Dual-branch CNNs for vehicle detection and tracking on LiDAR data

Víctor Vaquero*, Iván del Pino*, Francesc Moreno-Noguer, Joan Solà, Alberto Sanfeliu, and Juan Andrade-Cetto

***Abstract*—We present a novel vehicle detection and tracking system that works solely on 3D LiDAR information. Our approach segments vehicles using a dual-view representation of the 3D LiDAR point cloud on two independently trained convolutional neural networks, one for each view. A bounding box growing algorithm is applied to the fused output of the networks to properly enclose the segmented vehicles. Bounding boxes are grown using a probabilistic method that takes into account also occluded areas. The final vehicle bounding boxes act as observations for a multi-hypothesis tracking system which allows to estimate the position and velocity of the observed vehicles. We thoroughly evaluate our system on the KITTI benchmarks both for detection and tracking separately and show that our dual-branch classifier consistently outperforms previous single-branch approaches, improving or directly competing to other state of the art LiDAR-based methods.**

***Index Terms*—Deep convolutional neural network, vehicle detection and tracking, LiDAR, point cloud.**

I. Introduction

Autonomous vehicles are becoming a reality. Nowadays, level-3 automobiles are already present on our roads, allowed to circulate autonomously under certain circumstances but assuming that the human driver will take back the control if required [1]. Moreover, manufacturers have made clear their purpose of releasing fully automated level-5 agents in the near future, which will be able to circulate without any human supervision in all kind of scenarios.

In our opinion, the confluence of two factors has been essential for this fast progress. On the one hand, research advances in areas such as robotics, machine learning, and computer vision are endowing the systems with a high level of scene comprehension. On the other hand, new hardware and on-vehicle sensors are providing the community with enough data to develop new and robust algorithms as well as the ability to process them in real time.

However, there is still a long road until level-5 vehicles can drive freely in our cities. Real-world traffic situations raise very challenging scenarios that contain a great variety of elements like vehicles, cyclists, pedestrians, or even urban furniture. In this way, the ability to detect and consistently track those elements over time is of vital importance and a core module to elaborate further systems like cruise control, collision avoidance or vehicle platooning [2], [3], [4], [5].

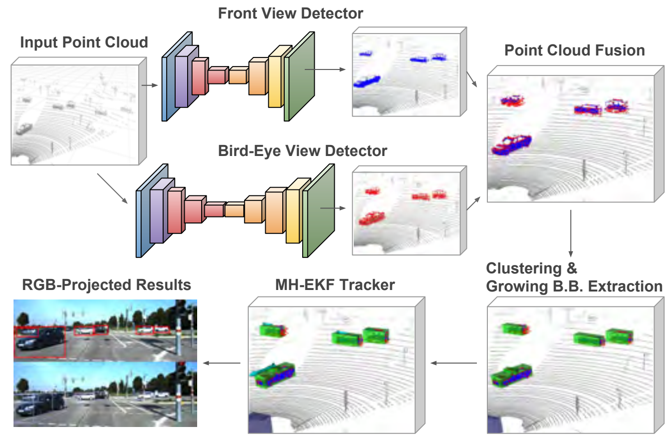


Fig. 1: Using solely LiDAR information, our dual-view deep convolutional architecture is able to detect and track vehicles in real driving scenarios. The presented approach uses two different representations of the 3D input point cloud that are fed into two independent CNN-based branches for per-point vehicle classification. The outputs are then fused in a single 3D point cloud and recursively clustered. A novel bounding box growing method that relies on the use of contextual information generates trustworthy observations that are then fed to a multi-hypothesis extended Kalman filter-based tracker.

With the advent of deep learning (DL), image-based scene understanding has experienced a remarkable boost, providing in-vehicle perception systems with strong capacities on tasks such as object detection, classification or semantic segmentation [6], [7], [8]. Nevertheless, optical cameras may fail to correctly capture the environment under certain conditions, such as abrupt changes of illumination (entering a tunnel, light flashes, etc.) or harsh weather conditions (heavy rain, fog, snow, etc.). Additional sensors are thus required to fulfill the need of robustness on autonomous vehicles, whether for providing complementary scene information to the existing algorithms or to act as a full backup system.

LiDAR sensors are especially suitable for this purpose since their performance independent of the scene illumination and are robust to harsh environmental conditions, while providing accurate spatial information. Despite the remarkable results on camera images, the adoption of DL-based methods for LiDAR data is far from the level achieved on image processing

*These authors contributed equally to this work: vvaquero@iri.upc.edu, ivan.delpino@ua.es.

V. Vaquero, F. Moreno-Noguer, J.Solà, A. Sanfeliu, J. Andrade-Cetto are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona, Spain. I. del Pino is with the Automation Robotics and Computer Vision Group (AUROVA), University of Alicante, San Vicente del Raspeig S/N, Alicante, Spain.

This work is supported by the EU project LOGIMATIC (H2020-Galileo-2015-1-687534), the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656), the ColRob-Transp project (DPI2016-78957-R AEI/FEDER EU), the EB-SLAM project (DPI2017-89564-P), and the FPU grant FPU15/04446.

tasks. Yet, recent works [9], [10], [11], [12], [13], [14], [15], [16], [17] are pointing at DL techniques as powerful tools to extract information from point clouds, expanding their applicability beyond optical cameras. However, those approaches commonly use additional RGB data, complex structures with 3D or sparse convolutions, or end-to-end systems yielding directly bounding boxes without allowing the introduction of prior knowledge.

In this paper, we present a fully working detection and tracking system for vehicles in driving scenarios based only on 3D LiDAR as shown in Fig. 1. In a preliminary version of this work [11], we segmented vehicles from a front projection of the LiDAR data with a deconvolutional neural network and used simple Euclidean clustering to extract bounding boxes that were later tracked trough time. We go beyond this initial work and significantly improve the system by introducing a dual-view deep-learning pipeline to segment vehicles from LiDAR information, along with several other novelties such as adaptive threshold recursive clustering, and a bounding box growing algorithm guided by contextual information. We perform extensive tests of our method on the Kitti benchmark [18] for both the detection and tracking tasks, quantitatively showing better performance in comparison to the state of the art. More specifically, our contributions are:

1) We present a novel dual-view deep learning architecture with specialized parallel branches to segment vehicles from two different 3D point cloud representations.

2) We devise a new deep model with customized fire modules that performs vehicle segmentation over a featured bird's-eye view representation of the 3D input LiDAR data. This representation preserves the geometrical features of the objects, providing invariance to its size, as they are represented on their real on-ground position.

3) We develop a novel fusion strategy to retrieve accurate BBs for the segmented 3D point cloud which has three main features: a) recursive Euclidean clustering with an adaptive threshold; b) a new confidence measure to fuse detections given the fact that false positives are uncorrelated between branches; and c) an algorithm to compute full-size BBs by expanding small ones towards occluded regions, thus resolving orientation ambiguities.

4) We introduce a novel bird's eye view evaluation of the tracker performance, which is more informative than the front image domain used on the Kitti tracking benchmark, as it is done over the X-Y ground plane; and

5) We extensively test our system both in the detection and tracking benchmarks of the Kitti dataset. We present a throughout ablation study for each task testing our capacities over different distances and perform fair comparisons showing how our dual-view deep-LiDAR method consistently outperforms previous approaches.

II. Related Work

Convolutional neural networks (CNNs) have demonstrated an outstanding performance solving image-based problems such as object classification [6], [19], detection [7], [20], [21], or semantic segmentation [8]. Despite this great success, the potential of these deep learning techniques has not yet been extensively deployed for analyzing 3D LiDAR point clouds. In this section we review the most relevant approaches proposed to detect objects with CNNs in such 3D sparse domain.

Deep convolutional models have been applied over 3D LiDAR point clouds as a way of learning useful features substituting hand-crafted ones as for example, in *Vote3D* [22]. A straightforward solution is to employ 3D convolutions, as in [23] for 3D vehicle detection. However, this implies a high and unnecessary computational cost due to the additional dimension and the sparse domain. To deal with the problem, sparse convolutions can be used on point clouds. In this way, [24] extends the work of *Vote3D* by replacing a support vector machine classifier with sparse 3D convolutions that act as voting weights for predicting the detection scores. Other recent works directly generate a structured space by subdividing the point cloud into voxels [13]. The group of points within each voxel is then transformed into a unified feature representation by a voxel feature encoding layer.

Very recently, some approaches are directly using the raw point cloud as input to deep architectures that are trained to directly get the desired ourputs. PointNet [25] applies a set of transformations and multi-layer perceptrons to generate global point cloud features which are then used for classification and segmentation tasks. The main drawback is that the method does not capture the local structures inherent to the points' metric space. Aiming to solve this issue, PointNet++ [26] proposes to recursively apply PointNet on nested areas of the input point cloud, learning local features with increasing contextual scales. The same ideas are shared in [12], which explores larger areas by extracting 3D frustums lifted from the corresponding bounding boxes predicted by a 2D CNN detector over RGB images. All these methods are often influenced by the point cloud density and can be considered as black-boxes containing ad-hoc steps that can greatly affect their performance, stability and human comprehension.

Notwithstanding, the most commonly adopted procedure is to pre-process the 3D LiDAR point cloud to obtain equivalent 2D representations from which to apply the well-known and optimized 2D CNN approaches. The outputs can be later fused and processed in order to achieve the final objective. For instance, [9] projects the point cloud to a front view representation encoding the range distance and height of each 3D point. They train a fully convolutional network to predict the *vehicleness* confidence of each point and regress the 3D bounding box of the containing vehicle, which increases the computational load of the method. Similarly, we have in the past segmented vehicles from a front view projection of the spherical LiDAR coordinates [11], [27] and then tracked them through time. In [28], the authors also favor a front view projection encoding normalized range distance, differentiated range distance (range difference for neighboring data points), and normalized height in three channels fed to a CNN for vehicle segmentation. Differently, other approaches such as BirdNet[29], TopNet [30] or RT3D [31], employ a bird's eye view projection of the point clouds, also known as zenithal view, and encode different features on each projected cell. Other further methods fuse different domains and combine

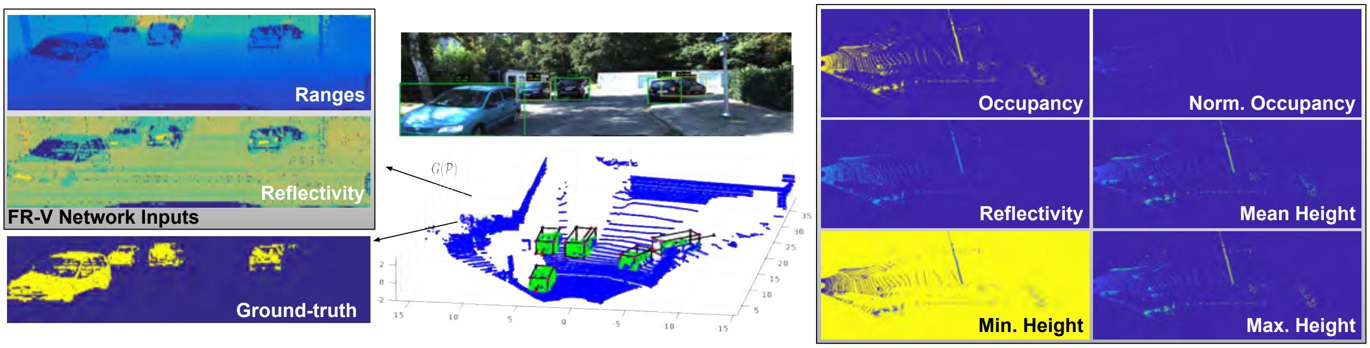


Fig. 2: We project each point cloud into two different view planes to obtain useful inputs for our deep CNN detectors: a front view with range and reflectivity channels (left), and a bird's-eye view with six channels encoding various occupancy, reflectivity, and height parameters (right). Ground-truth is generated projecting back the image-based Kitti tracklets over the 3D LiDAR data and selecting the points inside each bounding box. The RGB image is shown just for visualization purposes.

RGB images and LiDAR information. For example, [10] uses the LiDAR bird's eye view to generate 3D bounding box proposals, which are employed later to detect vehicles over the RGB image and the front point cloud projection using a region-based fusion network.

Our objective is to obtain a fully working and independent system for detecting and tracking vehicles in 3D with information from the minimum number of sensors. We chose to focus solely on LiDAR as these sensors are able to measure accurately the 3D in the scene and are more robust to changes of illumination and harsh weather conditions than cameras. Instead of using a bird's eye view for BB proposals as in [10], we devise a dual-branch LiDAR detector; one branch uses a novel featured bird's eye view of the 3D points that are processed through a specialized deep architecture, whereas the other branch employs a front view projection and a similar network to the one described in [11]. Each branch independently predicts the probability of each of its input points of belonging to a vehicle, after which we perform a late fusion step merging both segmented point clouds. Remarkably, our approach uses only LiDAR information and we do not make any use of RGB images during training nor inference, obtaining a full backup system suitable for safety purposes when cameras may fail.

III. Approach

The input is a point cloud $\mathcal{P} = \{q_1, \cdots, q_Q\}$, where each point $q_k \in \mathbb{R}^4$ contains the point's Euclidean coordinates with respect to the sensor center of projection and a measured reflectivity value. The system output is a list of tracked vehicle bounding boxes defined by their pose on the ground $\boldsymbol{\pi} = (x, y, z, \theta)$ and their bounding box parameters $\mathbf{d} = (w, l, h)$.

Our solution includes three steps: vehicle detection, bounding box extraction, and vehicle tracking. We formulate the vehicle detection task as a per-point classification problem in which we want to obtain the probability of each 3D point to either belong or not to a vehicle: $p(l|q_k)$, where l represents the labels $\{$*vehicle*, *not-vehicle*$\}$.

We define two different projections of $\mathcal{P}$: a front view $\mathbf{I}_{FR}$; and a bird's-eye view $\mathbf{I}_{BE}$. Our learning objective is to model the two functions $\mathcal{F}_{FR} : (\mathbf{I}_{FR}, \mathbf{Y}_{FR}) \to \hat{\mathbf{Y}}_{FR}$ and $\mathcal{F}_{BE} : (\mathbf{I}_{BE}, \mathbf{Y}_{BE}) \to \hat{\mathbf{Y}}_{BE}$, where $\mathbf{Y}_{FR}$ and $\mathbf{Y}_{BE}$ are the ground-truth binary masks indicating whether or not each projected point belongs to a vehicle and $\hat{\mathbf{Y}}_{FR}$ and $\hat{\mathbf{Y}}_{BE}$ contain the estimated *vehicleness* probability map for the two projection planes. We learn these functions $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ using two independent deep convolutional neural networks.

We next reconstruct an annotated point cloud from the estimated probability maps and generate 3D bounding boxes for each segmented vehicle via recursive clustering. A third probability map $\mathbf{S}_{BE}$ is then created in the bird's eye view to encode the probability of each cell to either be occupied, free, or occluded, which is used in turn to grow the boxes to standard vehicle sizes giving preference to occluded regions.

Our last objective is to track these generated bounding boxes through time using a number of multi-hypothesis extended Kalman filters, one per tracked vehicle, as detailed in [11]. A general sketch of our framework is shown in Fig. 1.

In the next sections, we present these three steps of our full system. Section IV shows the Deep LiDAR-based vehicle detection phase. Then, Section V-D details the bounding box extraction procedures. Finally, Section VI describe our tracker.

IV. Vehicle Detection

A. Point Cloud Pre-processing

To overcome common issues with the unordered and variable number of measurements existing in LiDAR data, we first project each point cloud into two different view planes, a front view $\mathbf{I}_{FR}$ and a bird's-eye view $\mathbf{I}_{BE}$ as shown in Fig. 2.

$\mathbf{I}_{FR}$ is obtained using the same front view projection described in [11], which models the Velodyne HDL-64 geometry and arranges the 3D input point cloud $\mathcal{P}$ into a 3D array such that $\mathbf{I}_{FR} \in \mathbb{R}^{H \times W \times C}$. Initially, each Cartesian point (x, y, z) is transformed to spherical coordinates (ϕ, θ, ρ). The elevation angle θ represents each of the $H = 64$ horizontal laser beams, covering the LiDAR vertical resolution from -24.5 to $+2.0$ degrees with variable resolution $\Delta\theta$ of $1/3$ degrees for the upper half of laser beams and $1/2$ degrees for the lower half. Even though the laser point cloud has a large horizontal field of view, we are forced to filter the point cloud in the

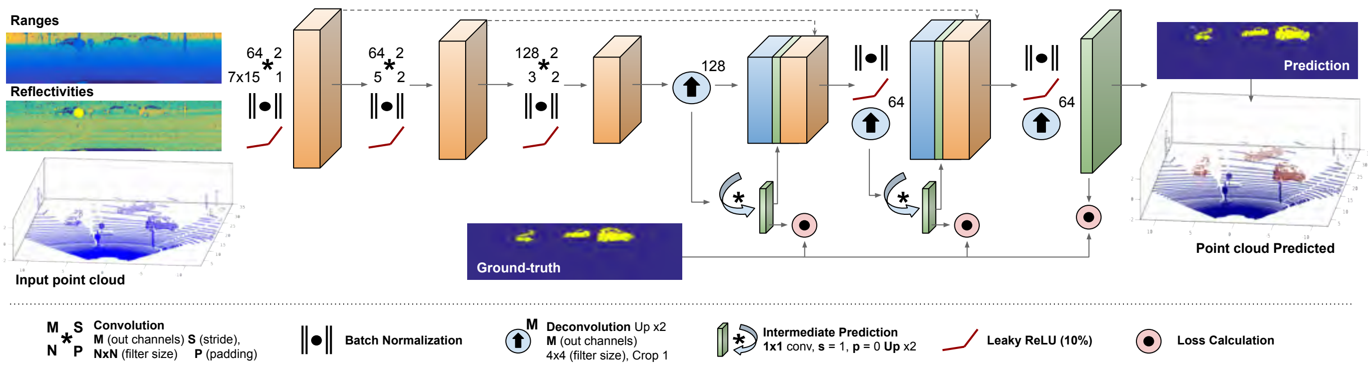


Fig. 3: Our front view network employs an encoder-decoder architecture with convolutional and deconvolutional blocks followed by batch normalization and Leaky-ReLU non-linearities. The first three blocks generate rich features controlling, according to our vehicle detection objective, the size of the receptive fields and the feature maps generated. The next three deconvolutional blocks expand the information. Feature maps from both parts of the network are also concatenated providing more gradient stability, which results in better learning performance. During training, three losses are computed at intermediate resolutions yielding to low-resolution predictions that are also concatenated in order to obtain a better final solution.

range $\phi \in [-40.5, 40.5]$ degrees because the Kitti dataset has only annotated ground truth for those objects inside the camera field of view. The resulting cropped point cloud is then discretized according to an azimuth step of $\Delta\phi = 0.18$ degrees, as specified by the sensor manufacturer, resulting in a map of width $W = 448$. The third dimension in our front view map represents the $C = 2$ channels, which store range ρ and reflectivity r values. For multi-echoes, only the closer detection is considered and missing points in the projected area, e.g. points without collision or absorbed by dark areas, are tagged as invalid. The left frame in Fig. 2 shows a sample of the created front view map, and its associated ground truth.

The bird's eye (zenithal) view $\mathbf{I}_{BE}$ is obtained by cropping the original point cloud in the volume ($x \in [3, 63], y \in [-25, 25], z \in [-2.1, 10]$), which maps an area of 60×50 meters in front of the LiDAR sensor. This design decision was chosen after carefully observing that roughly 95% of the annotated vehicles in the ground truth data are within these margins. Inspired by [10], we next project this cropped point cloud to the ground into a 2D cell grid with a resolution of 0.1 meters. Thus, we obtain a bird's eye view $\mathbf{I}_{BE} \in \mathbb{R}^{H' \times W' \times C'}$, where $H' = 600$, $W' = 500$, and $C' = 6$, accounting for six different features: 1) a binary occupancy term with zero value if no points are projected onto that cell and one otherwise; 2) an absolute occupancy term, counting the number of points that fall into that cell; 3) the mean reflectivity value of the points on the cell; and 4, 5, and 6) the mean, minimum and maximum height values calculated over the set of points projected onto the cell. The right frame in Fig. 2 shows a real sample of such six-dimensional feature map.

B. Ground Truth Generation

We obtain ground truth vehicle masks for the front view $\mathbf{Y}_{FR}$ and the bird's eye view $\mathbf{Y}_{BE}$ using the Kitti Tracking tracklets, which provide real 3D-oriented bounding boxes in the camera frame. These tracklets are transformed to the LiDAR frame and the 3D points that lay inside each bounding box are labeled as vehicle while the rest of the point cloud remains with the background label.

C. Network Architecture for Classification

To learn the functions $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ we formulate the task as a per-point classification problem, which in turn can be considered as a binary semantic segmentation one, i.e., each *pixel* in the maps $\mathbf{I}_{FR}$ and $\mathbf{I}_{BE}$ must be classified as either belonging or not to the vehicle class. With that purpose in mind, we follow our initial proposal [11] but go one step further and setup a dual-branch scheme with two parallel CNNs whose results are fused in a later step.

The learning problem is solved for both the front and bird's eye view branches in a supervised manner with end to end back propagation guided by a class weighted cross entropy loss function [32], defined as:

$$\mathcal{L}^{WCE}(\mathbf{I}^n, \mathbf{Y}^n) = -\sum_{h,w,c}^{H,W,C} \omega(Y^n_{h,w}) Id_{[Y^n_{h,w}]} \log(\mathcal{F}(I^n, Y^n)_{h,w,c}), \tag{1}$$

where $\mathbf{I}^n, \mathbf{Y}^n$ are respectively the n-th training and ground truth maps, ω is a class imbalance weighting function computed from the training set statistics as the inverse ratio between the vehicle and background classes, and *Id* is an index function that selects the predicted probability associated to the expected ground truth class.

We further solve the proposed classification problem in a multi-scale manner by introducing intermediate loss functions at different resolutions. This approach inserts valuable gradients at middle levels, guiding the network to a correct solution faster. Hence, for each branch we compute the final loss $\mathcal{L}$ as

$$\mathcal{L}(\mathbf{I}^n, \mathbf{Y}^n) = \sum_{m=1}^{M} \lambda_m \mathcal{L}^{WCE}(\mathbf{I}^n_m, \mathbf{Y}^n_m) \tag{2}$$

where m represents the multi-scale step in which the loss functions are computed and λ_m are regularization weights for such loss at each resolution.

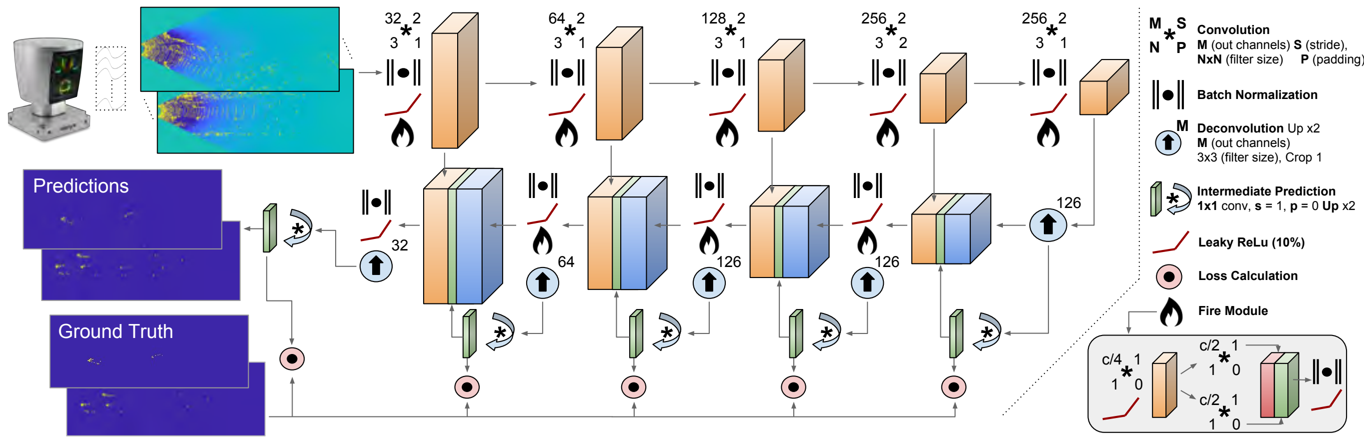


Fig. 4: Our bird's eye view network is a refined encoder-decoder architecture. As in this case the inputs are larger than in the front view case, we go further with reducing the feature maps and apply five contractive and five expansive levels. To get richer features at each level, we insert customized fire modules. These modules first reduce the number of feature channels, then apply two different sets of convolutional filters on them, and finally concatenate the results obtaining richer high and low-frequency features. Intermediate losses are also computed in this network, merging predictions at different resolutions.

We propose a contractive-expansive architecture which allows for a good embedding of the vehicle classification features. As this design could suppose a bottleneck in the flow of information, we additionally introduce skip connections concatenating equivalent feature maps between both contractive and expansive parts. These shortcuts help the learning process by building stronger features and by back-propagating purer gradients from the upper parts to the lower layers.

Front view network. The front view representation can introduce geometry distortions representing objects differently according to their position in the scene (for example, far objects are represented smaller), so we need to pay special attention to some key network designs. Initially, for this front view per-pixel classification task we employ the deconvolutional architecture shown in Fig. 3, which is similar to the one proposed in [11]. As key design decision, we observed that, independently of the area occupied on the view, vehicles in this view are usually represented twice wider than higher, and therefore we chose to design the initial convolutional filter sizes following this relation to be 7×15. Furthermore, we impose in the first convolutional layer a vertical vs horizontal stride ratio of 1:2, in order to obtain more tractable intermediate feature maps by reducing the size imbalance of $\mathbf{I}_{FR}$ (64 vs 448).

Bird's eye view network. Having a cell resolution of 0.1 meters, vehicle bounding boxes in $\mathbf{I}_{BE}$ occupy an average of 800 cells. This implies that, over the full training set, only a little more than the 10% of the cells per frame are occupied by a vehicle. Detecting such small areas is still a challenging problem for deep neural networks, which made us to design the novel architecture shown in Fig. 4 for this domain.

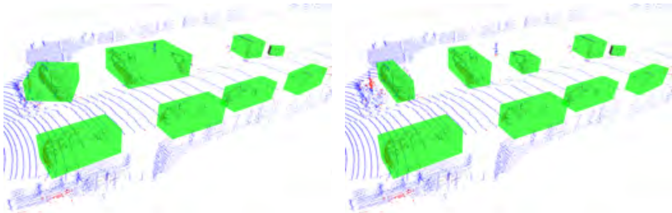
Ideally we would like our network to be small enough to process frames fast but at the same time accurate on details. To keep the number of network parameters small while still providing high accuracy, we employ a customized version of the established deep convolutional 'fire modules' [33], [34], [35] in our architecture. As shown in the bottom-right area of Fig. 4, our modified fire modules first apply a convolution to reduce the number of feature maps (channels of the input tensor). Then two parallel branches apply convolutions with two different filter sizes. The results are then fused obtaining robust features with local and context-aware information by using less parameters. We insert fire modules after each resolution variation of the feature maps, which accomplishes the strategy presented in [33] of downsampling late in the network, so that the convolution layers can retain large activation maps using fewer computations to process such broad areas.

V. Bounding Box Extraction

The above-mentioned architecture provides a powerful per-point vehicle class identification. However, we need to cluster these identified points into vehicle hypotheses in order track them. In this section we describe a mechanism to obtain bounding boxes from these vehicle hypotheses. The method is summarized in four steps: a) the fusion of the two vehicle identification results coming from the two deep neural network classifiers into a single annotated point cloud, b) the use of a recursive clustering algorithm that uses an adaptive threshold to group points into individual vehicle hypotheses, c) the extraction of bounding boxes, and d) the growth of individual bounding boxes using contextual information around the cluster to assign growing preference to occluded areas.

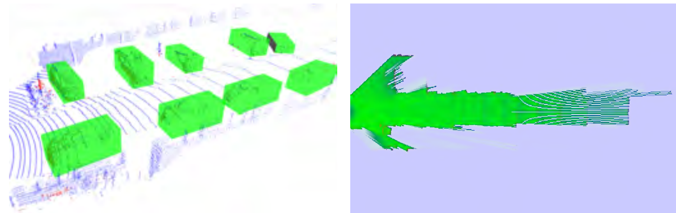
A. Fusion of Results from the Two Classifier Branches

In the case of the front view predictions $\hat{\mathbf{Y}}_{FR}$, we recover the 3D location (x, y, and z) of each classified vehicle point, from the azimuth and elevation angles ϕ and θ (encoded by the row and column indexes), and the range value ρ. The original 3D point cloud can be recovered from this representation by just inverting the initial projection (ϕ, θ, $\rho \rightarrow x$, y, z). To minimize the possible distortion introduced, we use a k-d tree



(a) Standard clustering (b) Recursive clustering

Fig. 5: Bounding Box extraction. (a) Euclidean clustering used on [11]. (b) Improved results of this paper using recursive clustering with dynamic threshold applied on over-sized elements.



(a) Bounding box growing (b) Free space on probability map

Fig. 6: Bounding Box growing. (a) Bounding boxes grown on small elements up to standard vehicle sizes (see the same top-right three boxes of Fig. 5), and (b) obtained probability map for each cell being free, occluded or occupied.

to efficiently find the closest neighbor in the original point cloud so that recovering the *real points sensed* by the LiDAR.

In the case of the bird's eye view, we recover the lower, middle and higher 3D points of each vehicle-annotated cell in the map $\hat{\mathbf{Y}}_{BE}$. These three 3D points are joined to the above-listed set of points from the front view into a fused point cloud, preserving the originating view id on each point. Given the use of a Softmax function in the final layer of the classifier, it is safe to use a *vehicleness* probability threshold of 0.5, for selecting the cells to be considered as positively classified in the above-mentioned cases.

B. Recursive Euclidean Clustering

In [11], we clustered the results of our front view detector using standard Euclidean clustering. Since vehicle context information was not considered in that case, the solution had the drawback that depending on the distance threshold used, one might end up joining together two or more vehicles or, on the contrary, over-clustering a single vehicle due to the inherent sparsity of the LiDAR measurements.

In contrast, we now propose a recursive clustering algorithm that gradually reduces the distance threshold for the clusters that exceed the maximum dimensions of a standard vehicle. The clustering step ends when all the extracted clusters have standard vehicle sizes or when the distance threshold reaches a minimum value, in which case, the cluster is discarded. Figs. 5a and 5b exemplify the importance of this step. We apply an statistical outlier removal algorithm over the obtained clusters to assure that no outlier points would distort the shape of the object when fitting the final bounding box.

Cluster Confidence. By definition, both neural network classifiers produce uncorrelated results. We can therefore obtain a confidence value for each detection by analyzing the ratio of contribution from each classifier on the fused point cloud:

$$\eta = \frac{min\{Q_{BE}, Q_{FR}\}}{max\{Q_{BE}, Q_{FR}\}}, \quad \eta \in [0, 1] \tag{3}$$

where Q_{BE} and Q_{FR} indicate the number of points that each classifier identifies as vehicle point in a particular cluster in the fused point cloud. For a fair comparison, Q_{FR} is obtained after reprojecting the point cloud to the bird's eye view. This η confidence measurement is high when a representative proportion of predictions from both classifiers exists in the same cluster. We will also use this η later to initialize vehicle tracks and to compute the detection score needed when comparing our system with other methods in the state of the art.

C. Bounding Box Extraction

Our tracking module uses a 2D-ground representation of the world, which is convenient when dealing with road vehicles since it reduces the degrees of freedom to just three. Instead of using the not directly observable cluster centroid to perform the tracking, we use a rectangular model that allows us to apply car-like kinematic constraints and to avoid false motions produced by changes in the point of view. Initially, we project the clustered points into the ground plane, keeping for each azimuth angle the points with shorter range values to establish the object perimeter. Since the Kitti Tracking benchmark considers only 2D bounding boxes in the RGB image plane, we also save the cluster height h and z coordinate to further compare our method against others.

Along with z-axis related variables (h and z), we need to estimate the bounding box 2D pose $\boldsymbol{\pi} = (x, y, \theta)$ and dimensions $\mathbf{d} = (w, l)$. Instead of searching in a five-dimensional space, we assume that the outlier rejection process has been successful, so we can reduce the problem to just one variable, searching for the orientation θ^* that produces the best (in Maximum Likelihood terms) minimum-sized oriented box containing the whole cluster. As the search space is small because the same rectangular fitting is obtained when adding multiples of $\frac{\pi}{2}$, we use a uniform sampling process to find the value that minimizes the mean quadratic error between predicted points (generated using the rectangular model and the parameter under evaluation) and observed ones, given by the real LiDAR sensor.

Occupancy Map. In contrast to [11], in which vehicle clusters would only grow in the tracking step by using information about the viewpoint change, we here introduce a more efficient manner to grow vehicle bounding boxes in the detection step by taking into account occlusion information.

We generate an extended occupancy map $\mathbf{S}_{BE} \in \mathbb{R}^{H' \times W'}$ that has the same size as the bird's eye view map and registers the probability of a cell of being either occupied, free, or occluded in the following way. As each point from the original point cloud is an evidence of occupancy, we extract a value

Q_h by counting the points that hit each cell. Furthermore, each processed point is backprojected to the sensor center, contributing this ray for counters Q_f on the cells along this line. We also store for each cell the lowest height value of either a point hit or any ray passing through that cell (z_m).

The probability of a cell for being occluded can therefore be computed as the ratio between the lowest observed point on that cell to the ground and the expected vehicle height h

$$p_o = (z_m - z_g)/h \tag{4}$$

We use z_g as the ground value computed from the LiDAR mounting point –known by calibration– or alternatively, the minimum point registered in a cell z_m, if it is lower. Finally, the probabilities of a cell to be either occupied (p_h) or free (p_f) are computed from the extracted point statistics as

$$p_h = (1 - p_o) \cdot (Q_h/(Q_h + Q_f)) \tag{5}$$
$$p_f = (1 - p_o) \cdot (Q_f/(Q_h + Q_f)) \tag{6}$$

D. Bounding Box Growing and Orientation Inference

We want to grow the vehicle bounding boxes $(\boldsymbol{\pi}, \mathbf{d})$ preferably along either occupied or occluded regions, and to guarantee that they do not grow along free areas. To this end we compute a bounding box cost C as the average p_f of all cells within the box. To grow the box we start from the box corner closest to the sensor and a minimum vehicle size and iterate along the two opposing sides of it. When the vehicle orientation is not well defined by the box dimensions $\mathbf{d}$ we maintain two perpendicular hypotheses, grow them accordingly and compute their normalized relative costs $C_a \in [0, 1]$ and $C_b = 1 - C_a$. The bounding box growing process consists on expanding the box dimensions in small increments until its mean cost stops decreasing, until the maximum box dimensions are reached or until a collision with another box is detected. We evaluate our confidence on each bounding box in terms of the relation between the costs of the individual hypotheses as follows

$$\nu_a = \frac{1}{2}(1 - C_a + C_b) \quad ; \quad \nu_b = \frac{1}{2}(1 - C_b + C_a) \, . \tag{7}$$

Once the growing process ends, we keep the box with largest confidence value ν. Finally the detection score S required to compute the average precision is computed as:

$$S = \nu \cdot \eta \cdot (1 - \epsilon) \, , \tag{8}$$

where the fitting error ϵ (defined in Section V-C) is saturated to one meter to avoid negative scores. However, this situation is very unlikely because ϵ is usually in the order of centimeters, or few decimeters in the more adverse cases.

VI. Vehicle Tracking

For the tracking task, we start from the 2D multi-hypothesis extended Kalman filter approach presented in [11], and include two new improvements: 1) we make use of the cluster confidence term η to decide whether new tracks should be initiated, 2) instead of considering equal probabilities for each alternative hypothesis, we use the bounding box confidence measure term ν to establish the prior probabilities.

We follow standard statistical tests (Mahalanobis distance) for performing cluster-to-vehicle track data association. When a cluster is not associated to an existing track, a large cluster confidence value η would indicate that a new track hypothesis can be initiated. On the contrary, track observations with low cluster confidence values are only used to track existing hypotheses. When a vehicle track is initiated, we maintain two hypotheses in the filter, one for each possible perpendicular orientation θ and $\theta_\perp$. While [11] uses equal weights (i.e. $\omega_\theta = \omega_{\theta_\perp} = 1/2$) for both hypothesis, here we use the bounding box confidence term ν to differently weight their contributions: $\omega_\theta = \nu$, and $\omega_{\theta_\perp} = 1 - \nu$, where $\omega_\theta \geqslant \omega_{\theta_\perp}$ because ν is the confidence value associated with the hypothesis with lower cost (as shown in equation 7).

Once the tracker disambiguates the track to follow thanks to the vehicle motion updates, it will settle with the one having strongest likelihood of correct orientation. This keeps the valid hypothesis alive even at stops or in poor detection scenarios (occlusions, etc). In the end, the bounding box confidence ν is only pertinent at the beginning of each track, when we have absolutely no evidence of present or past velocity.

VII. Experiments

The proposed system has been evaluated in both the Kitti vehicle detection and Kitti tracking benchmarks. For each of these tasks separately, we show results evaluated with three configurations of our dual-branch deep learning-based classifier: front view only, bird's eye view only, and using both classifiers as described in the paper. Additionally, for each system configuration, we provide an ablation study and show the performance with and without our bounding box growing algorithm. Moreover, we also include results comparing our deep-models with an *oracle* on the per-point classification task and their impact on the vehicle detection and tracking steps.

Compared to our previous work [11] we significantly improve the overall system performance, which we attribute to the confluence of two factors: a) the inclusion of the bird's eye view classifier and fusion approach with the cluster confidence term that allows us to remarkably reduce the number of false positives, boosting system performance to a level comparable to the results obtained when using the *oracle* detector with perfect per-point classification; b) the pipeline including recursive clustering and bounding box growing, which significantly decrease the number of sub-and over-clustering situations, producing better and more accurate bounding boxes.

Dataset. For learning purposes we use the training subset of the Kitti tracking dataset, which contains 21 driving sequences and a total of 8000 Velodyne HDL-64 annotated scans. To validate our approach separately for both the detection and tracking tasks we use the first three sequences in this dataset, that account for about 15% of the total vehicle-class points.

Network training. Both the front view and the bird's eye view classifiers are initialized with He's method [36] and we use Adam optimization with the standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for training. In order to preserve the geometry properties of the driving scene we only augment the dataset performing horizontal and vertical flips respectively in the

TABLE I: Ablation Study for the Vehicle Detection Task - Validation Set (IoU 0.7) (%).

Grow.BB	Method	2D Front (mAP 2D)			2D BE-V (mAP BEV)			3D detection (mAP 3D)		
		Easy	Med.	Hard	Easy	Med.	Hard	Easy	Med.	Hard
✗	FR-V	20.13	18.69	17.63	42.31	29.67	30.35	02.32	02.37	01.67
	BE-V	26.26	24.53	23.08	64.86	49.02	41.48	07.68	08.51	07.68
	Fusion	51.28	42.99	39.07	65.15	50.80	42.52	12.22	13.19	11.45
✓	FR-V	47.41	40.67	40.44	69.37	60.41	52.24	13.44	13.38	12.87
	BE-V	52.35	49.22	45.00	77.75	60.97	61.33	15.84	18.61	19.67
	Fusion	72.11	61.41	54.59	79.73	62.29	62.37	22.45	25.36	24.74
✓	'*Oracle*' FR-V	54.08	44.54	45.09	79.96	71.26	62.61	28.52	27.97	28.33
	'*Oracle*' BE-V	51.26	42.47	43.14	89.33	70.98	71.16	27.39	26.97	24.24
	'*Oracle*' Fusion	70.85	58.37	52.06	89.10	71.48	71.51	42.11	40.40	35.57

front view and bird's eye view inputs with a 50% chance. No further augmentation is done, We train each network independently on a single Nvidia 1080Ti GPU for $400,000$ iterations with a batch size of 10. The learning rate is fixed to 10^{-3} during the first $150,000$ iterations after which, it is halved every $50,000$. We set the regularizator ω to 25 in the front view and to 1000 in the bird's eye view classifier, attending to the ratio between classes on each domain. In both detector schemes, the multi-resolution loss regularizers λ_r are set to 1, assigning equal importance to each resolution.

Bounding Box Configuration. For the recursive clustering algorithm we start with an initial Euclidean threshold of $1.0\ m$ that is gradually decreased in steps of $0.1\ m$ for over-sized clusters (more than 2.2 meters width or 5.0 meters length) up to a minimum value of $0.1\ m$. If the minimum threshold is reached and the cluster is still over-sized, the observation is discarded. In the same way, clusters with less than 10 points or radius smaller than half a meter are also discarded. Prior to computing the bounding box fit, a statistical outlier rejection algorithm available in the Point Cloud Library [37] is used with a 1% of cluster points as neighbours and a standard deviation multiplier threshold of 0.5.

Growing Bounding Box Parameters. The minimum width and length value for a grown bounding box is set to $1.6\ m$ and $3.4\ m$ respectively, and the maximum length after the expansion (in increments of $0.1\ m$) is set to $3.8\ m$.

A. Vehicle Detection Results

Table I presents a thorough evaluation of our vehicle detector over the proposed validation set. Three variants of the classifier architecture are studied, a front view only (FR-V), a bird's eye view only (BE-V), and the fused architecture presented in this paper (Fusion). For each variant, we also evaluate the system with and without the growing bounding box module (Grow.BB). Additionally, we include results of the vehicle detection task over bounding boxes with perfect per-point classification (*Oracle*). At the light of the results, we can observe how BE-V alone produces better results in comparison to FR-V. This is mainly because pixel classification errors in the FR-V obtain a larger number of false positives than in the BE-V, which has a more uniform distribution of the points in the scene. Furthermore, we see how our approach for combining both projections boosts the results, producing

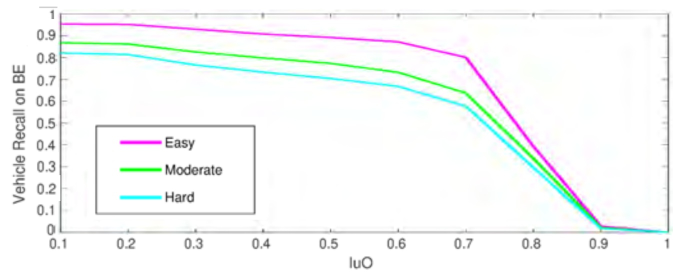


Fig. 7: Recall obtained by varying the IoU threshold on steps of 0.1. With an IoU of 0.5, we get approximately 80% of vehicle recall for moderate difficulty in the bird's eye view.

consistent gains in all the three Kitti difficulty levels. It is also evident how the growing bounding box module consistently helps on improving the results, which can be clearly appreciated in the 3D detections. Finally, observing the little difference of our full system against the *oracle*, we can state that our approach successfully solve the per-point classification problem. It can also be appreciated that on the FR-V projection of the point cloud some distortions are introduced that affect the *oracle* prediction, which is why the introduction of the BE-V projection is of great help.

We additionally analyze the Average Precision (AP) using different IoU thresholds in order to obtain better detection recall, which we consider is of most importance for autonomous driving. As shown in Fig. 7, our detector is able to locate more than 80% of moderate difficulty vehicles setting an IoU of 0.5.

We present in Table II results comparing our method with others in the state of the art for over the bird's eye view and 3D metrics both with an IoU value of 0.5. We can observe that our method outperforms MV, VeloFCN and BirdNet in almost all the difficulties for the bird's eye view detection, and performs favorably with respect to VeloFCN for 3D detection at this IoU value. We want to remark that the 3D detection mAP metric is an evolution of the standar 2D mAP, and may not be representative as a good 3D metric. IoU calculated over 3D volumes behaves more aggressively than over 2D planes, and a little height deviation over 2 bounding boxes could result on a wrong measurement. This is clearly appreciated in when comparing Table II and Table III and explaining the large difference obtained along this metric in both tables.

We tested our system on the online Kitti Detection bench-

TABLE II: Car Detection Results on Val. Set (IoU 0.5) (%).

Method	BEV detection (AP BEV)			3D detection (AP 3D)		
	Easy	Moder.	Hard	Easy	Moder.	Hard
Ours (FR-V)	81.29	80.62	71.85	46.17	47.38	43.01
Ours (BE-V)	**90.61**	81.25	72.30	55.92	52.92	52.90
Ours (FUSION)	90.31	**81.42**	72.40	71.73	62.19	54.97
MV(BV+FV) [10]	86.18	77.32	76.33	95.74	88.57	88.13
VeloFCN [9]	79.68	63.82	62.80	67.92	57.57	52.56
BirdNet [29]	90.43	71.45	71.34	88.92	67.56	68.59

mark, after carefully checking that there exists no correspondences between the Tracking training dataset (used for our training) and Detection testing set. Results from the Kitti website are presented in Table III. We can observe that although our approach is not focused on the detection task, it directly competes with other established works using only LiDAR information. For the 2D front detection, we outperform [31] and with [30]. We observe that there is room for improvement on the 2D bird's eye view case, which may be a matter of increasing our cell resolution as [29] and [30] do, so that our detected bounding boxes would overlap better. Notwithstanding, on the 3D detection task it can be seen that our method is well balanced due to our dual-branch architecture. In this hard problem, we obtain much better results than [29] and [30] on the three difficulty levels. Comparing in this domain with [31], we can conclude that although we performed better than this method in the 2D front case, we are penalized by the performance in the bird's eye view case, which indicates a future avenue for improvement.

B. Vehicle Tracking Results

We further analyze the performance of our system for bounding box extraction and tracking with a new ablation study with and without the deep detectors. Table IV shows the tracking results obtained over the validation dataset using either our deep-classifier or an **oracle** to compute the point classification step. We present success rates (as percentages) for the mostly tracked (MT), partially tracked (PT), and mostly lost (ML) tracking performance metrics. In addition, we show the precision and recall values, and the commonly used multi-object tracking accuracy (MOTA) metric [38].

As in the previous experiments, three detection modalities are compared in Table IV, front view (FR-V), bird's eye view (BE-V) and fusion. The *oracle* simulates perfect vehicle per-point classification, and thus establishes an upper bound for the learning task which helps us analyze the effect of the classifier performance in that of the tracker. Obtaining results close to those obtained when using the *oracle*, is an indicator of the resilience of our tracking module to the errors of the classifier.

The Kitti Tracking benchmark evaluates only on the front image plane (FR in the table). We consider this as a drawback, since this projection does measure 3D accuracy as well as does not represent the real geometry of the scene, introducing distortions. Therefore, as a novel contribution in this article, we include a new evaluation of the Tracking task in the bird's eye domain (BE in table) using the IoU of the ground-truth and tracked boxes directly on the $X-Y$ plane. This is clearly

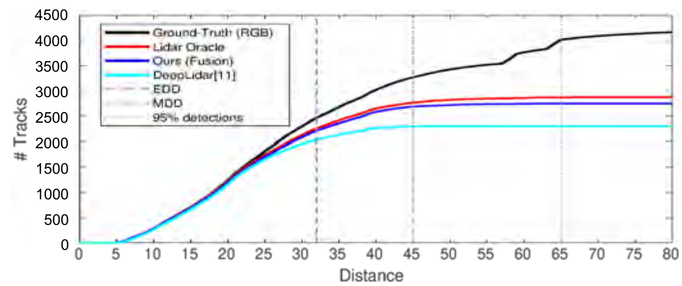


Fig. 8: Total number of true positives (TP) evaluated using different maximum distances. We observe that: 1) Our system obtains almost human performance level in near field. 2) Comparing to [11], our new method boosts the performance to almost reach the *oracle* level. 3) The bigger differences with the human annotators are produced by far vehicles, where the LiDAR information is much more scarce.

a better way of evaluating real world vehicle trackers, as it does not lose any 3D context of the scene.

Analyzing the results, we can observe how our dual-branch deep-classifier produces similar results in all tracking metrics than an *oracle* classifier. The improved performance of the presented detection approach is of great importance for the tracker, as it reduces the number of false positives that are fed to the tracker. Furthermore, this allows us to reduce the cluster size thresholds obtaining also less false negative detections.

Final results are shown in Table V over the online Kitti Testing benchmark using 2D bounding boxes in the RGB image plane. As far as we know, we are the first entry in this benchmark that performs the task using only LiDAR data which makes it harder to fairly compare our results with other approaches. Therefore, we compare the obtained metrics with our previous work [11] that also uses only LiDAR.

From Table V, we can extract several conclusions. Firstly, comparing both FR-V methods we see how the additional improvements introduced in this paper boost the final tracking performance in all the metrics. Furthermore, our fusion approach drastically beats the previous method [11] in all the metrics. We can also note that the number of false positives has been reduced, which is reflected in the precision improvement that evolves from 63.8% to 83.0%. Moreover, we remarkably increase the recall without producing undesired effects in the precision metric, which is directly translated into a more than double MOTA improvement.

We perform a final experiment to gain insights about the limits on detection performance that our system can handle, which is shown in Fig. 8. Here we identify the number of ground truth vehicles in the validation dataset that are within any given distance to the sensor, and also the number of true positives identified by our system as a function of the distance to the sensor (continuous lines in the plot). With this, we can define two metrics for performance in terms of the distance from the sensor: a) the '*effective detection distance*' (EDD), defined as the maximum distance at which the system recall $(TP/(TP+FN))$ is above 90%; b) the '*maximum detection distance*', a less restrictive metric to obtain the distance at which at least a third of the new appearing vehicles are correctly tracked. To calculate this second measurement, we

TABLE III: Vehicle Detection Task - Testing Set Comparison (IoU 0.7) (%).

Method	2D Front (AP 2D)			2D BE-V (AP BEV)			3D detection (AP 3D)		
	Easy	Med.	Hard	Easy	Med.	Hard	Easy	Med.	Hard
RT3D [31]	50.33	39.69	40.04	56.44	44.00	42.34	23.74	19.14	18.86
BirdNet [29]	78.91	57.02	55.08	76.88	51.51	50.27	13.53	09.47	08.49
TopNet-HighRes [30]	58.04	45.85	41.11	67.84	53.05	46.99	12.67	09.28	07.95
Ours Fusion	60.92	46.50	41.59	45.90	34.53	31.83	15.26	12.50	11.14

TABLE IV: Vehicle Tracking Task - Validation Results over standard Front and our novel Bird's Eye view Domain (%).

Evaluation		Ours			Oracle		
		FR-V	BE-V	Fusion	FR-V	BE-V	Fusion
FR	MOTA	15.4	33.6	36.7	38.5	38.2	43.3
	Recall	66.6	65.7	66.7	68.6	67.2	69.0
	Prec.	65.9	76.7	78.7	80.1	79.9	82.5
	MT	47.3	52.7	49.0	50.0	46.3	52.7
	PT	37.3	30.0	34.5	37.2	40.0	33.6
	ML	15.5	17.3	16.3	12.7	13.6	13.6
BE	MOTA	-14.5	12.0	13.8	29.2	28.3	29.5
	Recall	56.4	57.7	58.3	64.3	62.8	62.8
	Prec.	50.2	62.5	63.7	73.4	72.3	73.2
	MT	36.4	35.4	37.3	45.4	41.8	45.5
	PT	40.9	40.9	40.9	37.3	40.0	37.3
	ML	22.7	23.6	21.8	17.2	18.1	17.2

TABLE V: Vehicle Tracking Task - Testing Set Evaluated over the 2D front Image Plane (%).

Testing	Prev. [11]	Ours		
	FR-V	FR-V	BE-V	Fusion
MOTA	15.5	21.9	36.3	39.7
Recall	55.4	62.3	62.7	63.6
Prec.	63.8	69.0	78.9	83.0
MT	18.5	26.3	32.3	29.5
PT	52.2	59.3	52.0	54.6
ML	29.4	14.3	15.7	15.8

set an incremental recall metric $(\Delta TP/(\Delta TP + \Delta FN))$ that computes the recall variation produced by the increase on the distance threshold. The plot also shows the distance point at which 95% of the ground truth labels are visible.

Fig. 8 shows that system performance depends heavily on the distance from the vehicle to the sensor. We find our '*effective detection distance*' (EDD) at approximately 32 meters. Up to here our system performs close to the human level (represented as the ground truth of annotated objects). Going further, at approximately 45 meters we find our '*maximum detection distance*' (MDD). Although at this point the number of false negatives (FN) is increased, our system is still able to detect a significant part of the distant vehicles. We can clearly see here the benefits of our multi-branch approach in comparison to [11], as we almost match the results obtained with the *oracle* detector. Going beyond the system maximum distance we find at around 65 meters the point at which the 95% of the manual annotations have already appeared. However, it can be appreciated how the capabilities of our LiDAR only system are limited at these distances, in which laser observations become more sparse as the distance grows.

VIII. Conclusions

In this work we presented a system to detect and track vehicles using only 3D LiDAR data. The detection is performed separately in two CNN per-point classifiers that use a frontal and a bird's eye projection of the same input point cloud. These point-wise classifications are fused into a point cloud onto which a recursive clustering algorithm is applied. Bounding boxes are then grown using contextual information and other geometric features in addition to vehicle level information are then extracted to serve as observations to a multi-hypothesis extended Kalman filter tracking module.

The system is thoroughly evaluated separately in the challenging Kitti Detection and Tracking benchmarks, and is compared against previous approaches that only use front LiDAR information. The results show that the inclusion of our deep-classifiers drastically increases the system performance. On detection, we outperform or match other LiDAR-based approaches. Moreover, we get similar tracking scores than the ones obtained when using an ideal detector based on the point-level ground truth used to train the networks. This results confirm the hypothesis that the point-level false positives are –to some extent– uncorrelated between networks, and that the coincidence of different networks pointing the same cluster as vehicle, constitutes a strong evidence of vehicle existence. In addition, the presented approach shows an outstanding performance in the near field ($\approx 32\ m$) on pair to the human based annotation, where tracking results of the fusion detector are comparable to the ones obtained with the ideal predictor.

We leave for future research to explore the multi-class problem, detecting also pedestrians and cyclists, as well as to tackle the use of deep learning techniques for the bounding box extraction attending explicitly to context information.

References

[1] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *J3016 Standards*, pp. 2–35, 2018.

[2] X. Ji, G. Zhang, X. Chen, and Q. Guo, "Multi-perspective tracking for intelligent vehicle," *IEEE Trans. Intell. Transport. Syst.*, vol. 19, no. 2, pp. 518–529, 2018.

[3] M. Y. Abualhoul, P. Merdrignac, O. Shagdar, and F. Nashashibi, "Study and evaluation of laser-based perception and light communication for a platoon of autonomous vehicles," in *IEEE Intell. Transport. Syst. Conf.*, 2016, pp. 1798–1804.

[4] J. Larson, K. Y. Liang, and K. H. Johansson, "A distributed framework for coordinated heavy-duty vehicle platooning," *IEEE Trans. Intell. Transport. Syst.*, vol. 16, no. 1, pp. 419–429, 2015.

[5] R. Kohlhaas, T. Schamm, D. Lenk, and J. M. Zollner, "Towards driving autonomously: Autonomous cruise control in urban environments," in *IEEE Intell. Vehicles Sym.*, 2013, pp. 109–114.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Neural Inf. Process. Syst. Conf.*, vol. 1, 2012, pp. 1097–1105.

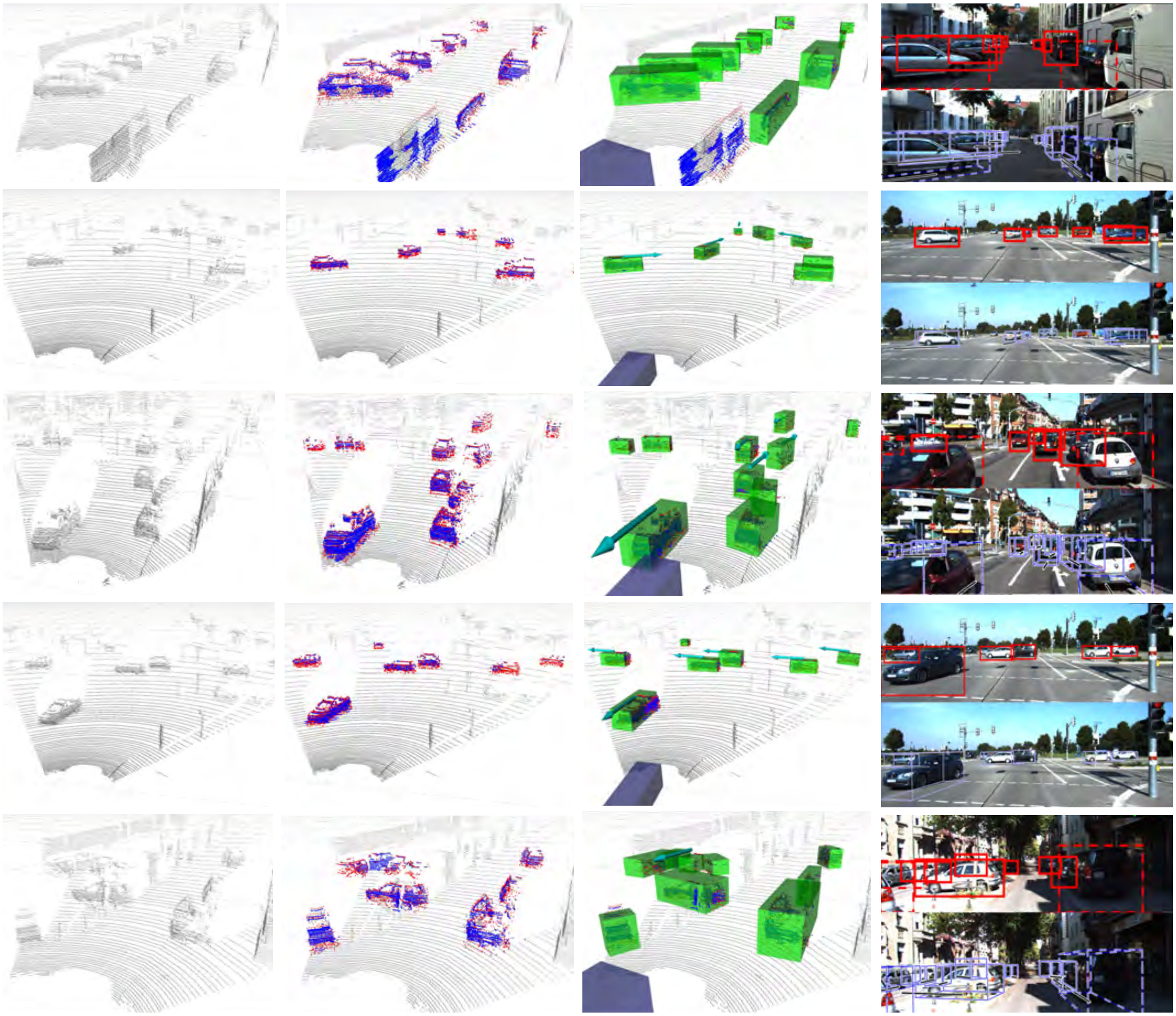


Fig. 9: Qualitative results of our system. All images are taken from the testing set of the Kitti Tracking benchmark, and therefore not previously seen during the training step. In columns, images show the raw input point cloud, the fused output of both deep-lidar classifiers, the final tracked vehicles and the RGB projected bounding boxes submitted for evaluation on Kitti (here just for visualization purposes). Notice that, despite the scarce information provided by the LiDAR sensor, our system is able to detect (red and blue colored points) and track (green boxes) vehicles in complex urban environments, even when they are very close to each other (first row), or even partially occluded (third and bottom row).

[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[8] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015, pp. 3431–3440.

[9] B. Li, T. Zhang, and T. Xia, “Vehicle detection from 3D lidar using fully convolutional network,” in *Robotics Sci. Syst. Conf.*, 2016.

[10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3D object detection network for autonomous driving,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.

[11] V. Vaquero, I. Del Pino, F. Moreno-Noguer, J. Sola, A. Sanfeliu, and J. Andrade-Cetto, “Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios,” in *Eur. Conf. Mobile Robots*, 2017.

[12] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum PointNets for 3D object detection from RGB-D data,” in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 918–927.

[13] Y. Zhou and O. Tuzel, “VoxelNet: End-to-end learning for point cloud based 3d object detection,” *Mem. & Cogn.*, vol. 5, no. 1, pp. 151–4, 2017.

[14] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, “Hallucinating dense optical flow from sparse lidar for autonomous vehicles,” in *IEEE Int. Conf. Pattern Recognit.*, 2018.

[15] C. You, C. Wen, C. Wang, J. Li, and A. Habib, “Joint 2-D-3-D traffic sign landmark data set for geo-localization using mobile laser scanning data,” *IEEE Trans. Intell. Transport. Syst.*, 2019.

[16] Y. Yu, H. Guan, and Z. Ji, “Automated detection of urban road manhole covers using mobile laser scanning data,” *IEEE Trans. Intell. Transport. Syst.*, vol. 16, no. 6, 2015.

[17] V. Vaquero, K. Fischer, F. Moreno-Noguer, A. Sanfeliu, and S. Milz, “Improving map re-localization with deep movable objects segmentation on 3d lidar point clouds,” in *IEEE Intell. Transport. Syst. Conf.*, 2019.

[18] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous

driving? the KITTI vision benchmark suite," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 770–778.

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6517–6525.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, 2020.

[22] D. Zeng Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Robotics Sci. Syst. Conf.*, 2015.

[23] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1513–1518.

[24] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE Int. Conf. Robotics Autom.*, 2017, pp. 1355–1361.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Neural Inf. Process. Syst. Conf.*, 2017, pp. 5099–5108.

[27] I. del Pino, V. Vaquero, B. Masini, J. Sol, F. Moreno-Noguer, A. Sanfeliu, and J. Andrade-Cetto, "Low resolution lidar-based multi-object tracking for driving applications," in *3rd Iberian Robotics Conf.* Springer, 2017, pp. 287–298.

[28] J.-S. Lee, J.-H. Jo, and T.-H. Park, "Segmentation of vehicles and roads by a low-channel lidar," *IEEE Trans. Intell. Transport. Syst.*, vol. 20, no. 11, pp. 4251–4256, 2019.

[29] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: A 3D object detection framework from LiDAR information," in *IEEE Intell. Transport. Syst. Conf.*, 2018, pp. 3517–3523.

[30] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias, "Object detection and classification in occupancy grid maps using deep convolutional networks," in *IEEE Intell. Transport. Syst. Conf.*, 2018, pp. 3530–3535.

[31] N. Sun, Y. Zeng, Y. Hu, Y. Han, J. Ye, X. Li, and S. Liu, "RT3D: Real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving," *IEEE Robotics Autom. Lett.*, vol. 3, no. 4, pp. 3434–3440, 2018.

[32] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe, "Training constrained deconvolutional networks for road scene semantic segmentation," *arXiv:1604.01545*, 2016.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[34] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud," in *IEEE Int. Conf. Robotics Autom.*, 2018, pp. 1887–1893.

[35] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *IEEE Int. Conf. Robotics Autom.*, 2019.

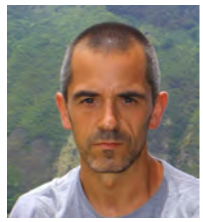
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE Int. Conf. Comput. Vis.*, 2015.

[37] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *IEEE Int. Conf. Robotics Autom.*, Shanghai, May 2011, pp. 1–4.

[38] B. Keni and S. Rainer, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *EURASIP J. Image Video Process.*, vol. 2008, p. 246309, 2008.

Víctor Vaquero obtained a B.Sc in Telecomm. Eng. from Universidad Autónoma de Madrid in 2010; an M.Sc in Robotics and Vision from Universitat Politècnica de Catalunya in 2012 as a *La Caixa* scholar; and a Ph.D in Robotics and Vision also from UPC in 2020. His PhD thesis focused on improving scene understanding for autonomous driving using deep learning, and has published various conference articles in this topic, as well as co-organized 4 workshops on Deep Learning for Automated Driving.

Ivan del Pino is an FPU Ph.D. student at the University of Alicante. He received an B.Sc in Telecomm. Eng. in 2014 and an M.Sc in Automation and Robotics in 2015 from the same university. His current interests are in outdoor autonomous navigation, including GNSS positioning systems and SLAM.



Francesc Moreno-Noguer is a Research Scientist at the Institut de Robòtica i Informàtica Industrial. His research interests are in Computer Vision and Machine Learning, with topics including human shape and motion estimation, and 3D reconstruction of rigid and nonrigid shape and camera pose. He received UPC's Doctoral Dissertation Extraordinary Award, several best paper awards (e.g. ECCV'18 Honorable Mention, ICCV'17 Workshop in Fashion, ICMVA'16), outstanding reviewer awards at ECCV'12 and CVPR'14, and Google and Amazon Faculty Research Awards in 2017 and 2019, respectively. Francesc's publication track includes 1 Springer Book Chapter, 31 "high-impact" journals (with 9 IEEE Trans. PAMI and 5 IJCV), 35 conferences in highly competitive venues with acceptance rates below 30% (21 CVPR, 7 ECCV, 5 ICCV) and 48 publications in other international conferences.

Joan Solà is currently a Ramón y Cajal researcher at the Institut de Robòtica i Informàtica Industrial in Barcelona. He obtained his MSc in Telecomm. and Electronics from the Universitat Politècnica de Catalunya, and a PhD in Robotics from the University of Toulouse in 2007. He has also worked in the industry in the renewable energies sector, and was involved in the construction of a manned submarine for depths up to 1200m. Joan has contributed to monocular SLAM, especially in the undelayed initialization of landmarks, and is interested in state estimation for robots with particularly large dynamics and degrees of freedom, such as humanoids and aerial manipulators. His current projects turn around whole-body estimation and control, including multi-sensor fusion, localization and mapping, machine learning, and model predictive control.

Alberto Sanfeliu BSEE (1978) and PhD (1982) both from Univ. Politècnica de Catalunya. Joined UPC's faculty in 1981 and is now full professor of Computational Sciences and Artificial Intelligence. He leads UPC's Artificial Vision and Intelligent Systems Research Group and is head of the Mobile Robotics Research Line at the Institut de Robòtica i Informàtica Industrial. Former director of IRI, of UPCs Automatic Control Dept., and past president of the Spanish Assoc. for Pattern Recognition. He works on theoretical aspects in pattern recognition, computer vision and robotics, and on applications for vision defect detection, tracking, object recognition, robot vision and SLAM. He has several patents on CV for quality control, has authored books in pattern recognition and SLAM, published more than 260 papers in international journals and conferences, and led/participated in more than 40 R&D projects, 15 of them funded by the European Commission. IAPR Fellow and IEEE Member.

Juan Andrade-Cetto is Research Scientist and Director of the Institut de Robòtica i Informàtica Industrial, CSIC-UPC. Juan holds a BSEE from CETYS Univ. (1993), an MSEE from Purdue Univ. (1995, Fulbright Scholar), and a PhD from UPC (2003, EURON Georges Giralt Best PhD Award 2005). He was a Juan de la Cierva fellow at CVC and a Ramón y Cajal fellow at IRI. His reserch work is in the areas of perception and state estimation for robotics. He has authored three monographs in the Springer STAR series, and over 100 journal and conference papers, has been PI of several basic and applied research projects for national and regional Spanish research agencies, as well as in the FP7 and H2020 research programs of the EU. Notwithstanding are his contributions in the theoretical aspects of the SLAM problem, and more recently, to the problem of unmanned aerial manipulation.