be capable of handling rare and dangerous events, such as accidents or extreme weather conditions. Replicating these events in real-world testing is challenging due to their infrequency and potential risks. Ensuring that autonomous vehicles can respond appropriately to such events is crucial for their safe deployment.

E. Data Collection and Analysis

Real-world testing generates vast amounts of data that need to be collected, stored, and analyzed. Extracting meaningful insights from this data and using it to improve the performance and safety of autonomous driving systems is a significant challenge. Developing efficient data collection and analysis methods is crucial for effective real-world validation.

III. ARCHITECTURE OF CARLA SIMULATOR

The architecture of the autonomous vehicle is composed of four major layers, as illustrated in Figure 2, that are sensor layer, perception layer, planning layer and control layer.

A. Sensor Layer

The sensor layer in the architecture of CARLA is responsible for simulating the various sensors that an autonomous vehicle typically uses to perceive its environment. These sensors provide crucial data for the perception module, allowing the vehicle to understand the surrounding world. CARLA simulates camera sensors that capture visual information. These cameras can be configured with different parameters, such as field of view, resolution, and position on the vehicle. Visual data is essential for tasks like object detection, lane keeping, and scene understanding. LiDAR sensors simulate laser-based devices that measure distances to objects in the environment. CARLA allows users to configure LiDAR sensors with different settings, such as the number of beams, rotation speed, and range. LiDAR data is valuable for creating detailed 3D maps of the surroundings and detecting obstacles. RADAR sensors are simulated to capture radio waves reflected off objects in the environment. CARLA provides radar sensors with adjustable parameters like range and field of view. RADAR data is useful for detecting objects and estimating their velocities, particularly in adverse weather conditions. The GPS sensor simulates the global positioning system, providing information about the vehicle's geographic coordinates. GPS data is essential for localization, helping the autonomous system understand its position within the simulated environment.

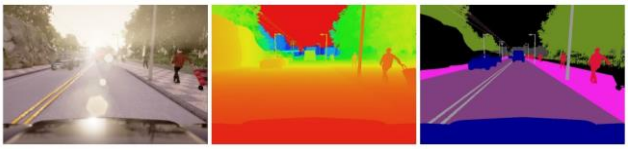

Fig. 2. : Three of the sensing modalities provided by CARLA. From left to right: normal vision camera, ground-truth depth, and ground-truth semantic segmentation. Depth and semantic segmentation are pseudo-sensors that support experiments that control for the role of perception. Additional sensor models can be plugged in via the API.

B. Perception Layer

The perception layer in the architecture of CARLA is responsible for interpreting and understanding the data collected from simulated sensors. It processes the information captured by cameras, LiDAR, RADAR, and other sensors to extract relevant details about the environment. The goal is to generate a comprehensive representation of the surroundings, which can then be used by higher-level modules, such as planning and control, for decision-making and action execution.

Object detection algorithms in the perception layer analyze sensor data to identify and locate objects in the environment. This includes recognizing vehicles, pedestrians, cyclists, and other relevant entities. Object detection is fundamental for understanding

the dynamics of the scene and predicting potential interactions. Lane detection algorithms process visual sensor data to identify lane markings on the road. This information is crucial for determining the vehicle's position within the lane and aiding in tasks such as lane-keeping and lane-changing. Perception algorithms are designed to recognize and interpret traffic signs and signals. This includes identifying stop signs, traffic lights, speed limit signs, and other regulatory signs. Recognition of these elements is vital for obeying traffic rules and ensuring safe navigation.

C. Planning Layer

The planning layer in the architecture of CARLA is responsible for generating high-level plans and trajectories for the autonomous vehicle based on the information provided by the perception layer. This layer focuses on decision-making, determining the optimal course of action for the vehicle to navigate its environment safely and efficiently.

The planning layer often begins with route planning, where the system decides the overall path the vehicle should take to reach its destination. This involves considering factors such as map information, user-defined way-points, and dynamic changes in the environment. Behavior planning involves making high-level decisions about the vehicle's actions. This includes determining whether the vehicle should change lanes, overtake another vehicle, make a turn at an intersection, or stop at a traffic light. The behavior planner considers the current situation and the desired destination. Path planning focuses on finding a collision-free path for the vehicle within the planned trajectory. This involves considering the vehicle's physical constraints, such as its dimensions and turning radius, while avoiding obstacles and maintaining a safe distance from other objects in the environment. Decision-making algorithms in the planning layer take into account various factors, including the current state of the vehicle, traffic conditions, legal requirements, and user-defined preferences. These algorithms aim to generate plans that balance safety, efficiency, and compliance with traffic rules.

D. Control Layer

The control layer in the architecture of CARLA is responsible for executing the plans and trajectories generated by the planning layer. It translates the high-level commands into low-level control signals that directly manipulate the vehicle's actuators, such as throttle, brake, and steering. The control layer ensures that the vehicle physically follows the planned trajectory while considering real-time feedback from sensors. The control layer models the dynamics of the vehicle, including its acceleration, braking, and steering behavior. The output of the planning layer is a high-level plan or trajectory, specifying the desired path and behavior of the vehicle. The control layer translates this high-level plan into low-level control commands that directly influence the vehicle's motion. These commands typically include throttle, brake, and steering inputs. Throttle control involves adjusting the engine power to control the vehicle's speed. Brake control regulates the braking force applied to the vehicle. Steering control determines the steering angle necessary to follow the planned path.

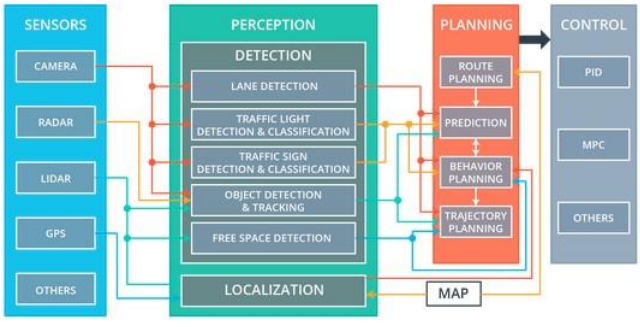

Fig. 3. Architecture of CARLA.

IV. OBJECT DETECTION

Object detection is a critical component in autonomous vehicle driving systems as it allows the vehicle to identify and locate various objects in its environment, such as pedestrians, vehicles, and obstacles. In CARLA, object detection is typically part of the perception layer, which processes data from simulated sensors to understand the surroundings.

You Only Look Once (YOLO) is an advanced approach for object detection. YOLO applies a single CNN to the entire image which further divides the image into grids. Prediction of bounding boxes and respective confidence score are calculated for each grid. These bounding boxes are analyzed by the predicted confidence score. The Architecture of YOLO has 24 convolutional layers and 2 fully connected layers.

The architecture of YOLO, is based on a deep Convolutional Neural Network (CNN). It follows the general principles of the YOLO algorithm, which is known for its real-time object detection capabilities. YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell.

The YOLO architecture typically consists of several convolutional layers followed by fully connected layers. These layers are responsible for extracting features from the input image and making predictions. The network architecture often includes skip connections, which allow information from earlier layers to be combined with later layers, aiding in the detection of objects at different scales.

One key aspect of YOLO is its multi-scale detection approach. It uses anchor boxes of different scales to handle objects at various sizes and aspect ratios. This allows YOLO to detect and localize objects of different scales within the same image. By considering multiple scales, YOLO can handle both small and large objects effectively.

The output of YOLO consists of bounding box predictions and class probabilities. For each grid cell, YOLO predicts multiple bounding boxes, each associated with a specific class probability. The bounding boxes are represented by their coordinates (x, y, width, height) relative to the input image. The class probabilities indicate the likelihood of each bounding box belonging to a particular object class.

During training, YOLO requires a labeled dataset with annotated bounding boxes and class labels. The network is trained using a combination of classification and regression loss functions. The classification loss measures the accuracy of the predicted class probabilities, while the regression loss measures the accuracy of the predicted bounding box coordinates.

Post-processing techniques are applied after object detection to refine the results. Non-maximum suppression (NMS) is commonly used to remove duplicate or overlapping bounding boxes, ensuring that only the most confident and non-overlapping detections are retained.

Fig. 4. Object Detection

A. Training

Training a dataset in YOLO for object detection involves several steps that are crucial for achieving accurate and reliable results. The first step is dataset preparation, where you collect a set of images or videos and annotate them with bounding box annotations

for the objects you want to detect. Each object should be annotated with a bounding box and assigned a corresponding class label. It's important to ensure the annotations are accurate and comprehensive.

Once the dataset is prepared, the next step is to configure the YOLO model. This involves setting up a configuration file that specifies various parameters such as the network architecture, input image size, number of classes, anchor box sizes, and other hyperparameters. The configuration file needs to be adjusted according to the specific requirements and characteristics of your dataset.

After configuring the model, you need to prepare the pre-trained weights. Download a pre-trained model, typically trained on a large-scale dataset like ImageNet, and load the weights into the YOLO model. This initialization helps the model to learn from the pre-existing knowledge and speeds up the training process.

The training phase involves optimizing the YOLO model using the annotated dataset. The model adjusts its weights through back propagation and gradient descent optimization algorithms to minimize the loss between predicted bounding boxes and ground truth annotations. It's important to split the dataset into training and validation sets to monitor the model's performance during training. Regularly evaluate the model's performance on the validation set and adjust training parameters if necessary.

Post-processing techniques are applied after training to refine the model's predictions. Non-maximum suppression (NMS) is commonly used to remove duplicate or overlapping bounding boxes, ensuring only the most confident and non-overlapping detections are retained. Applying confidence thresholds can also filter out low-confidence detections.

To evaluate the trained model's performance, use a separate test set and calculate metrics such as precision, recall, and mean average precision (mAP). These metrics provide insights into the model's accuracy and generalization capabilities.

If the model's performance is not satisfactory, fine-tuning becomes necessary. This involves adjusting hyper-parameters, increasing the training data, or using data augmentation techniques to improve the model's performance. Iterate through the training process, making adjustments and retraining the model until the desired performance is achieved. It's important to note that the specific implementation and tools used for training YOLO may vary depending on the framework or library you are using. It's recommended to refer to the documentation or tutorials specific to your chosen implementation for detailed instructions on training a dataset in YOLO for object detection.

Fig. 5. Training Dataset

V. Experimental Results

The experimental results of our paper on object detection for simulating vehicle driving using the CARLA simulator demonstrate the efficacy of our proposed approach. In our experimental setup, we utilized the CARLA simulator, a powerful tool for simulating real-world driving scenarios with high fidelity. Our objective was to develop and evaluate an object detection model capable of accurately detecting various objects in the simulated environment, including vehicles, pedestrians, and traffic signs. The objective of this research was to enhance the realism and accuracy of simulated driving scenarios, ultimately contributing to the development and evaluation of advanced driver assistance systems (ADAS) and autonomous driving technologies.

To train our object detection model, we employed a dataset comprising annotated images generated from the CARLA simulator. This dataset allowed us to leverage synthetic data, providing ample diversity in environmental conditions and object appearances. We adopted the YOLO (You Only Look Once) architecture due to its real-time processing capabilities and well-established performance in object detection tasks. We fine-tuned the YOLO model on our dataset, adjusting hyperparameters and incorporating data augmentation techniques to enhance its generalization ability.

In conclusion, our experimental results underscore the viability of utilizing object detection techniques for simulating vehicle driving in the CARLA simulator. By achieving high levels of accuracy and robustness, our approach contributes to the ongoing efforts aimed at enhancing the realism and effectiveness of autonomous driving simulations.

Prior studies have explored various aspects of CARLA simulator, such as environment modeling, vehicle dynamics, and sensor simulation.

YOLO is a single-stage object detector with three important parts in its architecture: the backbone, neck, and head. The backbone is responsible for the extraction of features from the given input images, the neck mainly generates the feature pyramids, and the head performs the final detection as an output

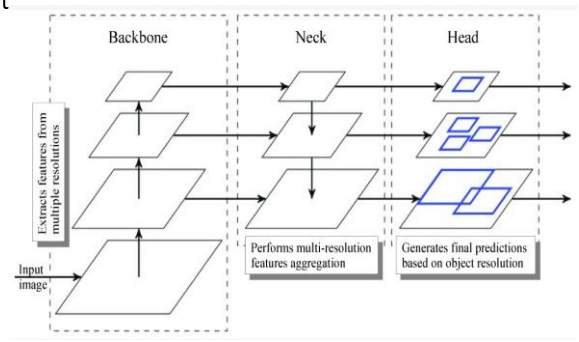

Fig. 6. Experimental Result

VI. Related Work

These works provide foundational knowledge and techniques for integrating advanced functionalities like object detection and collision avoidance.Research on object detection algorithms, including YOLO (You Only Look Once) variants like YOLOv7, has been extensively conducted. Understanding the strengths and limitations of these algorithms is crucial for implementing efficient and accurate object detection within the simulation environment.Some research endeavors have explored similar integrations within different simulators or environments. These studies provide valuable insights into the challenges and strategies involved in combining object detection with collision avoidance for simulated vehicle driving.Research on transfer learning techniques can be relevant for this project, particularly for bridging the gap between simulation and real-world deployment. By leveraging transfer learning, models trained in simulation environments can be fine-tuned to perform effectively in real-world scenarios.

VII. Conclusion

CARLA simulator marks a significant milestone in advancing the capabilities of autonomous vehicle simulation. This fusion of cutting-edge technologies enables more accurate perception and proactive decision-making, fostering safer and more realistic driving scenarios.

With YOLOv7, the simulator can efficiently detect and classify objects in the environment with remarkable speed and accuracy. This real-time detection capability enhances the vehicle's awareness of its surroundings, enabling it to respond promptly to dynamic changes in the environment, such as the presence of pedestrians, vehicles, or obstacles.

Furthermore, the incorporation of collision avoidance mechanisms adds an extra layer of safety by enabling the simulated vehicle to anticipate and mitigate potential collisions. By continuously analyzing the environment and predicting potential hazards, the vehicle can navigate complex scenarios with greater confidence and reliability.

VIII. FUTURE WORK

In the realm of simulating vehicle driving using CARLA simulator with object detection using YOLOv7 and collision avoidance, there exist several promising avenues for future exploration and enhancement. One direction for future work involves delving deeper into the realm of object detection algorithms. While YOLOv7 provides impressive real-time performance and accuracy, continued research into more advanced algorithms could further improve detection capabilities, particularly in complex scenarios with occlusions or varying lighting conditions. Exploring novel architectures or integrating multiple detection models for different object classes could potentially enhance the overall perception system within the simulator.

Additionally, future efforts could focus on advancing collision avoidance strategies within the simulated environment. While the current implementation employs proactive measures to mitigate potential collisions, further research could explore predictive modeling techniques to anticipate the behavior of other vehicles and pedestrians. By incorporating predictive capabilities into collision avoidance algorithms, simulated vehicles could make more informed decisions in dynamic and unpredictable scenarios, ultimately enhancing safety and adaptability.

REFERENCES

[1] Sreenivas, K.; Kamakshiprasad, V. Improved image tamper localisation using chaotic maps and self-recovery. J. Vis. Commun. Image Represent. 2017, 49, 164–176.

[2] Ali, A.; Hassan, A.; Ali, A.R.; Khan, H.U.; Kazmi, W.; Zaheer, A. Real-time vehicle distance estimation using single view geometry. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Snowmass, CO, USA, 1–5 March 2020; pp.

1111–1120.

[3] Ding, M.; Zhang, Z.; Jiang, X.; Cao, Y. Vision-based distance measurement in advanced driving assistance systems. Appl. Sci. 2020, 10, 7276.

[4] Lim, Y.-C.; Lee, C.-H.; Kwon, S.; Jung, W.-Y. Distance estimation algorithm for both long and short ranges based on stereo vision system. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 841–846.

[5] Liu, L.-C.; Fang, C.-Y.; Chen, S.-W. A novel distance estimation method leading a forward collision avoidance assist system for vehicles on highways. IEEE Trans. Intell. Transp. Syst. 2016, 18, 937–949.

[6] Hane, C.; Sattler, T.; Pollefeys, M. Obstacle detection for self-driving¨ cars using only monocular cameras and wheel odometry. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5101–5108.

[7] Tram, V.T.B.; Yoo, M. Vehicle-to-vehicle distance estimation using a low-resolution camera based on visible light communications. IEEE Access 2018, 6, 4521–4527.

[8] Kim, G.; Cho, J.-S. Vision-based vehicle detection and inter-vehicle distance estimation. In Proceedings of the 2012 12th International Conference on Control, Automation and Systems, Jeju, Republic of Korea, 17–21 October 2012; pp. 625–629.

[9] Min, K.; Han, S.; Lee, D.; Choi, D.; Sung, K.; Choi, J. SAE Level 3 Autonomous Driving Technology of the ETRI. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 16–18 October 2019; pp. 464–466.

[10] Sanil, N.; Venkat, P.A.N.; Rakesh, V.; Mallapur, R.; Ahmed, M.R. Deep Learning Techniques for Obstacle Detection and Avoidance in Driverless Cars. In Proceedings of the 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), Amaravati, India, 10–12 January 2020.

[11] Barea, R.; Perez, C.; Bergasa, L.M.; Lopez-Guillen, E.; Romera, E.; Molinos, E.; Ocana, M.; Lopez, J. Vehicle Detection and Localization

Using 3D LIDAR Point Cloud and Image Semantic Segmentation. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 3481–3486.

[12] Perez-Gil,´ O.; Barea, R.; L´ opez-Guill´ en, E.; Bergasa, L.M.; Revenga,´ P.A.; Gutierrez, R.; D´ ´ıaz, A. DQN-Based Deep Reinforcement Learning for Autonomous Driving. In Proceedings of the Advances in Physical Agents II, Alcala de Henares, Spain, 19–20 November 2020; Springer´ International Publishing: Cham, Switzerland, 2021; pp. 60–67.

[13] Marti, E.; de Miguel, M.A.; Garcia, F.; Perez, J. A Review of Sensor Technologies for Perception in Automated Driving. IEEE Intell. Transp. Syst. Mag. 2019, 11, 94–108.

[14] Serban, A.C.; Poll, E.; Visser, J. A Standard Driven Software Architecture for Fully Autonomous Vehicles. In Proceedings of the 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), Seattle, WA, USA, 30 April–4 May 2018; pp. 120–127.

[15] Dworak, D.; Ciepiela, F.; Derbisz, J.; Izzat, I.; Komorkiewicz, M.; Wojcik, M. Performance of LiDAR Object Detection Deep Learning´Architectures Based on Artificially Generated Point Cloud Data from CARLA Simulator. In Proceedings of the 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 26–29 August 2019; pp. 600–605.

[16] Mauri, A.; Khemmar, R.; Decoux, B.; Haddad, M.; Boutteau, R. Real-time 3D multi-object detection and localization based on deep learning for road and railway smart mobility. J. Imaging 2021, 7, 145.

[17] Zhu, J.; Fang, Y. Learning object-specific distance from a monocular image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3839–3848.

[18] Reddy, N.D.; Vo, M.; Narasimhan, S.G. Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 1906–1915.

[19] Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.

[20] Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2778–2788.

[21] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 2015, 28.

[22] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.

[23] Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv 2013, arXiv:1312.6229.

[24] Li, B.; Zhang, T.; Xia, T. Vehicle detection from 3d lidar using fully convolutional network. arXiv 2016, arXiv:1608.07916.

[25] Li, B. 3d fully convolutional network for vehicle detection in point cloud. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1513–1518.

[26] Zhang, S.; Benenson, R.; Omran, M.; Hosang, J.; Schiele, B. Towards reaching human performance in pedestrian detection. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 40, 973–986.

[27] Luo, H.; Yang, Y.; Tong, B.; Wu, F.; Fan, B. Traffic sign recognition using a multi-task convolutional neural network. IEEE Trans. Intell. Transp. Syst. 2017, 19, 1100–1111.

[28] Behrendt, K.; Novak, L.; Botros, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1370–1377.

[29] Weber, M.; Wolf, P.; Zollner,¨ J.M. DeepTLR: A single deep convolutional network for detection and classification of traffic lights. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 19–22 June 2016; pp. 342–348.