Multimedia Tools and Applications (2022) 81:26915–26940
https://doi.org/10.1007/s11042-021-11624-2

1202: MULTIMEDIA TOOLS FOR DIGITAL TWIN

360° real-time and power-efficient 3D DAMOT for autonomous driving applications

Carlos Gómez-Huélamo[1] · Javier Del Egido[1] · Luis Miguel Bergasa[1] · Rafael Barea[1] · Elena López-Guillén[1] · Javier Araluce[1] · Miguel Antunes[1]

Received: 2 February 2021 / Revised: 25 August 2021 / Accepted: 23 September 2021 /
Published online: 8 January 2022

Abstract
Autonomous Driving (AD) promises an efficient, comfortable and safe driving experience. Nevertheless, fatalities involving vehicles equipped with Automated Driving Systems (ADSs) are on the rise, especially those related to the perception module of the vehicle. This paper presents a real-time and power-efficient 3D Multi-Object Detection and Tracking (DAMOT) method proposed for Intelligent Vehicles (IV) applications, allowing the vehicle to track 360° surrounding objects as a preliminary stage to perform trajectory forecasting to prevent collisions and anticipate the ego-vehicle to future traffic scenarios. First, we present our DAMOT pipeline based on Fast Encoders for object detection and a combination of a 3D Kalman Filter and Hungarian Algorithm, used for state estimation and data association respectively. We extend our previous work ellaborating a preliminary version of sensor fusion based DAMOT, merging the extracted features by a Convolutional Neural Network (CNN) using camera information for long-term re-identification and obstacles retrieved by the 3D object detector. Both pipelines exploit the concepts of lightweight Linux containers using the Docker approach to provide the system with isolation, flexibility and portability, and standard communication in robotics using the Robot Operating System (ROS). Second, both pipelines are validated using the recently proposed KITTI-3DMOT evaluation tool that demonstrates the full strength of 3D localization and tracking of a MOT system. Finally, the most efficient architecture is validated in some interesting traffic scenarios implemented in the CARLA (Car Learning to Act) open-source driving simulator and in our real-world autonomous electric car using the NVIDIA AGX Xavier, an AI embedded system for autonomous machines, studying its performance in a controlled but realistic urban environment with real-time execution (results).

Keywords Real-time · CARLA · LiDAR · 3D multi-object tracking · ROS · DAMOT · Autonomous navigation

✉ Carlos Gómez-Huélamo
carlos.gomezh@edu.uah.es

Extended author information available on the last page of the article

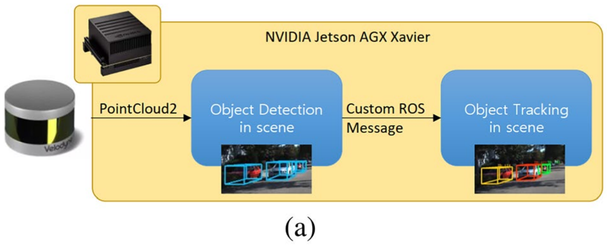
1 Introduction

Autonomous Driving Systems (ADSs) have to perform safe driving behaviours following conventional traffic rules to achieve a programmed destination.[1] In that sense, the perception layer is one of the most important modules of an AD stack, responsible of analyzing the online information, also referred as the traffic situation, through the use of a global perception system [12] which involves different on-board sensors as: Inertial Measurement Unit (IMU), Light Detection And Ranging (LiDAR), RAdio Detection And Ranging (RADAR), Differential-Global Navigation Satellite System (D-GNSS), Wheel odometers or Cameras. Regarding this, one of the most fundamental tasks in perception systems for AD is the capacity of tracking the most relevant obstacles (traffic participants) around the vehicle, also known as Multi-Object Tracking (MOT). A real-time and power-efficient MOT system is essential for self-driving, representing in most cases the preliminary stage before the prediction of the trajectory of the most relevant obstacles in the scene, allowing the car a reaction time to avoid critical situations or to anticipate its behaviour for the corresponding traffic scenario.

MOT systems aim to estimate the location, orientation and scale of all objects in the field of view of the vehicle over time. While object detection only captures the information of the environment in a single frame, a tracking system, which actually represents the next stage in the perception layer, must consider temporal information, filtering outliers (also referred as false positives) in consecutive detections and being robust to full or partial occlusions. Then, after tracking the most relevant obstacles in the environment (both static and dynamic), the vehicle can use this evolution of the scene over time to infer motion patterns and driving behaviour for trajectory forecasting (last stage of the perception module).

On the other hand, with the increase in object detection (especially in 3D object detection) performance in the last years has allowed the research community, especially those groups related to ADS, to focus on the tracking-by-detection [6] topic, which has become the dominant paradigm in 3D MOT. Essentially, in this paradigm a MOT system is made up by an object detector and a data association algorithm to establish the tracking-by-detection correspondence. The scope of this paper is to design a real-time and power-efficient 3D Multi-Object Tracking system as depicted in Fig. 1, based on the tracking-by-detection paradigm, for Intelligent Vehicles (IV) applications. This paper is an extension of our previous conference publication [7]. In this previous work, our results were limited to the analysis of our 3D MOT pipeline using the KITTI-3DMOT evaluation tool, based on [42], whilst in the present work we compare against more state-of-the-art methods and also introduce a preliminary version of our sensor fusion based DAMOT, merging the features extracted by a Convolutional Neural Network (CNN) using camera information for long-term re-identification and the obstacles retrieved by the 3D object detector. Despite we get some promising results incorporating the re-identification concept for long-term occlusion, specially in ter, the fusion in this preliminary version is not yet good enough to beat the metrics [2] obtained by our previous 3D object detection-only MOT pipeline. Implementing the most efficient pipeline in our real-world prototype is also part of this extension. To summarize, the new contributions extending our previous work are as following:

[1] https://cutt.ly/XkuypcH



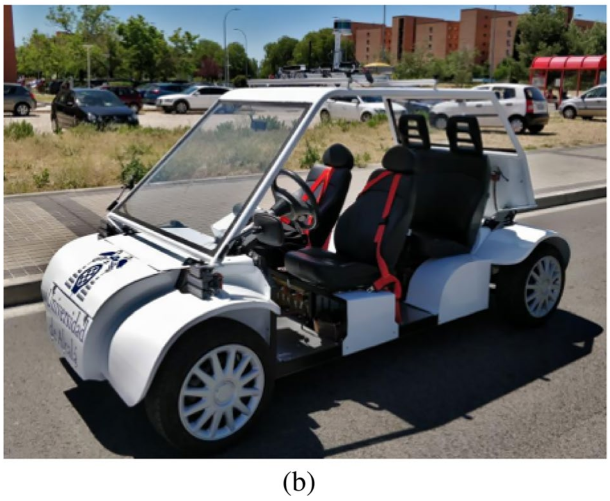


Fig. 1 (**a**) Our real-time and power-efficient MOT proposal running on the NVIDIA Jetson AGX Xavier embedded system (**b**) Our autonomous electric vehicle

- We carry out a comparison of different LiDAR based 3D object detectors with different types of input (voxel-based, point-based or combination) to analyze the tradeoff between Average Precision and Frequency (Hz).
- We introduce a sensor fusion based DAMOT, merging the extracted features by a Convolutional Neural Network (CNN) using camera information for long-term re-identification and obstacles retrieved by a LiDAR based 3D object detector.
- Implementation of our final DAMOT configuration in a power-efficient embedded system for real-world operation in an electric vehicle.

The remaining content of this work is organized as follows. The next section presents a review of the tracking-by-detection paradigm, covering both the concepts of vision and laser based 3D/Bird's Eye View (BEV) scene understanding in terms of Multi-Object Tracking. Section 3 illustrates a comparison of different 3D object detectors [40], presents our 3D object detection-only MOT pipeline and a preliminary version of our fusion with camera using re-identification. Section 4 compares both pipelines using the KITTI-3DMOT evaluation tool and implements the most accurate in our modular architecture [13] both in the CARLA (Car Learning to Act) [8] simulator and our autonomous electric car (see Fig.1b), in particular in an AI embedded system for autonomous machines such as the NVIDIA Jetson AGX Xavier (Fig. 1a). Finally, section 5 deals with the future works and concludes the paper.

2 Related works

A Multi-Object Tracking system is basically divided in two sequential stages: First, an object detector must obtain the most relevant obstacles in the scene. Second, a tracking module, based on a combination of state estimation and data association techniques, is used to track over time the obstacles throughout the scene. Many different technologies have been designed to accomplish an optimal environment detector following different approaches, based on different benchmarks such as KITTI [11], which provides manual labeled data from urban scenes taken from different cameras and LiDAR mounted on a vehicle. In this section we briefly cover the concepts of 3D scene understanding using both cameras and LiDAR.

2.1 Vision-only 3D object detection

Given their availability, widespread and affordability, cameras [47] are commonly used by nearly all algorithms presented so far as the primary perception modality. Nevertheless, in vision-based algorithms, object detection occurs in the projected image space in such a way the scale of the scene is unknown. In that sense, in order to make use of this valuable information for dynamic driving tasks (such as the position of pedestrians, traffic light or road infrastructure), it is required to bridge the gap from the 2D space (image) to 3D (metric or real-world). To do that, depth estimation is necessary, traditional methods use stereo cameras [9] or multi-view cameras to compute a pseudopointcloud by means the correspondence problem, in which 3D information can be extracted by examining the relative positions of objects in two different images. However, this image matching problem is expensive computationally, which adds an important amount of processing cost to an already complex perception pipeline. On the other hand, Machine Learning (ML) has recently taken over vision-only 3D object detection methods. These ML methods (based on a 3D occupancy grid called voxel grids) have also notably been applied to RGB-D [38] giving rise to similar, but coloured, pointclouds, though their range is limited and unreliability outdoors make RGB-D inpractical for AD applications. Shortly thereafter, 3D scene understanding through Deep Learning based algorithms is possible by processing the information using Convolutional Neural Networks (CNN) followed by applying complex projection transforms as in [17, 28] or [23], which first regresses relatively stable 3D object properties using a Deep CNN and then combines these estimates with geometric constraints provided by the corresponding 2D bounding box to produce a complete 3D bounding box. In this context, GPUs may be used to reduce computational cost, getting well processing time values. Minor pixel mistakes in image detection may produce meaningful differences between the 3D bounding box estimation and reality when the object is at a relevant distance.

2.2 3D LiDAR object detection

In contrast to vision-based 3D Object Detection, detecting the bounding boxes directly in 3D has the potential to design the appearance and motion models in 3D space without perspective distortion [42], instead of detecting the obstacles in image plane and then retrieving their remaining 3D information. 3D LiDAR sensors provide accurate 3D information, avoiding precision problems when positioning the objects in the three-dimensional space, as well as avoiding to well-known associated problems to cameras, such as their poor performance with adverse weather (deep night or fog) or challenging light conditions. Naturally, the quality of the detected bounding boxes is essential for the final tracking accuracy. Modern LiDAR based 3D object detection approaches usually belong to one of the following two categories [4] or a combination: Point- or Voxel-based. Regarding point-based methods do not required this quantization step, but they directly apply PointNet++ [27] on the input point cloud for detecting the objects in the 3D space. On the other hand, voxel-based methods divide the input point cloud (3D space) into equally-sized 3D voxels to generate 3D feature tensors based on the points inside each voxel. Then, the feature tensors are fed to 3D CNNs to predict the position of the bounding boxes. In this work we use PointPillars [19], a voxel-based state-of-the-art 3D object detector, due to its computational efficiency. This end-to-end network analyzes the raw point cloud by clustering it into upright columns to process the data as a 2D pseudo-image in such a way that highly efficient 2D convolutions can be applied on GPU, not requiring performing a fine-tuning process to improve the quality of the network parameters. We trained it by using the KITTI [11] Multi-Object Tracking dataset, containing information of vehicles, cyclists and pedestrians in arbitrarily complex urban environments.

2.3 2D multi-object tracking

Both 2D and 3D MOT systems can be divided into two branches based on the way data association is performed: Batch and online methods. While batch methods aim to find the global optimal solution by using the whole sequence, using network flow graphs that can be solved by minimum cost flow algorithms [33, 48], online methods consider the data association as a bipartite graph matching problem traditionally solved by a HA. Modern online methods solve the matching problem using deep association networks [41, 45], which are able to compute the association using neural networks. On the other hand, regardless the data association process, the design of a proper cost function for affinity measure is a crucial step to the MOT system. Some works employ hand-crafted features [26, 48] like color histograms and spatial distance as the cost function. Recent works apply the motion model and learn the appearance features [5, 44] to avoid identity switching when crossing trajectories.

2.4 3D multi-object tracking

As commented above, most 3D MOT systems present the same features than 2D pipeline. The point is, as expected, the detection stage. Regarding motion estimation and trajectory prediction, analyzing the scene with a 3D object detector has the potential to design the appearance and motion models in 3D space without perspective distortion, that is, a 3D MOT pipeline allows the system to estimate real velocity, undistorted by bi-dimensional transformations, being able to perform predictions based on linear and angular velocities and precise location and size information. Some works attempt to address this problem from different perspectives. [25] and [14] use an Unscented Kalman Filter (UKF) and a standard Kalman Filter in the BEV space to estimate both the linear and angular velocity of the obstacles, which is usually good enough for IV applications, preventing the tracking system to experience object occlusion and being computationally more efficient. [31] proposes an image-based method that estimates the position of objects in image plane as well as their distance to camera in 3D. Then, a Poisson Multi-Bernoulli Mixture (PMBM) filter is applied to estimate the velocity of the obstacles in the 3D space. Complexer-yolo

[37] presents a novel fusion of Deep Learning based state-of-the-art 3D object detection and visual semantic segmentation in the context of AD, processing voxelized input features with a variable depth of dimension instead of fixed RGB-maps to perform MOT and real 3D prediction. [1, 10] employ Siamese networks to learn the corresponding filters from data instead of using hand-crafted filters. [24] proposes the state estimation using a 2D/3D KF to jointly utilize the observation from the image and 3D world.

3 Our approach

As stated above, the goal of a 3D MOT pipeline is to associate the detected 3D bounding boxes over time. Previous works use relatively complicated filters to predict, in an accurate way, the spatial features of the obstacles in the scene. Our method, illustrated in Fig. 3 uses [42] as baseline, in which only the detection at the current frame and associated trajectories from the previous frame are required, which employs a 3D KF for tracking, where each obstacle state includes the 3D centroid position, rotation angle and 3D bounding box dimensions, excluding the angular velocity. Nevertheless, according to the original [42], the tracking stage is fed by a batch with all the results of the 3D object detector throughout the whole scene, which does not match with real-time requirements. In our case, our final objective is to implement the system in a power-efficient embedded system, so actual real-time operation is required, creating or removing tracklets (trackers associated to the corresponding objects) along the sequence in order to get a better perspective of what is happening around the ego-vehicle. Moreover, in this section we extend our previous work [7] by studying different 3D object detectors from the OpenPCDet [40] framework for the detection stage, as well as implementing a preliminary version of sensor fusion incorporating long-term re-identification with camera information. In that sense, both pipelines merge the concepts of online and real-time DATMO (Deteccion and Tracking of Multiple Objects), standard communication in robotics using the Robot Operating System (ROS) [29] and lightweight Linux containers using Docker [22] to enhance the integration of our MOT systems in fully-autonomous driving architectures [12].

3.1 3D object detection

The first step our MOT algorithm must carry out is to detect the bounding boxes of the most relevant obstacles in the environment around the vehicle. We study and run (Table 1) different Deep Learning (DL) based 3D object detectors from the OpenPCDet [40] framework, considering both the average precision for each class and the frequency. Recently, several neural networks have been applied to analyze the most relevant traffic participants, such as vehicles, cyclists and pedestrians. As commented above, the most typical classification of DL based 3D object detectors is point-based [34, 36], voxel-based [19, 46], which require preprocessing of the pointcloud, or a combination of both techniques [35]. Based on the PointNet [27] backbone (which analyzes the pointcloud using the detections computer over the image plane), [34] was the first neural network to analyze the raw pointcloud without preprocessing to obtain accurate 3D bounding boxes. It divides the clustering in two stages, being the former (Bottom-up 3D proposal Generation) responsible of studying the association of a laser point to a particular object and the latter (Canonical 3D box refinement) computes other features on top of the location of every point such as the location to the sensor to compensate the lower accuracy for further points. Part-A^2 [36]

Table 1 Comparative of LiDAR based 3D Object detectors in the KITTI validation set. We bold the best results in **black** for each metric and in **blue** the metrics of our chosen proposal

Method	Type of input	Frequency (Hz)	Car (AP)	Pedestrian (AP)	Cyclist (AP)
PointPillars [19]	Voxel	41.7	86.46	57.75	80.057
SECOND [46]	based	19.8	88.61	56.55	80.59
PointRCNN [34]		6.3	88.94	61.89	85.01
PointRCNN-IoU [34]	Point	6.3	89.01	62.69	87.48
Part-A^2-Free [36]	based	5.6	89.12	**70.31**	87.65
Part-A^2-Anchor [36]		7.5	89.56	65.69	85.50
PV-RCNN [35]	Combination	4.6	**90.35**	63.12	**88.34**

improves the results obtained by PointRCNN focusing on the rays of the sensor that may impact the interior of the vehicle (for example, going through a windscreen) instead of the exterior of the vehicle. This network also presents two stages: The former extract the main features of the object, learning to estimate the interior points of the vehicle while generating 3D proposals that fit the object. The latter, responsible of encoding the points to identify the objects, may present to behaviours: anchors-free, saving computational time, or using anchors in order to retrieve the best bounding boxes, as illustrated in Table 1.

On the other hand, we have the voxel-based object detectors. In 3D computer graphics, a voxel represents a value on a regular grid in three-dimensional space. In that sense, SECOND [46] discretizes the pointcloud in voxels with the final objective of condensing the whole information from the point cloud into a smaller number of points, keeping the detection precision but with a remarkably reduced inference time. This network presents four stages in row to process the pointcloud and return the classifier, box regressor and direction classifier: Voxel Features and Coordinates , Voxel Feature Extractor, Sparse Conv Layers and RPN. The first stage discretizes the pointcloud as a set of voxel computing and computing the number of original points that are stored for each voxel. Then, a neural network extracts the features from each point and voxel, encoding the information in a single vector. The third stage carrries out the convolutions to transform the three-dimensional information into the Bird's Eye View (BEV) image plane, also referred as pseudo-image. Finally, SECOND presents a Region Proposal Network (RPN) that analyzes the feature maps to generate the 3D bounding boxes that fit the corresponding objects. Furthermore, PointPillars [19] continues the work proposed by SECOND, being made up by three sequential stages: The first one (Pillar Feature Net) clusterizes the original pointcloud in upright columns. The XYZ information and the reflectance of each point, in addition to the distance of each point to the geometric center of the pillar and the geometric center of the pillar itself, are employed to encode the information of each pillar in a 9-dimensional vector. Then, a bi-dimensional image in the BEV plane is obtained (2D pseudo-image). The second stage (Backbone - 2D CNN) uses 2D convolutions applied on GPU to process this BEV pseudo-image in a similar way to VoxelNet [51] and finally the third stage (Detection Head - SSD) retrieves the 3D bounding boxes that fits the detected objects.

Finally, PV-RCNN [35] proposes a combination between both approaches (raw pointcloud and discretized pointcloud). First, it discretizes the pointcloud in voxels, applies 2D convolutions and proposes 3D bounding boxes that contain the corresponding objects, in a similar way to SECOND and PointPillars. Second, a set of keypoints are identified from

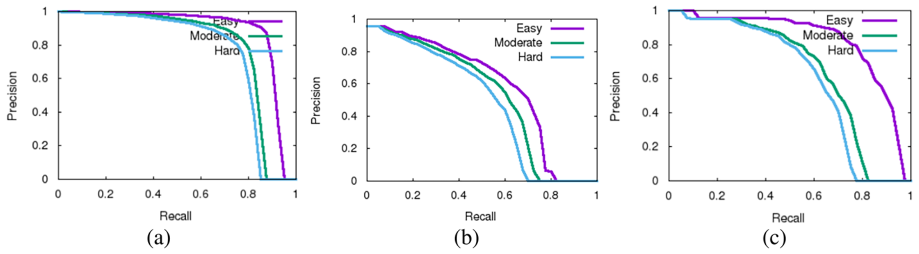


Fig. 2 Precision vs Recall curves obtained with PointPillars in the KITTI validation test for the (**a**) Car ,(**b**) Pedestrian and (**c**) Cyclist class, respectively

the retrieved 3D bounding boxes which are used in the final fine-tuning stage to predict the final shape of the bouding box. In that sense, this object detector leverages the efficiency of discretizing the pointcloud in voxels to improve the performance of the network, whilst it preserves the pointcloud without preprocessing to keep the accuracy of the information.

Table 1 shows an analysis of the previously mentioned 3D object detectors, we ran them over the KITTI [11] validation test in our PC desktop (Intel Core i7-9700k, 32GB RAM with CUDA-based NVIDIA GeForce RTX 2080 Ti 11GB VRAM) for the class Car, Pedestrian and Cyclist, which are the main classes evaluated by KITTI, and actually the vast majority of road participants in a standard traffic scenario. As it may be observed, PV-RCNN results beat the performance of the other object detectors, specially in terms of car. However, the system is relatively complex, specially in the non-processed pointcloud stage, so its frequency of execution (4.6 Hz) is almost four times slower than SECOND (19.8 Hz) and nine times slower than PointPillars. Then, considering that this work is focused on autonomous driving, in which the perception systems are expected to work at frequency of at least 10 Hz, we choose PointPillars for object detection, since it presents the best tradeoff between accuracy and real-time operation. Fig. 2 shows the different Precision vs Recall curves obtained with PointPillars in the KITTI validation test for the Car, Pedestrian and Cyclist class respectively, being clearly focused on the car class.

At a given frame *t*, the detections provided by PointPillars are given in the following form:

$$\mathbf{Det}_t = [\mathbf{Det}_t^1, \mathbf{Det}_t^2, ..., \mathbf{Det}_t^f] \tag{1}$$

Where *f* is the number of detected 3D bounding boxes at a given frame and threshold. Each detection in eq. 1 is represented as a 9-dimensional vector:

$$\mathbf{Det}_t^i = [x, y, z, l, w, h, \theta, type, score] \tag{2}$$

Where *x,y,z* correspond to the object centroid in LiDAR coordinates, *l,w,h* correspond to the length, width and height of the object respectively, θ its orientation angle around the LiDAR Z-axis, the object type (according to KITTI format) and detection confidence.

3.2 3D kalman filter - object state prediction

Once we have each 3D detection as shown in eq. 2, a 3D Kalman Filter is used to track the objects. Since the average frame rate of PointPillars is above 50 fps, rather than the threshold recommended for perception systems in IV applications (10 Hz), real-time is considered at the

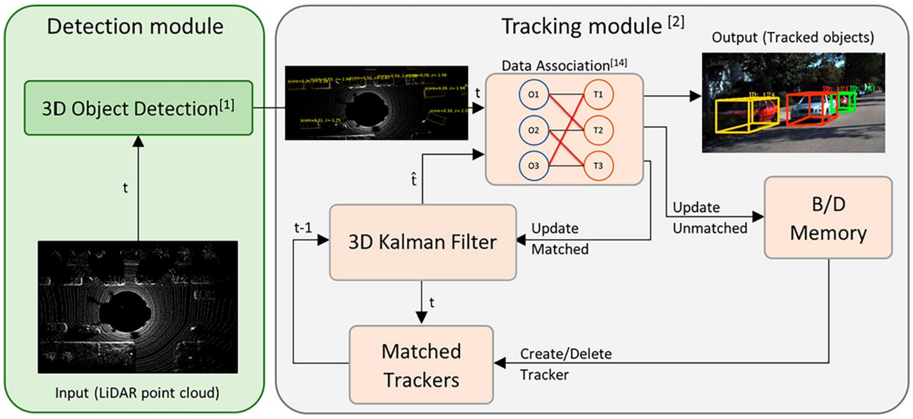


Fig. 3 *3DMOTsystempipeline*: (1) 3D object detection module provides the detected bounding boxes at frame *t* from the raw LiDAR pointcloud using ROS communications; (2) A 3D Kalman Filter predicts the state of trajectories in frame *t-1* to current frame $\hat{t}$ throughout the prediction step; (3) the detections at frame *t* and predicted trajectories at $\hat{t}$ are matched using the Khun-Munkres (a.k.a Hungarian) algorithm; (4) matched trajectories are updated based on their corresponding matched detections to obtain update trajectories at frame *t*; (5) Unmatched trajectories and detections are used to delete disappeared trajectories or create new ones respectively; (6) Matched predicted trajectories are returned to the system using ROS communications

detection module in such a way the inter-frame displacement of the objects can be approximated by using the constant velocity model, which is independent of other objects in the scene and of the LiDAR motion. Regarding this, the estimation of the measured variables in the following frame are:

$$x(\hat{t}) = x(t) + v_x$$

$$y(\hat{t}) = y(t) + v_y$$

$$z(\hat{t}) = z(t) + v_z$$

$$\theta(\hat{t}) = \theta(t) + v_\theta$$

It is important to note that unlike [42], we do include the angular velocity of the vehicle in the state space v_θ in order to predict the future trajectories of the vehicle considering a CTRV model (Constant Turn rate and Velocity magnitude model), which requires both the angular and linear velocity of the vehicle. Then, the state of each object trajectory is modelled as:

$$\mathbf{Tr}_t^j = [x, y, z, \theta, l, w, h, v_x, v_y, v_z, v_\theta] \tag{3}$$

As observed in Fig. 3, at every frame, a tuple $\mathbf{Tr}_t = [\mathbf{Tr}_t^1, \mathbf{Tr}_t^2, ..., \mathbf{Tr}_t^g]$, with length g is calculated, where each element corresponds to an association between a detection and a trajectory tracker. Then, based on the associations of the previous frame and the constant velocity model, the tuple $\mathbf{Tr}_{\hat{t}}$ is calculated, where each element corresponds to the predicted trajectory ($\mathbf{Tr}_{\hat{t}}^j$) in the current frame t expressed as:

$$\mathbf{Tr}^{j}_{\hat{t}} = [x(\hat{t}), y(\hat{t}), z(\hat{t}), \theta(\hat{t}), l, w, h, v_x, v_y, v_z, v_\theta] \quad (4)$$

Then, the set of detections at frame t and the predicted trajectories based on the previous frame *t-1* associations represent the input to the data association module at frame t, as shown in Fig. 3.

3.3 Data association

To associate the detections $\mathbf{Det}_t$ and the predicted trajectories $\mathbf{Tr}_{\hat{t}}$, the Khun-Munkres algorithm [18] (also referred as Hungarian algorithm) is used, a simple but accurate data associations algorithm. The resulting affinity matrix presents f rows (number of detections at frame t) and g columns, which correspond to the number of predicted trajectories based on the information of frame $t-1$. Each element of the matrix corresponds to the 3D-IoU between every pair of predicted trajectory and detection. Then, the bipartite graph matching problem is solved using the Hungarian algorithm, rejecting the matching if the 3D-IoU is lower than a given hyperparameter IoU_{th}, giving rise to a set of matched detections ($\mathbf{Det}_{matched}$) and predicted trajectories ($\mathbf{Tr}_{matched}$) (both sets present the same number of elements, h, that is, the number of matches), as well as a set of unmatched detections ($\mathbf{Det}_{unmatched}$), where $n = f - h$ is the number of unmatched detections, and a set of unmatched trajectories ($\mathbf{Tr}_{unmatched}$), where $m = g - h$ is the number of unmatched trajectories.

3.4 3D kalman filter - object state update

As observed in Fig. 3, once we have the corresponding sets of matched detections and trajectories, based on the Kalman Filter prediction-update cycle, the state space of each trajectory is updated based on its corresponding matched detection. To do that, we use the weighted average [16] between the matched detection values and the state space of the trajectory tracker. On the other hand, in the same way that [42], we appreciate that this state update step does not work properly for obstacle orientation. The reason is simple: Since the object detector is based on point cloud and no vision information is included, the object detector cannot distinguish if the obstacle is rotated a certain angle α or $\alpha + \frac{\pi}{2}$ around its Z-axis. That is, the orientation may differ by π in two consecutive frames. Then, if no orientation correction is applied, the Kalman Filter associated to the tracker can get easily confused, since it tries to adapts itself to the new orientation value rotating the object by π in following frames, giving rise to a low 3D-IoU value between new detections and predicted trajectories. Nevertheless, considering the high frequency of our object detector, we assume that obstacles must move smoothly and their orientation cannot be modified by π in one frame, in such a way when this happens the orientation of the corresponding matched detection or matched tracker can be considered wrong. To solve this problem, our detection module only considers angle from 0 to π (that is, if an angle exceeds π, π is substracted to the retrieved angle). Then, if the difference of orientation between a given matched detection and its corresponding matched trajectory is greater than $\frac{\pi}{2}$, as stated before, either the orientation of the detection or the orientation of the tracker is wrong. Then, we add π to the orientation of the tracker with the aim to be consistent with the matched detection.

3.5 Deletion and creation of track identities

When obstacles enter and leave the LiDAR grid, unique identities must be created or destroyed accordingly. In most tracking algorithms this is referred as the B/D (Birth and Death) Memory, which is based on the set of unmatched trackers and detections provided by the data association algorithm, where the unmatched trackers represent potential objects leaving the LiDAR grid, in the same way that unmatched detections represent potential objects entering in the analyzed environment. In order to avoid tracking of false positives (that is, clusters in the point cloud that actually do not represent a relevant obstacle, such a vehicle or a pedestrian), a new trajectory is not created until the unmatched detection has been continuously detected in the next f_{min} frames. Then, the tracker is initialised with the features of the detected bounding box, and the associated velocities set to zero. Note that, as stated in [3], since the velocity associated to the measured variables is unobserved at this moment (i.e., tracker initialization), the covariance initialises the value of the velocities (in the present work, v_x, v_y, v_z, v_θ) with large values, reflecting their uncertainty. To avoid removing true positives trajectories from the scene, they are not deleted unless they are not detected during a_{max} consecutive frames. This assumption prevents an unbounded growth in the number of localisation errors and trackers due to predictions over long duration where the object detector does not provide any correction. As shown in Fig. 3, the inputs to the Matched Trackers module are the updated matched trajectories from the 3D Kalman Filter and a set of created and deleted trackers, which jointly represent the input trajectories for the prediction step in the next frame.

3.6 Long-term re-identification

The previous points of this section represent the workflow illustrated in Fig. 3, which is summarized on the combination of a powerful 3D object detector and the Simple Online and Real Time [3] algorithm. As observed, our first pipeline represents a lean implementation of a tracking-by-detection framework for the problem of MOT where objects are detected in each frame and represented as bounding boxes. However, while achieving overall good performance in terms of accuracy and tracking precision, the SORT algorithm returns a high number of identity switches since the employed association metric is only accurate when state estimation uncertainty is low, for example, in the presence of a fast object detector, or when the scene does not present quite common situations such as occlusions or false negatives due to weather conditions (either considering camera or LiDAR information). To overcome this drawback, we propose a preliminary second version of our pipeline, where sensor fusion is added in order to improve the reliability of our system, in particular incorporating camera information.

The aim of sensor fusion is to use the advantages of each sensor so as to improve the robustness and redundancy of the system, being confident even if the performance of a particular sensor is noticeable reduced due to the road conditions. In the this second pipeline we try to solve problem of long-term occlusions or false detections due to road conditions incorporating Re-ID (Reidentification) to our second pipeline (Fig. 5). In particular, we make use of the DeepSORT [44], which is one of the most widely and elegant Visual Object Tracking (VOT) frameworks, as an extension of the SORT algorithm, depicted in Fig. 4. In summary, considering the VOT task, given an input image and a 2D object detector which returns the most relevant obstacles of the image, the DeepSORT algorithm uses

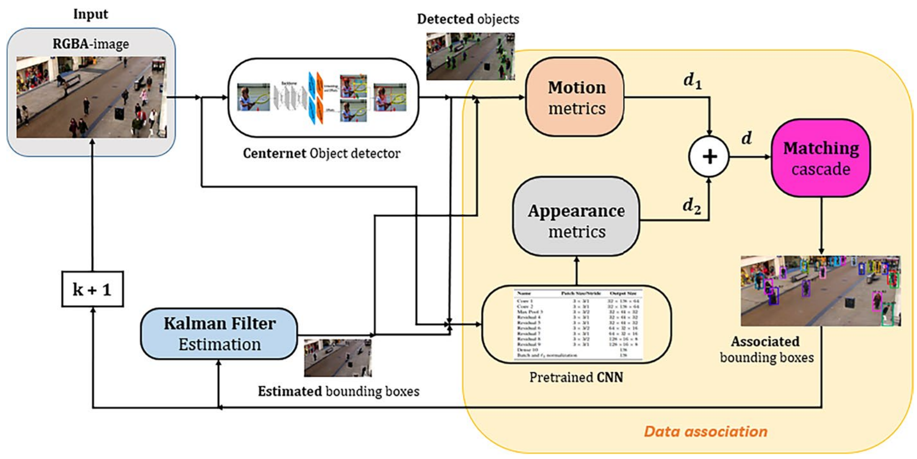


Fig. 4 Our Visual-Object Tracking flowchart based CenterNet [50] as object detector and DeepSORT as the tracking-by-detection algorithm

this bounding box information between the projection of the track distribution (Kalman Filter) and bounding boxes detection to compute the Mahalanobis distance (motion metric) between them. On the other hand, DeepSORT incorporates a CNN based feature extractor to replace the simple association metric formulated in SORT by a more informed metric that combines appearance information (fundamental for long-term re-identification) and motion information. The idea of this CNN based feature extractor is to compute a 128-dimensional feature vector for each detected bounding box. Then, by using a weighted sum of these motion and appareance metrics, DeepSORT is able to deal with long-term occlusions and predict in a very accurate way the feature pose of tracked objects.

We do not compute Visual Object Tracking using camera information since tracking objects in the 2D image plane and then retrieving the remaining 3D information present perspective distortion, even if the information is fused with a depth map prone to failure over long distances. In contrast, as observed in Fig. 5, using the corresponding calibration matrices the 3D bounding boxes are projected into the RGB image plane. On the other hand, 2D object detections are returned by our object detector (in this case, CenterNet [50] and then a 2D-IoU

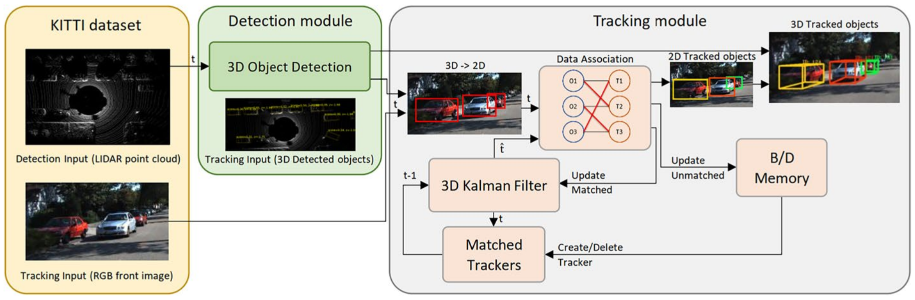


Fig. 5 *3D MOT system pipeline with long − term re − id*: Additional implementation over the pipeline illustrated in Fig. 3. In this case, we project the 3D object detections in the image plane at frame t, and 2D-IoU is computed between these projections and the 2D detections. Feature extraction is computed for each obstacle which IoU > IoU_{DS}, incorporating this information to the 3D Kalman Filter for long-term re-identification

is computed between the 3D projections and the 2D detections. The CenterNet algorithm is a CNN that detects each object as a triplet (topleft corner, center estimation and bottom-right corner), rather than a pair (only the corners) of keypoints, which improves both precision and recall. This technique is based on the CornerNet approach [20]. CornerNet represents each object by a pair of corner keypoints, which bypassed the need of anchor boxes and achieves the state of-the-art-one-stage object detection accuracy. Nevertheless, the CornerNet performance is restricted by its relatively weak ability of referring to the global information of an object. On top of that, since each object bounding box is constructed by a pair of corners, the algorithm is sensitive to detect the boundary of objects so not being aware of which pairs of keypoints should be grouped into objects. This weakness gives rise to some incorrect bounding boxes, most of which could be easily filtered out with complementary information, such as the aspect ratio. To address this weakness, CornerNet is equipped with the ability of perceiving the visual patterns within each proposed region in order to identify the correctness of each bounding box by itself. In that sense, CenterNet is a variation of CornerNet that explores the central part of a proposal (region that is close to the geometric center) with one extra keypoint. The statement is very simple: If a predicted bounding box has a high IoU (Intersection over Union) with respect to the groundtruth box, then, the probability that the center keypoint in its central region is predicted as the same class id is high, and vice versa. In other words, if is determined if the proposal is indeed an object by checking if there is a center keypoint of the same class falling within its central region. Since the approach only pays attention to the center information, the cost in minimal. In summary, we use CenterNet as our 2D object detector since it is simple (uses a keypoint detection technique to detect the bounding box center point and regress to all other object properties such as bounding box, pose or 3D information), versatile (works for standard object detection, multi-person pose estimation with minor modification and 3D bounding box estimation), fast (the whole process is included in a single network feed-forward) and strong (the best single model achieves 45.1 AP (Average Precision) on COCO test-dev). Moreover, it is important to consider that defining the bounding box and detecting the object largely depends on the size of the associated central region. For example, smaller central regions lead to a low recall rate for small bounding boxes, while larger central regions lead to a low precision for large bounding boxes. Both in simulation and of course in the real-world, an object must be tracked (and so previously detected) until it disappears from scene. Even if it is at a certain distance (small size in the scene) but still on-road, it must be detected since is relevant. In that sense, CenterNet is excellent because it proposes a scale-aware central region to adaptively fit the size of bounding boxes. The scale-aware central region tends to generate a relatively large central region for a small bounding box, while a relatively small central region for a large bounding box. Once 2D detections and image-projected 3D bounding boxes are matched using the Intersection-over-Union (IoU) metric, those bounding boxes with IoU value greater than a certain threshold IoU_{DS} are accepted and feature extraction is computed, in order to incorporate this 128-dimensional vector to the 3D object information for long-term re-identification, but maintaining the tracking in the 3D space to avoid perspective distortion.

4 Experimental results

In order to evaluate our proposed 3D DAMOT pipelines before implementing in our real prototype, we use the KITTI dataset, which provide LiDAR point cloud and 3D bounding box trajectories. Nevertheless, as the KITTI test set only supports 2D MOT evaluation (3D tracklets are projected onto the image plane for MOT evaluation using the corresponding calibration matrices) and its groundtruth is not released to users, the KITTI val set must be used for 3D MOT evaluation. Moreover, the evaluation is carried out using the MOT evaluation tool proposed by [42] also referred as KITTI-3DMOT. Mainstream metrics applied to MOT systems are extracted from CLEAR MOT metrics [2], such as MOTA (Multi-Object Tracking Accuracy), MOTP (Multi-Object Tracking Precision), ML/MT (Number of Mostly Lost / Tracked trajectories), IDS (Number of identity swutches), FRAG (Number of fragmentations generated by false negatives) and FN/FP (Number of false negatives / positives). The two most important metrics are MOTP and MOTA:

MOTP measures the precision by analyzing the total error in the estimated position for each object and its associated ground-truth.

$$MOTP = \frac{d}{num_{gt}} \tag{5}$$

On the other hand, MOTA considers all errors made by the system, such as false positives, false negatives and identity switching.

$$MOTA = 1 - \frac{FP + FN + IDS}{num_{gt}} \tag{6}$$

where num_{gt} is the number of ground truth objects in all frames.

Nevertheless, these metrics analyze the DAMOT system performance at a given threshold, not taking into account the confidence provided by the object detector and possibly misunderstanding the capability of the method. That means they do not take into account the full spectrum of precision and accuracy over different thresholds. Moreover, these traditional metrics evaluate the performance of the MOT system on the image plane (by projecting the detected 3D bounding box onto the image plane), which does not demonstrate the full strength of 3D DATMO. In that sense, AB3DMOT [42] recently presented a 3D extension of the KITTI 2D MOT evaluation, known as KITTI-3DMOT, which focuses on the dimensions, orientation and centroid position of the 3D bounding box instead of the projection onto the image plane to evaluate the performance of the MOT system. Moreover, two new integral MOT metrics are introduced in order to solve the problem of evaluating the MOTA and MOTP of the system across all thresholds, known as AMOTA and AMOTP (Average MOTA and MOTP), as shown in eq. 7:

$$AMOTA = \frac{1}{L} \sum_{\{\frac{1}{L}, \frac{2}{L}, \ldots, 1\}} (1 - \frac{FP + FN + IDS}{num_{gt}}) \tag{7}$$

Where L is the number of different recall values. Note that IDS, FP and FN are modified according to the results of each threshold value. Likewise, AMOTP can be estimated by integrating MOTP across all recall values.

4.1 3D MOT pipelines evaluation

We compare our two proposed DATMO pipelines (the former, 3D SORT, based on a Deep Learning object detector and the SORT algorithm extended to the 3D space, and the latter,

Table 2 Comparative of 3D Multi-Object Tracking pipelines using the KITTI-3DMOT evaluation tool in the validation set (car class) using the traditional pipeline and incorporating long-term re-identification using camera information. We bold in **black the best results for each category**

Method	AMOTA (%)	AMOTP (%)	MOTA (%)	MOTP (%)	IDs
3D DeepSORT (Ours)	27.84	69.09	79.87	**84.13**	43
3D SORT [42] using PointPillars [19] (Ours)	39.90	**79.31**	**94.20**	82.06	150
mmMOT [49]	33.08	72.45	74.07	78.16	10
FANTrack [1]	**40.03**	75.01	74.30	75.24	35
Monocular 3D [43]	31.37	64.29	62.38	68.26	**1**

3D DeepSORT, including sensor fusion with camera and a feature extractor to achieve long-term re-identification) against modern open-sourced 3D MOT systems such as mmMOT [49], FANTrack [1] and Monocular3D [43] using the proposed KITTI-3DMOT. Results are observed in Table 2. Note that these results were obtained with default values of the hyperparameters in the tracking stage ($age_{max} = 1$, $min_{hits} = 1$, $IoU_{thr} = 0.1$). For a deeper information of these hyperparameters, we refer the reader to the next subsection.

We can observe that our proposed 3D MOT systems outperform other modern 3D MOT systems (mmMOT, FANTrack and Monocular 3D), in particular 3D SORT using PointPillars as 3D object detector. Our second system (3D DeepSORT) presents slight errors when incorporating the camera information and the feature extraction and must be improved, specially in terms of identity switching and false negatives which directly affect to the MOTA and AMOTA metrics, in such a way a deeper analysis of the fusion process between the 3D projected bounding boxes and 2D object detections, as well as the incorporation of the encoded detector for long-term re-identification, must be conducted. In that sense, our final configuration for this work is the classic 3D SORT algorithm fed by the detections provided by PointPillars.

4.2 Ablation study over 3D SORT

Once we decide to implement the 3D SORT pipeline in our real-world prototype, we carry out an ablation study that allows us to observe the performance in function of the tracking hyperparameters. These are:

- age_{max}: Maximum number of frames for a tracker (Kalman Filter) to be associated again to a certain detection
- min_{hits}: Minimum number of consecutive frames in which a tentative tracker must be associated to a detection to be considered as an actual tracker
- IoU_{thr}: Threshold to match a predicted trajectory and a detection in the data association module

Table 3 shows an ablation study by modifying these parameters. With a threshold IoU_{thr} of 0.01 we get quite similar results in terms of MOTA and MOTP, decreasing by 36 %

Table 3 Ablation study of the final DAMOT configuration using the KITTI-3DMOT evaluation tool in the validation set (car class). We bold the best results in **black** and the second best in **blue** for each metric

age_{max}	min_{hits}	IoU_{thr}	AMOTA (%)	AMOTP (%)	MOTA (%)	MOTP (%)	IDs
1	1	0.1	**39.90**	79.31	94.20	82.06	150
1	1	0.01	39.84	70.96	95.13	81.84	54
1	1	0.25	39.37	**79.35**	89.10	82.42	682
1	3	0.1	39.54	71.24	91.38	83.23	2
1	5	0.1	39.26	71.36	88.84	**83.68**	3
2	1	0.1	39.49	79.24	94.91	81.48	154
3	1	0.1	39.50	79.15	**95.16**	81.15	152

the number of identity switches (150 to 54). On the other hand, increasing the minimum number of hits allows us to reduce the identity switching noticeably, overcoming one of the main drawbacks associated to the motion metric proposed by SORT. Moreover, modifying the maximum age to consider a tracker has left the scene barely modifies the studied metrics. Finally, we bold in black the best values for each metric and in blue our final configuration ($age_{max} = 1$, $min_{hits} = 3$, $IoU_{thr} = 0.1$) that achieves an impressive number of 2 identity switches and quite acceptable CLEAR and integral metrics, which are key as a preliminary stage to predict the short-term for each trajectory in the motion prediction stage.

4.3 Qualitative KITTI evaluation

In order to evaluate our final configuration (3D SORT, using PointPillars as objects detector and Age_{max}, min_{hits} and IoU_{thr} hyperparameters equal to 1, 3 and 0.1 respectively), firstly we carry out the evaluation in the KITTI MOT benchmark based on the method proposed by [42]. The KITTI MOT benchmark is composed of 29 testing and 21 training video sequences, where each sequence is provided with the corresponding RGB images (left and right camera of the stereo pair), LiDAR point cloud and the corresponding calibration file. Using this information, we obtain some interesting qualitative results in the KITTI tracking dataset as a preliminary stage before implementing our final configuration in our real-world prototype. Results are shown in Fig. 6. We both create 3D bounding boxes (Fig. 6b) in the RVIZ simulator (ROS), here shown in BEV perspective, related to the LiDAR sensor, as well as projecting these 3D proposals to the front-view RGB image (Fig. 6a) in order to understand the output of the pipeline in a better way. It is important to highlight how our algorithm is able to perform 360^o DATMO (Fig. 6b), instead of only tracking the most relevant objects that fall in the Field of View (FoV) of the camera.

4.4 CARLA autonomous driving simulator

Despite the impressive efforts made by AB3DMOT [42], where a tool for evaluating 3D DAMOT systems directly in 3D space is designed, it is based on KITTI dataset [11], which provides prerecorded sequences over which the user cannot interact with the environment. Moreover, these sequences are usually based on common driving scenarios, such as a daily quiet street or a highway in which no challenging traffic scenarios as pedestrian crossing, give way, *etc.* takes place.

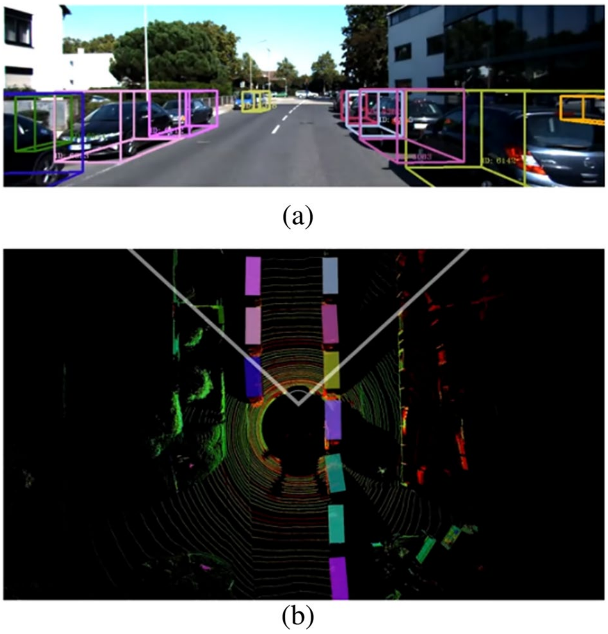


Fig. 6 Detection and Tracking of Multiple Obstacles evaluated in the (a) RGB left camera (b) LiDAR sensor

Regarding levels of automation, no industry organization has demonstrated a ratified testing methodology for L4/L5 (being identified the level 5 with a fully-autonomous navigation architecture, according to J3016 SAE document [39]) autonomous vehicles. The reason is quite simple: even though some regulations have been defined for these L4/L5 levels, simulation is a critical aspect to build safe autonomous vehicles. Nevertheless, in spite of the fact that current automotive companies are very good at testing the individual components of the navigation architecture, these tests are not powerful enough to validate a fully-autonomous navigation architecture on the road, so there is a need to figure out how to test intelligent vehicles full of advanced sensors and sharing information among them [32].

In terms of 3D Multi-Object Detection and Tracking, the answer is quite similar. Since the urban environment is highly complex, the whole architecture and particularly the 3D DAMOT system must be tested in countless traffic scenarios and environments, which would escalate the development time and cost exponentially with the physical approach, either testing at the real-wold or waiting for using new sequences of KITTI (recorded by a physical system), not studying the global advantages and drawbacks of the DAMOT system. For that reason, virtual testing (simulation) and an appropriate design of the traffic scenarios are the keys to build robust and safe autonomous vehicles in the future, as shown in [12]. Since the proposed 3D DAMOT architecture of this work is open-source, we decided to validate the ability to detect and track the most relevant objects around the vehicle in CARLA (Car Learning to Act) [8], an open-source hyper-realistic autonomous driving simulator that offers an outstanding environment in terms of perception, flexibility, traffic situations and real-time, which are key concepts for our system.

CARLA is an open-source autonomous driving simulator implemented as a layer over Unreal Engine 4 (UE4) [30]. This simulation engine provides to CARLA an ecosystem

Fig. 7 CARLA driving simulator

of interoperable plugins, a realistic physics and a state-of-the-art image quality. CARLA is designed as a server-client system so as to support this functionality provided by UE4, where the simulation is rendered and run by the server. The environment is composed of 3D models of static objects, such as buildings, infrastructure or vegetation, as well as dynamic objects like pedestrians, cyclists or vehicles. These objects are designed using low-weight geometric textures and models though maintaining visual realism by making use of variable level of detail and carefully crafting the materials. Moreover, one of the main advantages when using CARLA is the possibility to modify in an easy way the vehicle on-board sensors and their features in order to obtain accurate data, the weather and even the possibility to create realistic traffic scenarios as in Fig. 7.

In order to obtain the point cloud required by our 3D object detector, we use the CARLA ROS bridge, associated to the CARLA simulator. The CARLA ROS bridge is a ROS package that aims at providing a bridge between CARLA and ROS (Robot Operating System [29]), sending the information captured by the on-board sensors and other variables of interest to the vehicle in the form of parameters and topics understood by ROS. In this paper we use the 0.9.9 version of CARLA in such a way the ROS bridge was configured according to this version. In terms of the sensors perspective, the agent sensor suite can be modified in a flexible way. Most common sensors in CARLA world are GPS, RGB cameras and LiDAR (in addition to their corresponding pseudo-sensors, such us the semantic segmentation and the ground-truth associated to the RGB information of a camera). Since our final configuration is only LiDAR based, we configure the sensor as shown in Table 4. Based on the bridge, the 3D point cloud captured by the LiDAR is published in ROS format as PointCloud2, with the X axe inwards, Y left and Z pointing up.

As shown, CARLA provides a straightforward way to add or remove sensors from the vehicle or even modify their parameters, to adjust the simulation to the real-world as best as possible.

Table 4 LiDAR configuration in CARLA simulator

Parameter	Value
X (m)	0.0
Y (m)	0.0
Z (m)	2.5
Points per second	640,000
Upper FoV (^{o})	2.0
Lower FoV (^{o})	-26.0
Rotation frequency (Hz)	20

4.5 Qualitative DAMOT evaluation in CARLA

One of the best advantages of CARLA is the possibility to create ad-hoc urban layouts, helpful to validate the 3D DAMOT system under different traffic and weather conditions. CARLA Scenario Runner module can be downloaded from the CARLA GitHub, obtaining an execution engine for CARLA and traffic scenario definition. These scenarios can be modified by editing an OpenSCENARIO [15] script definition where town, vehicles, climate conditions and also driving behaviours are defined. Fig. 7 depicts an scenario on Town10 with the ego_vehicle and a predefined route, showing waypoints on a curved street to achieve the destination point. Furthermore, DAMOT evaluation in KITTI is mainly carried out along daily streets where many cars are parked on the road, so it mostly evaluate the system performance to track static vehicles when the main difficult is found in dynamic obstacles, such as pedestrians, vans, trucks or cars. Then, we design several traffic scenarios in the CARLA driving simulator to observe how the pipeline faces these more challenging situations. Nevertheless, we do not provide a quantitative evaluation of our MOT pipeline since CARLA does not offer a benchmark with labelled data and a validation tool. In that sense, we are developing a tool, referred as AB4COGT, that stands for A Baseline for CARLA Groundtruth generation, which aims to extract the groundtruth information of the CARLA objects and store the corresponding pointcloud frames, so as to be used as a tracking and detection benchmark in a similar way to KITTI. Fig. 8 illustrates this ongoing research. Hereafter, as an example of the power of our tool, two scenarios designed with it are described.

Scenario 1. Roundabout in a rainy day. This scenario represents a roundabout in a rainy day with low traffic density. One of the main problems on LiDAR sensors is their performance when they face bad weather conditions. Nevertheless, the quality of our object detector and our pipeline are able to overcome this drawback, as shown in Fig. 9a (BEV RGB image with the 3D proposals projected) and Fig. 9b (3D proposals in the RVIZ 3D visualization tool).

Scenario 2. Parked aside vehicles at night. In this scenario we reproduce a very common situation (as observed in the KITTI dataset) which is the ego_vehicle driving in narrow streets full of parked obstacles aside, evaluating its performance in night conditions. Despite this is probably the major disadvantage when using camera information (very poor performance in night conditions), we get impressive results in this situation, as illustrated in Fig. 10. This is pretty much coherent since LiDAR sensors are not passive sensors like cameras but they supply their own illumination source, which hits objects the reflected energy is detected and measured by the sensor in order to compute the distance to the object.

A video demo of the working of our proposal in the above scenarios and any others can be found in (video)[2].

[2] https://youtu.be/t5Dp4fbGUAw

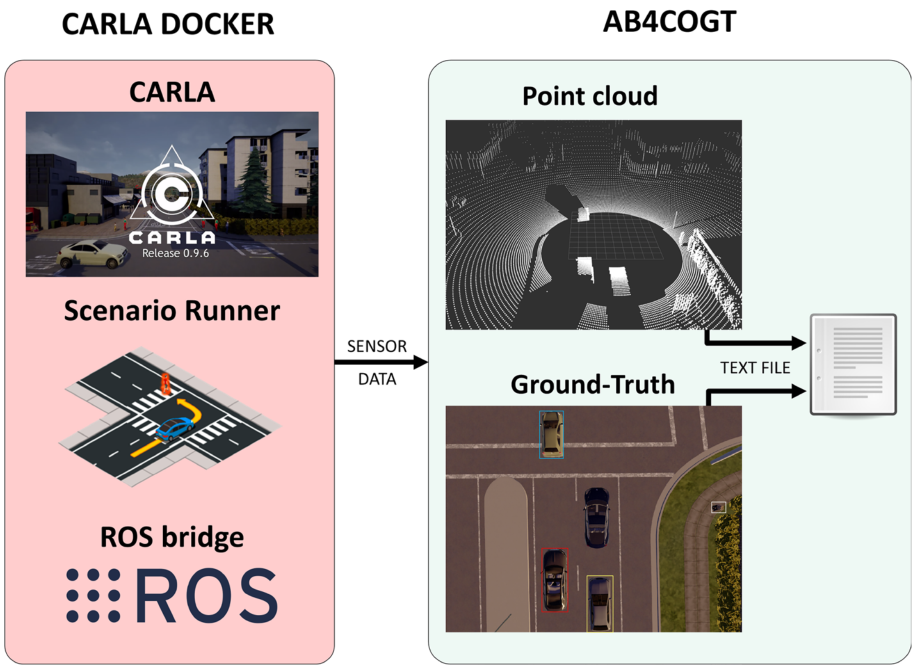


Fig. 8 DAMOT validation through AB4COGT and KITTI-3DMOT

4.6 Real-World evaluation in a power-efficient embedded system

The main objective of this paper is the design of a 360^o real-time and power-efficient DATMO pipeline, in order to run it on our real-world prototype. Perception systems in autonomous driving must process a huge amount of information coming from at least one

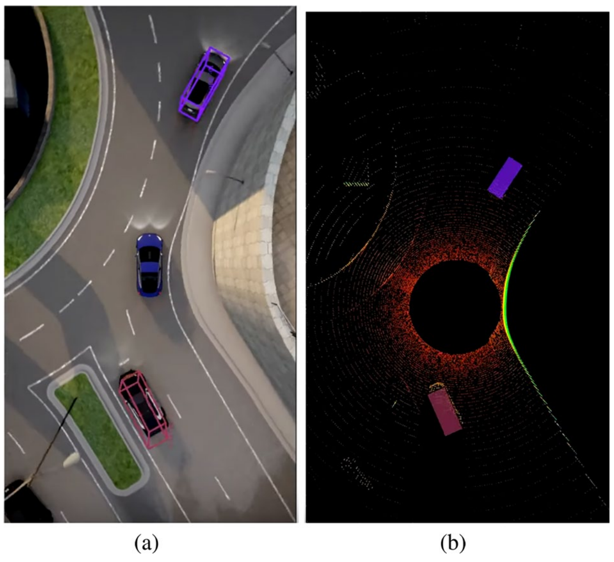


Fig. 9 DATMO in the roundabout with rain traffic scenario evaluated in (a) RGB left camera (b) LiDAR sensor

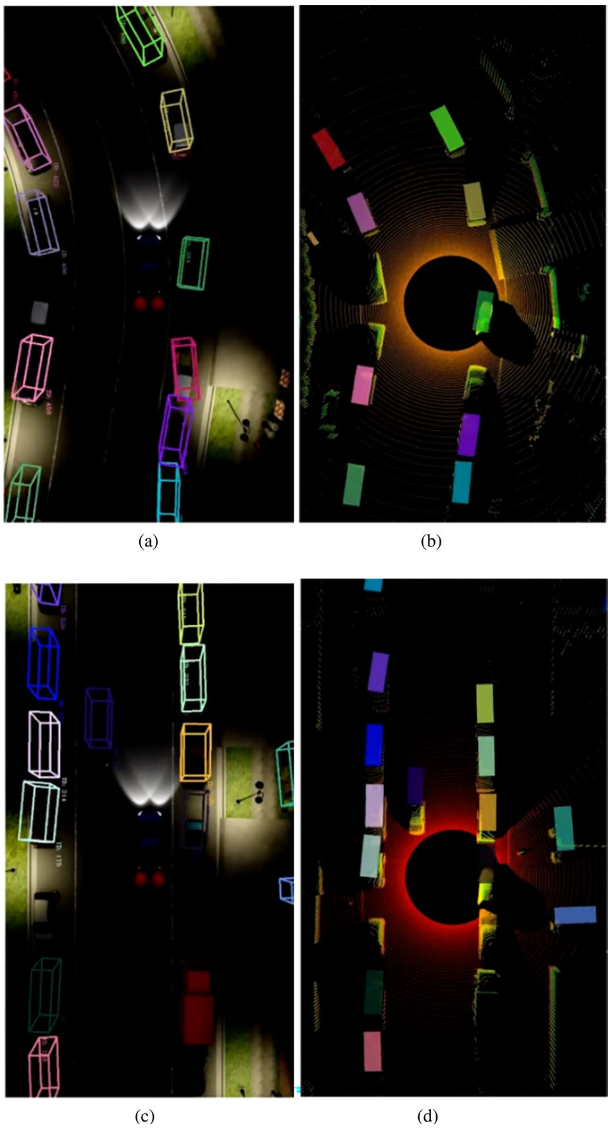


Fig. 10 DATMO in the parked aside vehicles at night traffic scenario considering a curved trajectory (a,b) and straight trajectory (c,d)

sensor in order to understand the environment. However, the physical space occupied by the processing units in the vehicle or their power consumption are metrics to be deeply analyzed, even more if these processing units will be integrated in an electric vehicle, where the state of the batteries is crucial. In that sense, the current approach is to use powerful but power-efficient AI embedded systems as computation devices for autonomous machines, since they present a remarkable ratio between performance and power consumption in a reduced-size hardware. Regarding the advantage of using neural networks in GPU, these embedded systems present a powerful GPU unit as well as fast storages based on solid state disks and a large RAM memory size. At the time of writing this paper, the best ratio of performance vs power consumption and size is represented by the NVIDIA Jetson embedded

Table 5 Comparative of inference frequency between the NVIDIA Jetson AGX Xavier and our PC desktop (Intel Core i7-9700, 16GB RAM) with CUDA-based NVIDIA GeForce RTX 1080 Ti 11GB VRAM

Stage	Frequency AGX Xavier (Hz)	Frequency PC desktop (Hz)	Ratio
Detection	7.3	**41.7**	5.7x
Tracking	15	**101.9**	6.7x

computing boards. NVIDIA Jetson is the world's leading AI computing platform for GPU-accelerated parallel processing in mobile embedded systems. These kits allow to implement state-of-the-art frameworks and libraries to conduct accelerated computing, such as CUDA, cuDNN or TensorRT (Tensor RealTime).

In this particular work we make use of the NVIDIA Jetson AGX Xavier, as observed in Fig. 1, which is as far as we know one of the most powerful AI embedded system specially designed for autonomous machines. Table 5 shows a comparative between the embedded system and our PC frequency in the inference stage, where the detection (PointPillars) is reduced by almost 6 times and the tracking by almost 7 times. Nervertheless, although the detection and tracking frequencies are on the border to be considered real-time according to the requirements of the perception systems for autonomous machines, the embedded system consumes 30 W whilst only the 1080 Ti GPU consumes 250 W at full power respectively. Considering that the embedded system computation power is reduced by 6.2 times (average between the detection and tracking frequency ratios) but only the GPU (not considering the whole PC desktop) presents a power consumption 8.3 higher, makes the

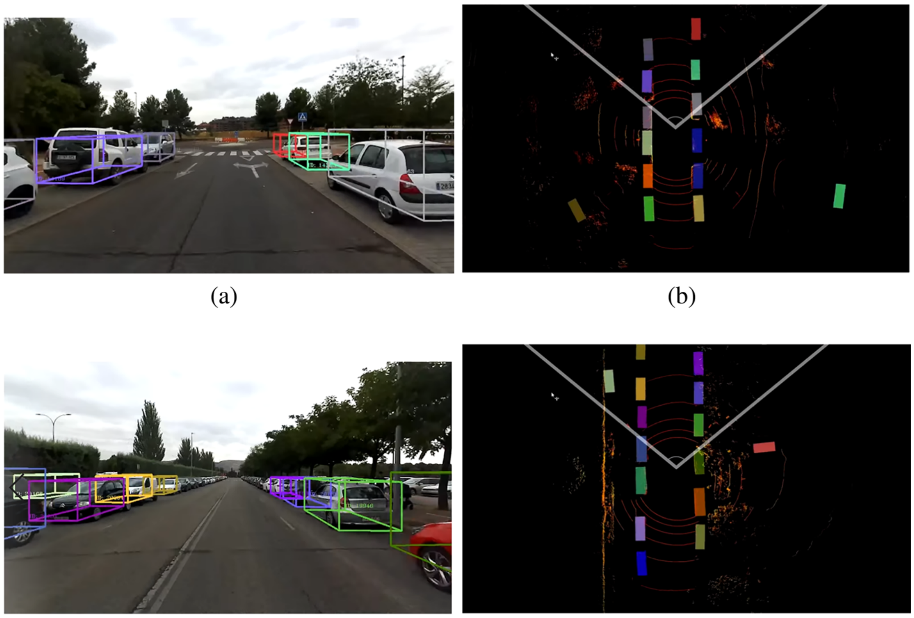


Fig. 11 DATMO in our campus with our real-world vehicle

current NVIDIA Jetson AGX Xavier a better suitable option for large scale-deployment in the autonomous driving field rather than using desktop graphic cards. Distributing several sensor processing across multiple embedded systems for parallelization will result in lower power consumption than using conventional GPUs in future autonomous driving prototypes. Qualitative results of running our DAMOT pipeline in our own vehicle, equipped with a VLP-16 LiDAR instead of the HDL-64 shown in CARLA and KITTI, are illustrated in Fig. 11. It can be appreciated that although the obtained results are slightly worse than with the KITTI dataset (equipped with a HDL-64 sensor), we obtain quite promising results, validating the pipeline studied in this work both in terms of accuracy and real-time operation.

5 Conclusions and future works

This paper presents an extension of our previous work, which studies the paradigm of 360^o real-time DAMOT for Intelligent Vehicles applications. Furthermore, we focus in this work in the power-efficient concept to design an optimal system which can be implemented in a embedded system for autonomous electric vehicles, with a remarkable ratio between power consumption and computing power. We first carry out an interesting review of different state-of-the-art Deep Learning based 3D object detectors, in which we opt for the voxel-based approach since it presents an impressive accuracy and rather meets the real-time requirements in the autonomous driving field. We formulate a second pipeline incorporating camera information in which a CNN feature extractor aims to help in the long-term re-identification to avoid identity switching throughout the route. Then, we compare our previous work with the new proposal using the KITTI-3DMOT evaluation tool and concludes that the sensor fusion is yet well formulated. The best pipeline allows to track static and dynamic objects around an autonomous car in real-time to enhance its safety system by only using a 3D LiDAR point cloud as input. Then, we design some interesting scenarios in CARLA and use our NVIDIA Jetson AGX Xavier to get some interesting qualitative results in simulation and real-world respectively. Further information can be found in attached (results)[3], showing the performance of the proposed method in different environments and situations. As future works, we will evaluate our DATMO model with higher order metric for evaluating MOT [21], in addition to use more complex datasets, such as KITTI-360, NuScenes, Waymo or AIODRIVE (CARLA simulator based). Moreover, we plan to analyze the effect of several 2D object detectors regarding our proposal of 3D MOT system pipeline with long-term re-id. In terms of embedded processor for real-time operation, we will thoroughly make a comparison between running the 3D MOT pipeline in a Jetson AGX Xavier, our PC desktop and our MSI laptop used in real-world tests. Regarding domain adaption, we will focus on reformulating our sensor fusion proposal using long-term re-identification in order to beat the results exposed throughout this work as well as studying the effects of Generative Adversarial Networks (GANs) in the long-term re-id process when analyzing camera raw data taken to adverse conditions such as nighttime or fog. To do that, we plan to formulate an optimal tool to carry out the quantitative DAMOT evaluation in CARLA by storing the groundtruth of the objects which fall in the grid of the object detector as well as the corresponding pointclouds, covering the concept of detection,

[3] https://cutt.ly/XkuypcH

tracking and motion prediction benchmarking in CARLA. Finally, the enhanced DAMOT pipeline will be integrated in our autonomous driving project, as a preliminary stage before conducting motion forecasting, in order to help the behavioural decision-making and the planning modules to improve the robustness and reliability of our system.

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) and cofunded by EU Structural Funds.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Baser E, Balasubramanian V, Bhattacharyya P, Czarnecki K (2019) Fantrack: 3d multi-object tracking with feature association network. In: 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE 1426–1433
2. Bernardin K, Stiefelhagen R (2008) Evaluating multiple object tracking performance: The clear mot metrics. EURASIP J Ima Video Proc 2008. https://doi.org/10.1155/2008/246309
3. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. 1602.00763
4. Chiu Hk, Prioletti A, Li J, Bohg J (2020) Probabilistic 3d multi-object tracking for autonomous driving. arXiv preprint arXiv:200105673
5. Choi W (2015) Near-online multi-target tracking with aggregated local flow descriptor. In: Proceedings of the IEEE international conference on computer vision. 3029–3037
6. Dao MQ, Frémont V (2021) A two-stage data association approach for 3d multi-object tracking. arXiv preprint arXiv:210108684
7. Del Egido J, Gómez-Huélamo C, Bergasa LM, Barea R, López-Guillén E, Araluce J, Gutiérrez R, Antunes M (2020) 360 real-time 3d multi-object detection and tracking for autonomous vehicle navigation. In: Workshop of Physical Agents, Springer 241–255
8. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) Carla: An open urban driving simulator. 1711.03938
9. Fan R, Wang L, Bocus MJ, Pitas I (2020) Computer stereo vision for autonomous driving. arXiv preprint arXiv:201203194
10. Frossard D, Urtasun R (2018) End-to-end learning of multi-sensor 3d tracking by detection. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE 635–642
11. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE 3354–3361
12. Gómez-Huelamo C, Bergasa LM, Barea R, López-Guillén E, Arango F, Sánchez P (2019) Simulating use cases for the uah autonomous electric car. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE. 2305–2311
13. Gómez-Huélamo C, Del Egido J, Bergasa LM, Barea R, López-Guillén E, Arango F, Araluce J, López J (2020) Train here, drive there: Simulating real-world use cases with fully-autonomous driving architecture in carla simulator. In: Workshop of Physical Agents, Springer. 44–59
14. Gómez-Huelamo C, del Egido J, Bergasa LM, Barea R, Ocaa M, Arango F, Gútierrez R (2020) Real-time bird's eye view multi-object tracking system based on fast encoders for object detection. In: 2020 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE

15. Jullien JM, Martel C, Vignollet L, Wentland M (2009) Openscenario: a flexible integrated environment to develop educational activities based on pedagogical scenarios. In: 2009 Ninth IEEE International Conference on Advanced Learning Technologies. IEEE 509–513
16. Kalman RE et al (1960) A new approach to linear filtering and prediction problems. J basic Eng 82(1):35–45
17. Konigshof H, Salscheider N, Stiller C (2019) Realtime 3d object detection for automated driving using stereo vision and semantic information. 1405–1410. https://doi.org/10.1109/ITSC.2019.8917330
18. Kuhn HW, Yaw B (1955) The hungarian method for the assignment problem. Naval Res Logist Quart 83–97
19. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O (2019) Pointpillars: Fast encoders for object detection from point clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). http://dx.doi.org/10.1109/CVPR.2019.01298
20. Law H, Deng J (2018) Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European conference on computer vision (ECCV). 734–750
21. Luiten J, Osep A, Dendorfer P, Torr P, Geiger A, Leal-Taixé L, Leibe B (2021) Hota: A higher order metric for evaluating multi-object tracking. Int j comp vision 129(2):548–578
22. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux j 2014(239):2
23. Mousavian A, Anguelov D, Flynn J, Kosecka J (2017) 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 7074–7082
24. Osep A, Mehner W, Mathias M, Leibe B (2017) Combined image-and world-space tracking in traffic scenes. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE. 1988–1995
25. Patil A, Malla S, Gang H, Chen YT (2019) The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. 1903.01568
26. Pirsiavash H, Ramanan D, Fowlkes CC (2011) Globally-optimal greedy algorithms for tracking a variable number of objects. In: CVPR 2011. IEEE 1201–1208
27. Qi CR, Yi L, Su H, Guibas LJ (2017) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. 5099–5108
28. Qin Z, Wang J, Lu Y (2019) Triangulation learning network: From monocular to stereo 3d object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
29. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng A (2009) Ros: an open-source robot operating system. vol 3
30. Sanders A (2016) An introduction to unreal engine 4. AK Peters/CRC Press
31. Scheidegger S, Benjaminsson J, Rosenberg E, Krishnan A, Granström K (2018) Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. In: 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE 433–440
32. Schöner H (2017) The role of simulation in development and testing of autonomous vehicles. In: Driving Simulation Conference, Stuttgart
33. Schulter S, Vernaza P, Choi W, Chandraker M (2017) Deep network flow for multi-object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 6951–6960
34. Shi S, Wang X, Li H (2018) Pointrcnn: 3d object proposal generation and detection from point cloud. 1812.04244
35. Shi S, Guo C, Jiang L, Wang Z, Shi J, Wang X, Li H (2020a) Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 10529–10538
36. Shi S, Wang Z, Shi J, Wang X, Li H (2020b) From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. IEEE transactions on pattern analysis and machine intelligence
37. Simon M, Amende K, Kraus A, Honer J, Samann T, Kaulbersch H, Milz S, Michael Gross H (2019) Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 0–0
38. Song S, Xiao J (2014) Sliding shapes for 3d object detection in depth images. In: European conference on computer vision, Springer. 634–651
39. Taxonomy S (2016) Definitions for terms related to driving automation systems for on-road motor vehicles (j3016). Tech. rep., Technical report, Society for Automotive Engineering
40. Team OD (2020) Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet

41. Voigtlaender P, Krause M, Osep A, Luiten J, Sekar BBG, Geiger A, Leibe B (2019) Mots: Multi-object tracking and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 7942–7951
42. Weng X, Kitani K (2019a) A baseline for 3d multi-object tracking. 1907.03961
43. Weng X, Kitani K (2019b) Monocular 3d object detection with pseudo-lidar point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. 0–0
44. Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. 1703.07402
45. Xu Y, Ban Y, Alameda-Pineda X, Horaud R (2019) Deepmot: A differentiable framework for training multiple object trackers. arXiv preprint arXiv:190606618
46. Yan Y, Mao Y, Li B (2018) Second: Sparsely embedded convolutional detection. Sensors 18(10):3337
47. Yurtsever E, Lambert J, Carballo A, Takeda K (2020) A survey of autonomous driving: Common practices and emerging technologies. IEEE access 8:58443–58469
48. Zhang L, Li Y, Nevatia R (2008) Global data association for multi-object tracking using network flows. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition IEEE. 1–8
49. Zhang W, Zhou H, Sun S, Wang Z, Shi J, Loy CC (2019) Robust multi-modality multi-object tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2365–2374
50. Zhou X, Wang D, Krähenbühl P (2019) Objects as points. In: arXiv preprint arXiv:1904.07850
51. Zhou Y, Tuzel O (2018) Voxelnet: End-to-end learning for point cloud based 3d object detection. 4490–4499. https://doi.org/10.1109/CVPR.2018.00472

Authors and Affiliations

Carlos Gómez-Huélamo[1] · Javier Del Egido[1] · Luis Miguel Bergasa[1] · Rafael Barea[1] · Elena López-Guillén[1] · Javier Araluce[1] · Miguel Antunes[1]

Javier Del Egido
javier.egido@edu.uah.es

Luis Miguel Bergasa
luism.bergasa@uah.es

Rafael Barea
rafael.barea@uah.es

Elena López-Guillén
elena.lopezg@uah.es

Javier Araluce
javier.araluce@edu.uah.es

Miguel Antunes
miguel.antunes@edu.uah.es

[1] Department of Electronics, University of Alcalá (UAH), Madrid, Spain