Towards LiDAR and RADAR Fusion for Object Detection and Multi-Object Tracking in CARLA Simulator

Santiago Montiel-Marín, Carlos Gómez-Huélamo, Javier de la Peña, Miguel Antunes, Elena López-Guillén, and Luis M. Bergasa

Electronics Department, Universidad de Alcalá, Spain,
{santiago.montiel, j.pena, miguel.antunes}@edu.uah.es,
{carlos.gomezh, elena.lopezg, luism.bergasa}@uah.es

Abstract. Detection and Multi-Object Tracking (DAMOT) systems have a critical role to play in scene understanding in the context of autonomous driving. Modern Autonomous Driving Stacks (ADS) require a software processing unit or module that allows them to understand the data in the environment and convert it into vital information for further decision making. In this context, this work develops a DAMOT module based on Machine Learning techniques, such as DBSCAN or BEV-SORT, that receives information from LiDAR and RADAR sensors in CARLA Simulator. This module uses containerisation techniques with Docker and standard robotics communications with ROS. The performance of the method is evaluated in terms of detection in the AD PerDevKit dataset, developed by the authors.

Keywords: LiDAR, RADAR, object detection, multi-object tracking, CARLA Simulator.

1 Introduction

In recent years, academia and industry have both shown great interest in the development of autonomous vehicles (AV) and self-driving cars as they are seen as the main characters of a booming sector that marks the next transportation revolution. An AV is an intelligent transportation system capable of performing the functions related to the five domains of an AV architecture: perception, localisation, mapping-planning, decision making and control.

Faced with this situation, two types of architectures for solving this problem appear: modular [1] or end-to-end [2]. Moreover, the diversity of these architectures lies not only in the way they deal with the available data, but also in the sources of information they bring into play, the sensors. Typically, LiDAR, RADAR and monocular or stereo cameras are used to capture the instantaneous state of the environment surrounding the ego-vehicle and are accompanied by others such as odometry, Inertial measurement units (IMU), Differential Global Navigation Satellite System (D-GNSS) and High-definition Maps (HD Maps),

Fig. 1: **Performance of the DAMOT method** in a Traffic Jam use case in AD PerDevKit dataset. Tracked objects are shown as green Bounding Boxes.

which provide information on the state of the ego-vehicle and its control variables, as well as the environment itself [3].

This work is framed in the domain of perception in the context of a modular architecture, a layer in charge of understanding the scene and extracting information for decision making in the respective domains. In this work, we propose a work framed in the Object Detection and Multi-Object Tracking (DAMOT) tasks using Machine Learning techniques, achieving satisfactory results in challenging and complex test scenarios in CARLA Simulator [4].

Regarding the sensing of the environment, this work chooses to merge the information coming from a LiDAR and a RADAR synthetically generated in CARLA. Both sensors provide point clouds that describe the space in a three-dimensional way and while LiDAR is much more common in recent literature [5], it shows some shortcomings that RADAR can overcome, such as:

- It can take advantage of the Doppler effect to infer velocity while LiDAR is unable to natively infer it.
- It can excel in long range detections, as it typically performs better given the smaller gradient of information with respect to distance.
- Native velocity inference can also contribute to the Object Tracking task, since more variables are contributed to the model than are normally inferred or predicted by Bayesian models.
- It can also contribute to the categorical identification of an object by means of the Radar Cross Section (RCS) together with the Reflection Intensity of LiDAR. As it is not a feature included in CARLA, this aspect is postponed to future studies.

Therefore, this method will be implemented and validated over the novel AD PerDevKit dataset [6], in which the performance of the Object Detection method will be measured. This tool includes complex test scenarios recorded in

CARLA Simulator under diverse weather and environment conditions. Finally, the main contributions of this paper are:

- The development and integration of a DAMOT pipeline based on LiDAR and RADAR into a state-of-the-art autonomous driving architecture [3].
- An evaluation of an Object Detection system in our challenging and open-source AD PerDevKit dataset.

2 Related works

The following section will discuss different approaches taken by the community in the field of LiDAR and RADAR based Object Detection and Multi-Object Tracking, paying special attention to the methods that have set the guidelines in these fields. Typically these two tasks, which are crucial in the field of autonomous vehicle perception, have been tackled with only one of the two sensors used in this work or by fusion of one with cameras, with LiDAR and RADAR fusion being the least explored in recent literature [7] as can be seen in Fig. 2.

In the field of object detection using LiDAR, [8] set a guideline on which a great diversity of works have been developed. Point-based object detection methods take as input the raw information from point clouds, so the key concept for object detection is the iterative clustering and sampling of groups of points through the concept of voxelisation. The model took as input segmented 3D point clouds on which object classification tasks were performed by partial cloud segmentation. Empowered by a neural network working at point-wise level, transformations were performed using convolutional and max pooling layers.

In [9], it was acknowledged that 3D convolutions were a bottleneck to achieving a robust 3D detection in real time. Therefore, it is proposed a model that uses 2D convolutions with an encoder that learns scene features in pillars (vertical columns). The pillars allow the 3D scene to be converted into a 2D pseudo-image. Moreover, the model does not need hand-tuning to adapt to new sensor conditions (agnosticism), and the authors claim that it is even valid for use with radar point clouds.

With respect to the field of radar object detection, a number of methods to achieve this task are proposed in [10]. Five methods are developed and reviewed, covering a wide spectrum of possibilities with which to tackle this task, from traditional clustering to bird's eye view imaging (BEV):

- DBSCAN [11] clustering for detection and Long Short-term Memory (LSTM) [12] for classification.
- PointNet++ [8] for 3D semantic segmentation and detection with DBSCAN clustering [11].
- Grid-mapping and YOLO-v3 [13] for BEV-image detection.
- Object detection based on PointPillars [9].
- Combined pipeline of PointNet++ [8], DBSCAN [11] and LSTM [12].

In [14], the concept of tracking within the context of LiDAR and RADAR fusion for detection in urban environments was introduced. Applying an Extended Kalman Filter (EKF) [15] to cope with non-linearities coming from RADAR measurements, tracking was made possible.

With respect to similar works in tracking terms, in [16], a pipeline is presented that fuses LiDAR and camera information for object detection and tracking tasks, in addition to a forward prediction of the movement of vehicles surrounding the ego. In this case, BEV-SORT [17] is also used for tracking and the concepts of containerization and standard communications with ROS and Docker are exploited.

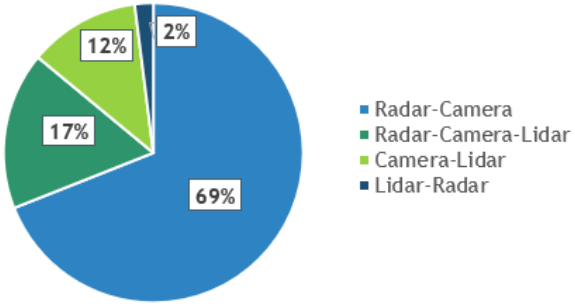

Fig. 2: **Status of Multi-Sensor Fusion** in the context of Autonomous Driving. Adapted from [7].

3 Methodology

In this work we take as a starting point the premise that the problem of perception in the field of autonomous driving can be solved by the coherent combination of a set of traditional processing techniques, which constitute a modular architecture. This modularisation enables a large problem to be solved by breaking it down into smaller, simpler tasks. Although these systems are weaker and more prone to error propagation between modules, a policy of strict constraints and filtering makes them viable [18]. This type of systems have a greater capacity for explainability and determinism in each of the modules, as opposed to end-to-end systems.

The present architecture is divided into four modules: two Object Detection modules, one each for LiDAR and RADAR, a sensor fusion module and a final Object Tracking module, which are shown in Fig. 3. In addition, our system pipeline exploits the concepts of lightweight Linux containers using Docker [19] to provide the system with isolation, flexibility and portability, and standard communication in robotics using the Robot Operating System (ROS) [20]. A

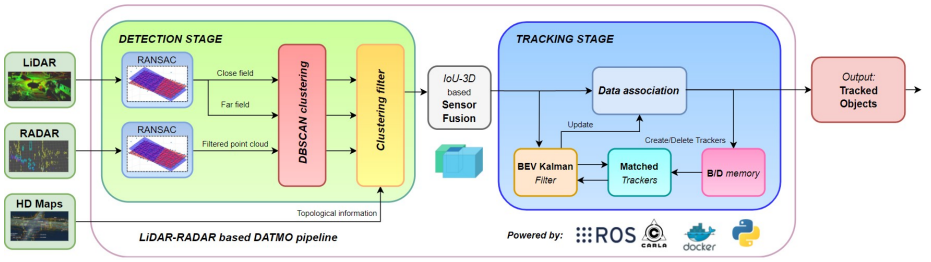

Fig. 3: **LiDAR-RADAR based DAMOT method.** From left to right all tasks involved in the method can be seen. With the information from the RADAR, LiDAR and HD Maps, the detection module is entered. In this module, the stages of plane filtering, clustering and flitting of tracker candidates are concatenated. Sensory fusion is then carried out, which provides the input for the tracking stage. Finally, the tracking is carried out through successive stages of data association and cycles of the BEV Kalman Filter, which are accompanied by the Birth-Dead memory module for the management of the tracked objects. These objects make up the output of the system.

use case of this method in action is the Traffic Jam on a highway which can be seen in Fig. 1.

3.1 LiDAR and RADAR Object Detection

The first stage of this modular pipeline is the Object Detection over the inputs of the system: the point clouds provided by a LiDAR mounted on the top and a RADAR mounted on the front of the ego-vehicle in CARLA Simulator. On the one hand, the LiDAR point cloud consists of detections in $P_L = x, y, z, i$ format: Cartesian coordinates and intensity of the reflection. On the other hand, the RADAR point cloud consists of detections in $P_R = x, y, z, r, \theta, \phi, v$ format: Cartesian and spherical spatial coordinates, as well as Doppler velocity. Before proceeding to the object detection, it is convenient to filter the input data, as it will facilitate this task. In addition, it should be noted that the spatial distribution of point clouds is different, so each point cloud will require a specific set of filters.

First, the processing of the LiDAR point cloud will be discussed. Taking into account that this sensor has been configured to imitate a Velodyne HDL-64, it can be stated that the light beams are emitted with a vertical FoV of [-25º, 2º]. This configuration makes it possible to take advantage of the height position of the sensor and to avoid occlusions caused by objects very close to the ego-vehicle. A large number of these light beams hit the ground directly, resulting in high density cluster candidates. To avoid the formation of these, the road plane is segmented using a RANSAC algorithm [21]. It is a robust model fitting algorithm that performs the plane segmentation task in a iterative repeated sequence, generating a random hypothesis and verification. It proposes

a random plane for n iterations and it maximizes the number of inliers fitting that plane according to a given threshold and is robust to a strong presence of outliers in the data. In addition, since it was observed that the density of incident points on the objects in the scene decreases considerably with distance, it was decided to divide the cloud into two parts at a distance of 25 m, giving rise to the concepts of close field and far field. This is done in order to be able to apply an adaptive clustering with distance.

Once the point cloud has been filtered and divided according to distance, a DBSCAN-based clustering process is applied for each of the defined zones. This process allows grouping the points belonging to a vehicle for a correct identification or detection. DBSCAN [22] is a clustering model that aims to find regions in a multi-dimensional space whose density is higher than certain user-defined parameters and which are separated by zones of lower density. The concept behind cluster formation is based on two parameters: `minPts` and ϵ. A point can be considered:

- **Core point:** if within a distance defined by a circle or sphere of radius ϵ there is a number of neighbours greater than `minPts`.
- **Border point:** if it is in the neighbourhood of a *core point*, but does not have a number of neighbours greater than `minPts`.
- **Noise point:** if the point do not meet any of the above cases.

Therefore, the resulting clusters will be formed by the *core* and *border points* directly reachable in density. This algorithm allows not having a priori knowledge of the number of vehicles (or clusters) present in the scene. The appearance of objects change with range, and after some distance, very few data points per objects are available to detect an object. This poses some challenges for detection, but this approach allows defining a series of different parameters for the close field and for the far field, in order to adapt to the far field cluster appearance.

On the other hand, RADAR is configured with a horizontal FoV of 160^{o} and a vertical FoV of $\pm 8^{\text{o}}$, emulating commercial configurations currently on the market or available in datasets [23]. The spatial distribution of the points in this cloud corresponds to a sparse point cloud, whose rays have no tendency to hit the ground. Given the scarcity of points and the importance of obtaining the native Doppler velocity of the detected objects, it is decided not to apply additional filters to those provided by the topological information of the map, as is done in the case of LiDAR. Consequently, another instance of the DBSCAN algorithm is applied to find the clusters belonging to the vehicles in the RADAR field of view. Given the clusters that are candidates for detection a positional filtering is made taking into account the information from HD Maps, remaining only those within the conduct zones. Lastly, a further geometrical filtering step is carried out. Those clusters with much larger dimensions, either in height, width or length or the product of the latter two, are rejected. All the remaining candidates pass to the fusion stage in Bounding Box (BB) format, and are defined by the vector $B = x, y, z, l, w, h, \theta, v$, where $[x, y, z]$ are the position in space of its centroid,

$[l, w, h]$ are its dimensions for each axis, θ is the yaw angle and v is Doppler velocity obtained with RADAR.

3.2 Sensor Fusion

The next stage of the pipeline is the sensor fusion of the object candidates clustered in the previous stage. How to fuse objects is an open question in academia and in this work we have chosen to consider as valid those objects detected by both sensors, which leads to redundancy, as well as those that are detected by one of the sensors and are out of the field of view of the other. For the first case, in which both sensors provide a detection, a spatial association mechanism using IoU-3D is introduced, which combines the spatial information of the clusters and assigns the velocity obtained by the RADAR to this candidate. Therefore, it is considered as a detected object and proposed for the tracking stage. It is only consistent to opt for a fusion based on full redundancy of detections when the FoVs of the sensors involved in the fusion are identical, as this implies no detection of occluded objects. Therefore, for the candidates in the second case, the occluded angles for both sensors are examined and objects that are proposed by one sensor, but are out of range of the other, are taken as detected objects. The approach enables the height advantage of LiDAR at close distances to be exploited, although these objects will not have a native velocity inference. Another advantage is the greater ease of clustering at long distances for RADAR. This combination of criteria results in an exploitation of the native strengths of both sensors.

3.3 Multi-Object Tracking

The third and last stage of the pipeline is a Multi-Object Tracking (MOT) module based on SORT that performs 3D Tracking from a Bird's Eye View (BEV). Adding the height dimension in solving this problem does not provide a clear improvement in the results and increases the computational cost and the completeness of the module. Since representing the space surrounding the ego-vehicle on a two-dimensional XY axis eliminates the main problem of traditional 2D tracking, occlusion, such an assumption has no negative effect on the result. As the Tracking module is fed by the filtered and refined detections, the concept of Tracking-by-Detection is reached.

BEV-SORT addresses MOT problem as a pipeline resulting from the combination of traditional techniques, mainly Kalman Filter (KF) [24] and Hungarian Algorithm (HA) [25], that are used for state estimation and data association, respectively. The algorithm makes use of a motion model to propagate the identity of a detected object from one timestamp to the next. The identity of each detected object is stored in a vector of 8 variables:

$$B_{trk} = [x, y, l, w, \theta, x', y', \theta']' \tag{1}$$

As a result of the detection and fusion modules, these vectors are fed with 7 of the 8 variables, with only θ' being inferred. When a detection is associated to

Table 1: Evaluation in the AD PerDevKit dataset for all scenarios tackling Object Detection with the fusion between LiDAR and RADAR. For evaluation difficulties: E stands for Easy, M for Medium and H for Hard, depending on the evaulation range.

Scene	Town	Weather	Environment	F1 score		
				E (25m)	**M (50m)**	**H (75m)**
1	03	Day	Urban	46.84	14.68	10.51
2	03	Night	Urban	43.67	21.09	17.91
3	05	Day	Urban	26.40	14.69	11.99
4	05	Day	Highway	49.05	29.03	21.42
5	06	Day	Highway	**57.45**	29.40	20.07
6	06	Day	Highway	31.47	19.43	14.60
7	06	Rainy	Highway	23.23	14.86	10.97
8	07	Day	Rural	19.86	11.08	8.99
9	07	Rainy	Rural	54.70	**42.86**	**31.66**
10	10HD	Day	Urban	29.26	14.22	10.73
11	10HD	Night	Urban	21.97	9.70	7.28
TOTAL				36.43	20.67	16.31

an object in the scene, a B_{trk} is produced and used to update the state of the object and predict its behaviour at the next timestamp via KF. If there is no detection associated with the object, it is predicted without correction or update using the motion model. To associate the detections to the objects stored by the algorithm, a cost matrix is proposed that computes the IoU of all detections with the objects. This provides the necessary input to apply the HA and see which new detection corresponds to the saved object, an assignment problem.

The identities of the objects and their motion models are created as they appear in the scene and are destroyed when they no longer appear in the detections. In addition, the algorithm has two user-customisable parameters that allow defining the behaviour in case of object creation and destruction. As a mechanism to avoid false positives, the user can decide the number of times an object is detected to be considered by the algorithm. In other words, if the same target appears n times in the detections, its identity and its motion model are created so that it counts in the processing. The second parameter is the number of times an object is not detected so that its identity is destroyed and it is not counted.

To conclude the method, it is indicated that the output provided by the approach is a list of tracked and identified objects that will serve as input to the other modules of a modular architecture for the successful completion of autonomous navigation.

4 Validation and Experimental Results

In order to provide qualitative and quantitative metrics that reflect the strengths and weaknesses of the proposed approach, a quantitative validation method is chosen.

4.1 Validation method

The metrics of the method's performance are obtained by using our AD PerDevKit, a dataset that allows scoring the goodness of Object Detection in use cases of interest and complex scenarios designed in CARLA Simulator. This dataset, in its first version, consists of a series of complex scenarios and use cases recorded on the different CARLA maps, representing conventional autonomous driving scenes in urban, rural and highway environments.

A suite of sensors integrated in the tool is composed of a 360^{o} LiDAR, a 90^{o} frontal RADAR, a frontal monocular camera, and IMU and GPS sensors. Given the nature of our proposal, it is decided to carry out an evaluation on the 11 available scenarios in a 90^{o} frontal field of view to measure the performance of Object Detection. Goodness of the approach is validated with F-beta score. For the case in point, $\beta = 1$.

- **F-beta score (F_β) [Eq. 2]:** Measure of a test accuracy calculated from the precision and the recall: True Positives (TP), False Negative (FP) and False Negative (FN) detections.

$$F_\beta = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2 \cdot FN + FP} \tag{2}$$

To carry out the validation, an efficient evaluator is developed which takes as input the groundtruth provided by the dataset and the detections made by the presented method and calculates the offline metrics.

4.2 Experimental results

A series of experiments is decided to evaluate the performance of the method, comparing the fusion performance against simple Object Detection pipelines with LiDAR or RADAR only. Furthermore, thanks to the diversity of AD PerDevKit scenarios, it is possible to compare the performance against different types of driving environments and weather conditions. The evaluation is performed in three difficulties, differentiated by the range of distance to be detected: Easy (25m), Medium (50m) and Hard (75m). The best scores are marked in **blue** in Tables 1 and 2.

Table I shows the performance of the algorithm for each and every AD PerDevKit scenario. On Easy difficulty, a maximum score of 57.45 is obtained for Scenario 5 (day and highway). For Medium and Hard, the maximum score is

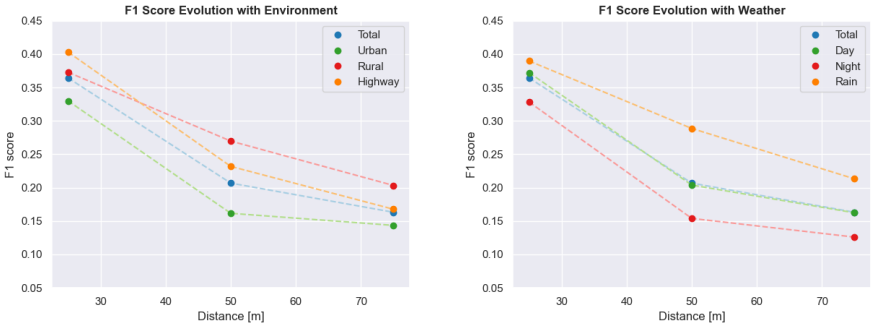

Fig. 4: Evolution of F1 score with respect to the distance evaluated over the different environments (left) and climate (right) scenarios offered by AD PerDevKit.

obtained in Scenario 9 (rainy and rural environment), with 42.86 and 31.66, respectively. Weighing all scenarios, a score of 36.43, 20.67 and 16.31 is obtained for each difficulty. Regarding the driving environments, Fig. 4 (left) shows that the best environment in which the method performs is Highway for Easy and Rural for Medium and Hard. In the case of weather conditions, the method performs best in rainy scenarios, as shown in Fig. 4 (right), presumably thanks to RADAR information.

Table II shows an ablation between the Object Detection performance and the full fused module and the sensors alone. It can be seen that the system improves significantly (11.21 points) by keeping the same detection method but including fusion and tracking over the sensed detections.

Table 2: Complete evaluation in the AD PerDevKit dataset.

Mode	F1 score (%)		
	Easy (25m)	Medium (50m)	Hard (75m)
Only LiDAR	25.22	16.14	12.13
Only RADAR	16.23	7.74	5.96
LiDAR+RADAR	**36.43**	**20.47**	**16.31**

5 Conclusions and Future Works

This paper presents the development, implementation and validation of an Object Detection and Multi-Object Tracking pipeline module using LiDAR and RADAR in CARLA Simulator. Using state-of-the-art software tools, such as ROS and Docker, and Machine Learning techniques such as DBSCAN and BEV-SORT. In addition, a quantitative evaluation is performed over AD PerDevKit

dataset, which demonstrates that sensor fusion is beneficial over the inclusion of a single sensor, while maintaining the same detection methods. This work opens the way for future works on carrying out an ablation on Tracking and how to improve this fusion with state-of-the-art methods, or with different configurations or arrangements of the same sensors at different mounting positions.

Acknowledgements. This work has been funded in part from the Spanish MICINN/FEDER through the Techs4AgeCar project (RTI2018-099263-B-C21) and from the RoboCity2030-DIH-CM project (P2018/NMT- 4331), funded by Programas de actividades I+D (CAM), cofunded by EU Structural Funds and Scholarship for Introduction to Research activity by University of Alcalá.

References

1. Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A Wright, Joseph E Gonzalez, and Ion Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.
2. Raphael Chekroun, Marin Toromanoff, Sascha Hornauer, and Fabien Moutarde. Gri: General reinforced imitation and its application to vision-based autonomous driving. *arXiv preprint arXiv:2111.08575*, 2021.
3. Carlos Gómez-Huélamo, Alejandro Díaz-Díaz, Javier Araluce, Miguel E Ortiz, Rodrigo E Gutiérrez-Moreno, Felipe Arango, Ángel Llamazares, and Luis M Bergasa. How to build and validate a safe and reliable autonomous driving stack? a ros based software modular architecture baseline. In *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022.
4. Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 13–15 Nov 2017.
5. Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. Automotive radars: A review of signal processing techniques. *IEEE Signal Processing Magazine*, 34(2):22–35, 2017.
6. Javier de la Peña, Luis M Bergasa, Miguel Antunes, Felipe Arango, Carlos Gómez-Huélamo, and Elena López-Guillén. Ad perdevkit: An autonomous driving perception development kit using carla simulator and ros. In *2022 IEEE International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022.
7. Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *Ieee Access*, 8:2847–2868, 2019.
8. Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.
9. Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

10. Nicolas Scheiner, Florian Kraus, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. Object detection for automotive radar point clouds–a comparison. *AI Perspectives*, 3(1):1–23, 2021.
11. Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.
12. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
13. Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
14. Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.
15. Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.
16. Carlos Gómez-Huélamo, Luis M Bergasa, Rodrigo Gutiérrez, J Felipe Arango, and Alejandro Díaz. Smartmot: Exploiting the fusion of hdmaps and multi-object tracking for real-time scene understanding in intelligent vehicles applications. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 710–715. IEEE, 2021.
17. Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.
18. Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.
19. Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.
20. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
21. Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
22. Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.
23. Ole Schumann, Markus Hahn, Nicolas Scheiner, Fabio Weishaupt, Julius Tilly, Jürgen Dickmann, and Christian Wöhler. RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications, March 2021.
24. Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.
25. Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.