IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 15, 2021, accepted November 24, 2021, date of publication November 30, 2021,
date of current version December 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3131389

BirdNet+: Two-Stage 3D Object Detection in LiDAR Through a Sparsity-Invariant Bird's Eye View

ALEJANDRO BARRERA[1], JORGE BELTRÁN[1], CARLOS GUINDEL[1], JOSÉ ANTONIO IGLESIAS[2], (Member, IEEE), AND FERNANDO GARCÍA[1], (Member, IEEE)

[1]Department of System Engineering and Automation, Universidad Carlos III de Madrid (UC3M), Leganés, 29811 Madrid, Spain
[2]Department of Computer Science and Engineering, Universidad Carlos III de Madrid (UC3M), Leganés, 29811 Madrid, Spain

Corresponding author: Alejandro Barrera (alebarre@pa.uc3m.es)

This work was supported in part by the Government of Madrid (Comunidad de Madrid) under the Multiannual Agreement with the University Carlos III of Madrid (UC3M) in the line of "Fostering Young Doctors Research" (PEAVAUTO-CM-UC3M), and in part by the Context of the V Regional Programme of Research and Technological Innovation (PRICIT).

ABSTRACT Autonomous navigation relies upon an accurate understanding of the elements in the surroundings. Among the different on-board perception tasks, 3D object detection allows the identification of dynamic objects that cannot be registered by maps, being key for safe navigation. Thus, it often requires the use of LiDAR data, which is able to faithfully represent the scene geometry. However, although raw laser point clouds contain rich features to perform object detection, more compact representations such as the bird's eye view (BEV) projection are usually preferred in order to meet the time requirements of the control loop. This paper presents an end-to-end object detection network based on the well-known Faster R-CNN architecture that uses BEV images as input to produce the final 3D boxes. Our regression branches can infer not only the axis-aligned bounding boxes but also the rotation angle, height, and elevation of the objects in the scene. The proposed network provides state-of-the-art results for car, pedestrian, and cyclist detection with a single forward pass when evaluated on the KITTI 3D Object Detection Benchmark, with an accuracy that exceeds 64% mAP 3D for the Moderate difficulty. Further experiments on the challenging nuScenes dataset show the generalizability of both the method and the proposed BEV representation against different LiDAR devices and across a wider set of object categories by being able to reach more than 30% mAP with a single LiDAR sweep and almost 40% mAP with the usual 10-sweep accumulation.

INDEX TERMS Bird's eye view (BEV), LiDAR, object detection, autonomous driving.

I. INTRODUCTION

3D object detection is broadly identified as one of the most critical tasks performed by the perception stack of an autonomous vehicle. Unlike 2D approaches, whose results are expressed in image coordinates, 3D detection methods aim to describe the traffic scene geometry by localizing relevant elements in the vehicle's environment. This information is fundamental to building a reliable environment model that allows the planning and control modules to make safe and efficient navigation decisions well in advance.

Nevertheless, accurately grasping the geometric characteristics of objects from on-board sensor data is a challenging task. Cameras, often the preferred source of exteroceptive information for automated driving systems, are not well suited to this task due to the loss of depth information involved in the capture process. Although feasible [1], monocular 3D detection approaches usually rely on approximations that negatively affect the results. On the contrary, active modalities are naturally fitted to the task since they provide mainly spatial data. In particular, LiDAR sensors provide very accurate range information while being reasonably robust to lighting and weather conditions, making them an ideal choice for vehicle sensor setups [2]. However, the high sparsity of LiDAR data, in contrast to the compact

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang.

representation provided by images, makes them non-trivial to handle and process.

Recent advances in deep learning have meant a dramatic change in the way sensor information is processed. Pushed by the concept of convolutional neural network (CNN), 2D detection in images has reached an unprecedented degree of maturity with well-established architectures such as Faster R-CNN [3] and YOLO [4]. On the other hand, the processing of information structured as a 3D point cloud, as is the case with LiDAR data, has also been a matter of great interest, resulting in ideas such as PointNet [5]. However, unlike what happens with images, LiDAR data representing a significantly large region of interest, such as a vehicle's environment, becomes too bulky to be straightforwardly fed to a neural network.

This paper proposes a 3D object detection framework that combines the best of both worlds: it works with LiDAR data as input but performs inference through a CNN architecture based on Faster R-CNN. The gap is closed through a convenient bird's eye view (BEV) representation of LiDAR data that mitigates the differences in density due to distance and is agnostic to the LiDAR scanner's resolution. Fig. 1 shows a general overview of the approach. We will prove that LiDAR data can be effectively processed through structures initially intended for images, achieving state-of-the-art 3D object detection performance on two major autonomous driving benchmarks: KITTI [6] and nuScenes [7].

Our overall contribution is therefore twofold. First, we introduce a novel bird's eye view representation of LiDAR which:

- represents the key features of the original point cloud despite its voxelization, enabling a precise object characterization.
- is robust against differences among unlike scanner specifications thanks to its sparsity-invariant encoding.

Second, we present an object perception framework that:

- proves the adequacy of bidimensional CNN networks, originally tailored to image processing, to exploit LiDAR features to perform end-to-end 3D detection.
- is able to detect and classify different types of objects in a traffic scene with a single forward pass.

Both components together constitute an approach with a differentiating advantage: it only needs LiDAR data as input to provide identification and proper 3D representation of different obstacles in a 360° range. This feature makes it an ideal component of an automated vehicle's perception stack, either working on its own (when a limited sensor suite is available) or as a complement and fallback of a more sophisticated multi-modal algorithm.

The BirdNet framework was preliminarily introduced by two previous conference publications [8], [9]. This manuscript offers a comprehensive description of the method as a whole and provides deep insight into it through a broad set of experimental results, allowing valuable conclusions to be drawn. Source code for training and validation is publicly available at https://github.com/AlejandroBarrera/BirdNet2.

The remainder of the text is structured as follows. Section II provides a summary of recent works aimed at 3D detection from on-board LiDAR scanners. Section III describes the details of the proposed BEV representation, whereas the 3D object detection framework itself is presented in Section IV. Experimental results are included in Section V. Finally, Section VI highlights the main conclusions and discusses future work lines.

II. RELATED WORK

The task of object detection using LiDAR sensors benefits from the spatial representation provided by modern laser scanners [10]. These devices are able to faithfully capture both the 3D shape and reflectiveness of the objects in the environment, usually spanning the whole horizontal field of view. Although this information is sufficient to identify and classify the different road participants, the characteristics of the collected data pose a challenge when deploying such algorithms for real-time applications. Thus, its lack of structure and sparsity has led to the creation of different lines of research based on distinct representations of LiDAR information.

A. RAW POINT CLOUD

The first group of works uses the LiDAR input as-is in order to obtain better features based on the fine-grained geometry and intensity values provided by the captured point clouds. Although some approaches utilize the information from the raw cloud through all the layers to produce the final 3D detection boxes [11], [12], most approaches rely on a previous downsizing of the cloud to reduce the computational load. Among these methods, PointNet-based architectures are commonly found, as they are able to build a robust representation of objects regardless of the order and invariant to motion transformations. In [13], [14], the output of a 2D object detector is used to create frustums, which are then used as inputs. Likewise, semantic segmentation in the camera space is used in [15]. Alternatively, Point-RCNN [16] reduces the size of the LiDAR cloud by performing a background-foreground classification of the points before feeding the positive set into the final 3D box estimation networks. Other proposals, such as STD [17], lie between raw and voxel-based pipelines using a point-wise feature extraction over the whole LiDAR set followed by a discretization step that significantly reduces the inference time.

B. VOXELS

The second subset of networks takes a voxelized version of the LiDAR data as input. This kind of representation reduces the information volume and guarantees a regular structure that enables its processing using 3D convolutions. Space is divided into a volumetric grid where each 3D cell, also known as voxel, stores features computed from the points lying inside. Many approaches combine voxel feature encoders with region proposal networks, either using a single [18], [19] or multiple voxel scales [20] as inputs. Similarly, Part-A2 [11] follows a two-stage approach, where,

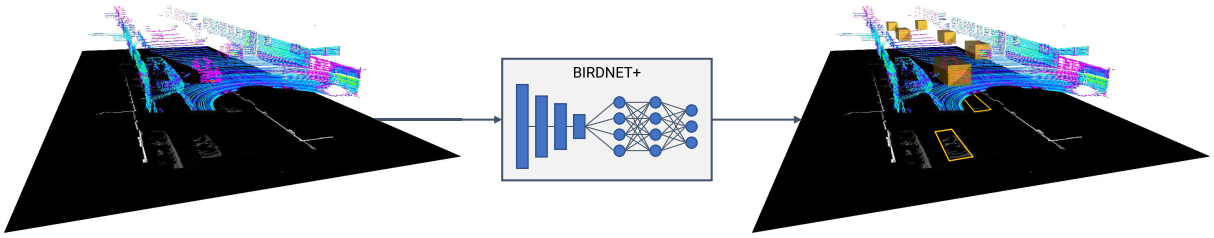


FIGURE 1. Input/output data involved in the BirdNet+ detection framework. The raw LiDAR point cloud is projected onto a BEV image where each pixel contains information of a 3D pillar. This BEV is fed to our CNN-based model that generates a set of fully parameterized 3D bounding boxes.

firstly, both intra-object parts and point-wise semantic segmentation are estimated and fed to a part-aggregation step that produces the final detections. Other works predict the final 3D boxes following single-stage [21] or anchor-free [22] schemes. More recently, self-attention modules [23] have been applied to voxel-based networks obtaining state-of-the-art results.

C. 2D PROJECTIONS

Finally, a third approach aims to further compact the input by making use of bidimensional representations of the LiDAR cloud. To this aim, the laser data is projected into pseudo-images, decreasing the processing time and enabling the use of 2D object detection networks.

Among this group of frameworks, those taking the bird's eye view (BEV) projection as input are the most popular. Some works use hand-crafted BEV images to feed single-stage [24], [25] or two-stage object detectors [9], [26]. Each cell in the 2D input typically includes information on the intensity, height, and distribution of the points lying inside. In MODet [27], the representation of the BEV is simplified to a binary occupancy grid. Alternatively, PointPillars [28] introduced a learned BEV encoding produced by a PointNet network able to compute features directly from the original 3D points contained in the cell.

Although less frequently, other papers make use of the Range View (RV) projection of the LiDAR to perform end-to-end object detection. LaserNet [29] is able to predict per-point class and bounding box distributions that are then clustered to obtain the final BEV detections. Recently, RangeRCNN [30] presented a novel approach able to learn cues in the range projection of the LiDAR cloud and transfer them to the BEV representation to produce the final 3D box estimation.

Our work falls into the group of approaches making use of a BEV projection but pushes the concept to its limit by aiming at the end-to-end detection of all the parameters of the 3D box simultaneously. This task entails the design of a BEV encoding able to meet the inherent demands of homogeneity and compactness and an object detector able to infer the complete set of parameters involved in 3D detection.

III. INPUT DATA REPRESENTATION

The first stage in the BirdNet pipeline involves the conversion of raw LiDAR data into a 2D image-like structure that can be accepted by the CNN detector. As stated before, we use the representation commonly referred to as bird's eye view (BEV), which is a grid map where each cell encodes information about the LiDAR points contained in a tall, narrow voxel lying on the ground plane. The BEV representation has many advantages over other alternatives; for instance:

- The dimensionality reduction is almost harmless in the context of vehicle perception, as it is usually safe to assume that all traffic participants will move on the same plane.
- Real scales are preserved so that assumptions based on the objects' physical dimensions can be made.
- Objects do not overlap with each other, which makes them easier to detect.
- Convolutions in the detector will conserve the local range information, as desired.

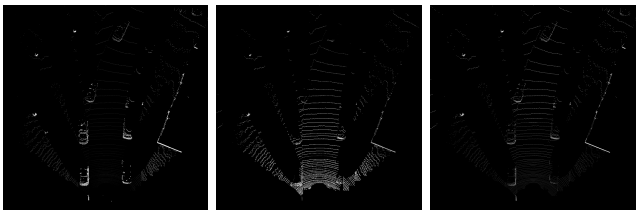
A. BIRD'S EYE VIEW IMAGE

Formally, we define the BEV as an array made of $M \times N$ elements, each representing a square cell of side s. Therefore, the total area covered by the BEV is $sM \times sN$. Each cell can be described by a variable number of features, C, so the BEV becomes a 3D tensor with dimensions $M \times N \times C$.

As a baseline, we adopt $s = 5\,\text{cm}$ as the cell size since it offers a suitable resolution for the detection of small objects, such as pedestrians and cyclists. However, it will be later shown that $s = 10\,\text{cm}$ can also be a good choice for certain use cases. On the other hand, the BEV size ($M \times N$) determines the region where obstacles can be detected. It should be noted that, in any case, detection performance degrades with distance from the sensor due to the reduction of LiDAR point density, so it is reasonable to find an optimal trade-off taking into account the computation time.

The LiDAR scanner's characteristics, mainly the number and distribution of scan layers, determine the optimal choice for each application. For instance, our previous works [8], [9] used a 1400×700 (70×35m) BEV image for the area in front of the vehicle in the KITTI dataset; however, as will be discussed in Section V, we experimentally found that a 900×1000 (45×50m) BEV image is better suited to the particularities of the application, containing more objects of interest in fewer data.

As for the feature encoding, different alternatives will be studied in Section V, although our baseline, which will be proven effective, uses $C = 3$ channels, mimicking an RGB image. They express the maximum height ($H_{\max}$), mean intensity (I), and density (D) of the points in a $s \times s \times H_{\text{top}}$ pillar lying on the theoretical ground plane at the cell location. Points above H_{top}, set to 3 m and 4 m in our experiments in KITTI and nuScenes, respectively, are not considered since they are unlikely to belong to relevant objects. All magnitudes are normalized from their input domain to the 0–255 range. Fig. 2 provides an example of each channel's content in a particular scene.



(a) (b) (c)

FIGURE 2. Baseline BEV feature encoding: (a) Maximum height. (b) Mean intensity. (c) Normalized density. Contrast corrected for visualization.

It is noteworthy that this encoding does not provide information about the lowest point in each cell, i.e., the ground's height; experimental results will support this decision. Furthermore, whereas the maximum height and intensity are roughly homogeneous over the entire LiDAR measurement range, the number of points in each cell will depend on the distance and the resolution of the LiDAR scanner. For that reason, we use a normalized density representing the number of points in each cell divided by the maximum possible, which is computed considering the device's characteristics, as described in the following section.

B. DENSITY NORMALIZATION

Mechanically rotating multi-layer LiDAR devices differ from each other, essentially, by the number and vertical distribution (elevation angles) of the set of scan layers, L, and their horizontal angular (azimuth) resolution, $\Delta\theta$. The maximum number of LiDAR points that can fall in each BEV cell depends not only on the distance but also on these properties.

As BEV features should desirably be invariant to both kinds of factors, we propose a novel density encoding that is normalized by the maximum number of possible points in a cell according to the physical constraints:

$$D(x, y) = \frac{N_{\text{points}}(x, y)}{N_{\max}(x, y, L, \Delta\theta)}, \tag{1}$$

where $D(x, y)$ is the value of the density channel in cell (x, y), $N_{\text{points}}(x, y)$ the number of points measured in that cell, and $N_{\max}(x, y, L, \Delta\theta)$, the maximum number of points.

The function $N_{\max}(x, y, L, \Delta\theta)$ can be obtained by geometric derivation. Let us limit the analysis to the 0–H_{top} range with respect to the theoretical ground plane considered for the BEV image's computation. The maximum number of points from a particular LiDAR layer that can be contained in a BEV $s \times s$ cell is given by the case in which a solid cuboid is placed at that location, spanning the whole volume represented by the cell ($s \times s \times H_{\text{top}}$). The problem then boils down to finding the number of points that would fall in that hypothetical cuboid.

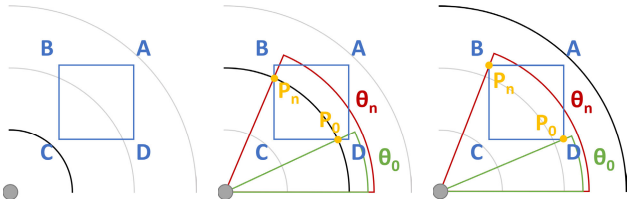
The analysis will be performed on a per-layer basis so that the final value will be obtained as the sum of the contributions of each layer, $N_{\max,l}$:

$$N_{\max}(x, y, L, \Delta\theta) = \sum_{l \in L} N_{\max,l}(x, y, l, \Delta\theta). \tag{2}$$

When taking into account the paths of the laser beams, each layer can be seen as a cone (without the base) whose vertex is in the LiDAR rotation axis. By definition, the planes limiting the represented volume (i.e., planes at $z = 0$ and $z = H_{\text{top}}$) are aligned with the LiDAR scanner. Therefore, the intersection of each of the cones with these planes is a circle.

A top view of the situation is shown in Fig. 3, where the intersection of the LiDAR layers with the upper and lower limits are depicted in grey/black, and the cell for which $N_{\max}$ is being computed is outlined in blue as the $ABCD$ square. As shown, three different cases can be distinguished, depending on the relative position of the square representing the cell with respect to the circle generated by the intersection of the LiDAR layer with the vertical limits:

a) The square is completely outside the circle.
b) The circle cuts the square at two points, P_0 and P_n, depicted in yellow in Fig. 3b.
c) The square is completely inside the circle.



(a) (b) (c)

FIGURE 3. Horizontal cross-section of possible LiDAR-cell intersection scenarios: (a) No intersection. (b) LiDAR ring intersects with the upper/lower plane of the 3D pillar. (c) The cell is fully intersected by the laser plane.

In case a), no point from that layer can fall in the cell under consideration, so $N_{\max,l} = 0$. In the other two cases, LiDAR beams from the analyzed layer would indeed collide with the $s \times s \times H_{top}$ cuboid. Points in the cell would be those in the interval between θ_0 and θ_n, each representing the azimuth angle of one of the border points (P_0 and P_n). Then, $N_{\max,l}$ can be straightforwardly computed given the horizontal resolution of the LiDAR scanner, $\Delta\theta$:

$$N_{\max,l} = \left\lceil \frac{|\theta_n - \theta_0|}{\Delta\theta} \right\rceil. \tag{3}$$

The calculation of θ_0 and θ_n from P_0 and P_n is trivial. Let $P_{i,x}$ and $P_{i,y}$ denote the x and y coordinates of point P_i, for $i \in \{0, n\}$; then

$$\theta_i = \arctan\left(\frac{P_{i,x}}{P_{i,y}}\right). \tag{4}$$

Therefore, the focus is on the localization of points P_0 and P_n. In case c), P_0 and P_n coincide with the B and D vertices of the square representing the cell, as shown in Fig. 3c. As the grid is arbitrarily created, these positions are known. However, in case b), P_0 and P_n are in intermediate positions in the BC and CD sides of the square. The computation of their positions involves solving the system of equations posed by the equations of the circle and the lines where the BC and CD segments lie. Let us denote V_y and V_y the x and y coordinates of one of the vertices of the square, V. Thus, point P_i (either $i = 0$ or $i = n$) must satisfy

$$\begin{cases} P_{i,x}^2 + P_{i,y}^2 = d^2 \\ P_{i,y} - C_y = \dfrac{V_y - C_y}{V_x - C_x}(P_{i,x} - C_x) \end{cases} \tag{5}$$

Here, d denotes the radius of the circle, and V is one of the vertices connected with C; that is, either $V = D$ (if $i = 0$) or $V = B$ (if $i = n$). Among the two solutions obtained, only one would be feasible at the current location.

This is also the way to determine the case corresponding to each cell (x, y) among the three discussed above: in a), the points of intersection will be closer to the origin than the square, and in c), they will be further away.

This approach deals with the differences in data density across the measurement range inherent to the LiDAR modality. Hence, the resulting BEV image fits better with the parameter sharing paradigm of convolutional layers, which assumes that features are invariant to location. Besides, it enables seamless domain adaptation whenever a trained model needs to be deployed in a different sensor setup.

IV. OBJECT DETECTION

The central component of BirdNet is a two-stage object detector based on the Faster R-CNN meta-architecture [3]. Similarly, our architecture accepts a 2D image-like structure (the BEV representation) as input and processes it through a CNN responsible for extracting meaningful features. Later, these features are used for proposal generation and ROI classification and regression, as usual. However, our BirdNet+ variant extends the set of detection heads with custom branches that take advantage of the shared features to estimate all the parameters involved in 3D detection.

Consequently, the BirdNet+ detector, sketched in Fig. 4, is an end-to-end trainable model from the BEV image to the set of 3D detections. Training is carried out through a multi-task loss that optimizes all the branches and the feature extractor jointly.

The choice of Faster R-CNN as detection core is based on the fact that two-stage detectors generally offer better detection performance than their one-stage counterparts [31]. This advantage is inherent to their design: the first stage is a region proposal network able to filter out most negative regions and provide an informed guess about the remaining candidates; thus, the second stage, heavier than the first, can focus on these regions, sampling high-quality features through ROIAlign and providing accurate class and bounding box estimates on top of the initial proposals.

The following sections are devoted to describing the details of the different components of the BirdNet+ detector, from the feature extractor to the different classification and regression branches used for 3D detection. The multi-task loss that ties everything together is also discussed.

A. FEATURE EXTRACTION

Various alternatives have been tested as the convolutional backbone of the detection framework, which is in charge of encoding the input BEV image into a set of feature maps valid for the diverse inference tasks involved.

Generally speaking, we have observed that most of the models typically used in image detection, such as VGG-16 [32] or ResNet-50 [33], are suitable for the task, despite the peculiarities of the BEV representation. However, one caveat applies: as objects span very few pixels in BEV input data, the downsampling factor of the resulting feature maps (usually, 16×) proves excessive for some object categories. Resolution can be enhanced by removing an intermediate pooling layer [8] or extracting features from an early stage [9] (resulting in an 8× downsampling).

As an optimal alternative, we propose the use of a ResNet-50 backbone together with a feature pyramid network (FPN) [34]. On the one hand, ResNet-50 offers an excellent trade-off between accuracy and computation speed, being less prone to overfitting than other alternatives. On the other hand, the FPN efficiently extracts feature maps at different resolutions, allowing accurate detection of objects of very different sizes, as is the case with the diverse types of traffic participants. Our framework employs those feature maps with strides 4, 8, and 16 with respect to the input image.

B. REGION PROPOSAL

The lightweight region proposal network (RPN) acts as a first detection stage generating candidate proposals from a set of predefined anchors. That is, proposals' bounding boxes are obtained by regression from the base anchors.

Unlike in generic object detection in images, the sizes of objects in the BEV representation are constrained to a very narrow range. Because of this, we employ an ad-hoc set of nine anchors obtained by statistical analysis, with scales (area, in pixels) 16^2, 48^2, and 80^2, and aspect ratios 1:1, 1:2, and 2:1. These anchors fit better the representation of typical traffic participants in BEV.

Contrary to the usual practice where each stride is assigned a single anchor scale, we apply the whole set of anchors over each feature map from the FPN. This setup offers increased flexibility while keeping the total number of anchors handled by the RPN manageable.

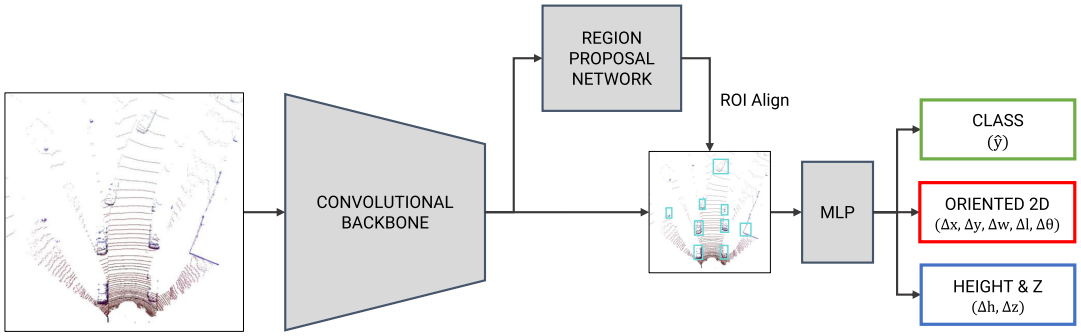


FIGURE 4. Overview of the proposed 3D object detector.

The RPN is trained to generate axis-aligned proposals representing the minimum enclosing 2D boxes of the objects in BEV data. In practice, labels for training are generated from the extreme coordinates (minimum/maximum) of the rotated annotations in the dataset, and therefore the proposals actually represent the minimum enclosing boxes of the rotated boxes representing the final detections. These will be computed in the second stage by regression from these proposals; this approach, in contrast to the rotated RPN (RRPN) introduced in [35], is highly efficient and will be proven effective.

In the second detection stage, features are sampled at these candidate regions through a 7×7 ROIAlignV2 pooling and fed to a common trunk composed of two 1024-dimensional fully connected (FC) layers. The pipeline ends in several sibling branches responsible for the different classification and regression tasks.

C. DETECTION HEADS

Our framework aims to provide 3D detections encoded by nine parameters: one for the category, three for the 3D location of the object center (x, y, and z), another three for the dimensions of the enclosing cuboid (length l, width w, and height h), and two additional ones for its orientation on the road plane, or yaw angle θ (θ_{bin} and $\Delta\theta$). This is a compact encoding that is unambiguous but optimizes the number of parameters by avoiding redundancies (e.g., separate regression for each vertex, as in [36], [37]).

Detections are given in a coordinate system fixed in the LiDAR scanner where x points forward, y towards the left, and z upwards. Within the inference framework, lengths are expressed in pixels in the BEV image; conversions in both directions are straightforward using the grid cell resolution s.

The branches proposed for estimating these parameters can be divided into three groups: category classification, 2D BEV rotated box regression, and 3D box regression. The first two are needed to fulfill the BEV detection task in an end-to-end fashion, whereas the third provides a proper 3D estimation.

1) CLASSIFICATION

The classification branch replicates the original Faster R-CNN configuration, with an FC layer made of as many units as possible classes, N_{cls}, plus one extra to allow room for background instances. As usual, a softmax operation is applied to the output to normalize the scores. Unlike other approaches in the literature, we intend to perform simultaneous detection and classification of all the relevant classes using a single model. We have experimented with $N_{\text{cls}} = 3$ and $N_{\text{cls}} = 10$, as will be shown in Section V.

2) ROTATED 2D BOX REGRESSION

The next group of tasks comprises the estimation of the rotated 2D boxes representing the detections in BEV, which involves the inference of the parameters x, y, l, w, and θ.

The computation of the location and dimension parameters follows the design of the bounding box refinement task in Faster R-CNN, where final bounding boxes are regressed from the RPN proposals by estimating an offset per each encoding parameter. Hence, we define the targets to be estimated by the bounding box regression branch as follows:

$$\begin{aligned} \Delta x &= \lambda_x \cdot \frac{x - x_p}{w_p} \qquad \Delta w = \lambda_w \cdot \frac{\ln(w)}{w_p} \\ \Delta y &= \lambda_y \cdot \frac{y - y_p}{l_p} \qquad \Delta l = \lambda_l \cdot \frac{\ln(l)}{l_p} \end{aligned} \tag{6}$$

In these equations, x, y, l, and w represent the actual values of the parameters, whereas their equivalents with subscript p refer to the magnitudes of the RPN proposal from which the bounding box is regressed. Weights λ_i; $i \in \{x, y, l, w\}$ are empirically chosen so that the regression targets, which are already normalized over the proposal dimensions, have approximately unit variance. Note that the source proposals are axis-aligned rectangles, whereas the target boxes are rotated rectangles that should be ideally inscribed in the proposals. Fig. 5 illustrates this concept with an example.

The offsets are regressed through an FC layer with $N_{\text{cls}} \times 4$ outputs, each representing the value of one of the parameters for a particular object category.

However, the 2D BEV boxes representing the detections are not fully defined until an estimate of their orientation, θ, is computed. As RPN proposals do not provide orientation cues, a different strategy is adopted here: as in [13], the yaw angle is computed as the combination of a coarse estimation,

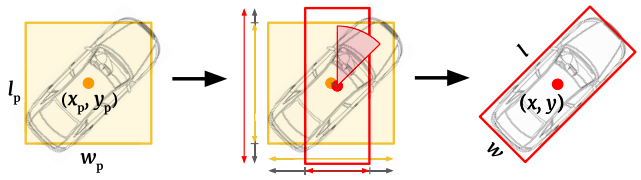


FIGURE 5. Proposed regression method. From left to right: an (imperfect) RPN proposal; required regression to obtain the target proposal (red) from the RPN's axis-aligned box (yellow); and the final object proposal.

obtained through classification into discrete angle bins, and a refinement step based on the regression of the residual angle.

For the former, the circle is discretized into N_{bin} slices, each spanning an interval of $360\,^\circ/N_{\text{bin}}$. A specific branch, made of an FC layer with $N_{\text{cls}} \times N_{\text{bins}}$ elements, is responsible for performing category-aware classification to assign every object to one of these bins. In practice, a coarse orientation estimation can be obtained as the center of the selected bin, as detailed in [38].

This approach is more robust than direct regression and provides a reasonably accurate estimation in most situations despite being limited by the slice resolution, especially if bins are selected to be aligned with the most usual directions (forward/backward, left/right). Nevertheless, we employ an additional refinement task to complement this discrete estimation, expanding the resolution to the full $360\,^\circ$ range. The idea is to regress the residual value between the actual object's orientation (θ) and the center of the estimated angle bin (θ_{bin}). As with the bounding box regression, the regression target is normalized to the unit, becoming

$$\Delta\theta = \frac{\theta - \theta_{\text{bin}}}{\theta_{\text{res,max}}}, \tag{7}$$

where $\theta_{\text{res,max}}$ is the maximum value that the residual angle can adopt; that is, half of the bin size: $360\,^\circ/2N_{\text{bin}}$.

This task is carried out by an $(N_{\text{bin}} \times N_{\text{cls}})$-element FC layer implementing a category-aware, bin-aware regression; i.e., a residual is estimated for each discrete bin and each object category. In our experiments, we use $N_{\text{bin}} = 12$ (and, therefore, $\theta_{\text{res,max}} = 15\,^\circ$), which we found a satisfactory trade-off.

Remarkably, as a result of this orientation estimation strategy, our framework can not only correctly place the 3D cuboids representing the objects in the scene but also distinguish between forward and backward orientations.

3) HEIGHT AND VERTICAL COORDINATE REGRESSION

Although the previous tasks are sufficient to provide 2D detections in the BEV image, a final step is needed to estimate the only two parameters left to fully define each 3D detection: elevation (z) and height (h).

As the BEV representation involves a dimensionality reduction that affects the height information, existing approaches based on this data structure frequently use simple models making strong assumptions, such as constant objects' height or flat ground [36], [39], [40]. A previous iteration of our framework used a coarse ground estimation based on a grid map computed from the lowest LiDAR points in each cell [8]. Conversely, we will prove that it is possible to include the regression of these parameters into the inference framework.

The task is posed as a natural extension of the 2D bounding box regression introduced in the previous section. As depicted in Fig. 6, z and h are regressed for each proposal from a hypothetical 3D anchor resting on a flat ground plane (i.e., at the same vertical position as the ego-vehicle), with a fixed height chosen as the typical value among the instances of that category on the training set. Therefore, even though the height of the LiDAR scanner with respect to the ground is assumed known and constant, significant variations in the vertical location of the detected obstacles, e.g., due to terrain elevations or slopes, can be suitably represented.

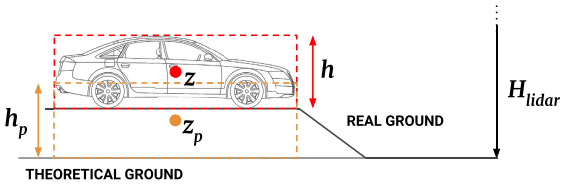


FIGURE 6. Parameters involved in the height regression task.

The regression targets are normalized following the same encoding used for the estimation of the rotated 2D boxes:

$$\Delta z = \lambda_z \cdot \frac{z - z_p}{h_p} \quad \Delta h = \lambda_h \cdot \frac{\ln(h)}{h_p}, \tag{8}$$

where z_p and h_p denote the vertical position and the height of the prototypical 3D anchor, and weights λ_z and λ_h are empirically adjusted. Values used for h_p in the KITTI dataset are 1.53 m, 1.76 m and 1.74 m for cars, pedestrians, and cyclists, respectively.

As with the rest of the inference tasks, the estimation of each of these parameters is carried out through N_{cls} FC units that perform category-aware regression of the above targets. Notably, input data lack information about the lowest points in a cell (which should represent the ground's position) when using the baseline BEV representation described in Section III (made of maximum height, density, and intensity). However, the features used in the detection branches, sampled from loose axis-aligned proposals, contain enough contextual information to describe the floor around the objects, thus enabling geometrical reasoning about the vertical parameters.

D. MULTI-TASK TRAINING

To train our end-to-end framework, we follow the approximate joint training strategy described in [3] so that the RPN and every detection head are optimized together. To that end,

we use a multi-task loss L that blends all the classification and regression tasks:

$$L = L_{p,\text{cls}} + L_{p,\text{bbox}} + L_{\text{cls}} + L_{xywl} + L_{\theta,\text{bin}} + L_{\theta,\text{res}} + L_{zh} \tag{9}$$

Here, $L_{p,\text{cls}}$ and $L_{p,\text{bbox}}$ account for the RPN contributions, addressing the classification and regression of the proposals. Regarding the second stage, L_{cls} represents the final category classification loss, whereas L_{xywl} and L_{zh} express the regression losses corresponding to the estimation of the rotated 2D bounding box and the height and vertical position, respectively. Finally, $L_{\theta,\text{bin}}$ corresponds to the orientation bin classification and $L_{\theta,\text{res}}$ to the residual angle regression.

All these loss elements follow the spirit of the vanilla Faster R-CNN [3]. Therefore, classification components, namely $L_{\text{p,cls}}$, L_{cls}, and $L_{\theta,\text{bin}}$, are included as cross-entropy losses, L_{cls}:

$$L_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i^* \cdot \log(\hat{\mathbf{y}}_i), \tag{10}$$

where $\mathbf{y}^*$ denotes the one-hot ground-truth vector for the classification and $\hat{\mathbf{y}}$, the network's prediction after the softmax operation. Among the classification loss components, $L_{\text{p,cls}}$ is a binary classification where proposals containing objects are separated from those representing the background, L_{cls} is used for the final classification into the N_{cls} possible categories, and $L_{\theta,\text{bin}}$ refers to the classification into N_{bin} non-overlapping angle slices. In each case, the loss is normalized over the number of training samples used in each iteration at the corresponding stage, N.

On the other hand, regression components make use of a pure ℓ_1-loss, L_{reg}:

$$L_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i^*|, \tag{11}$$

which, in practice, represents the mean absolute error between the predicted magnitude, $\hat{y}_i$, and the ground-truth value, y_i^*. The regressed magnitudes are different for each loss component; hence, they are the size and position offsets of the axis-aligned proposals in $L_{p,\text{bbox}}$, the offsets of the 2D rotated box dimensions in L_{xywl}, the residual angle in $L_{\theta,\text{res}}$, and, finally, the vertical parameters in L_{zh}. Again, the loss is always normalized by the number of samples in each training iteration.

As in [3], proposals are associated with ground-truth labels according to their Intersection over Union (IoU). The regression losses and the bin classification loss only consider foreground samples, for which a target 3D bounding box can be defined. However, it should be noted that the normalization term, i.e., N in (10) and (11), accounts for the whole set of samples used in each training iteration; in that way, the influence of a particular sample in the final loss does not depend on the ratio of foreground objects in the training set.

As suggested in [8], we fine-tune the models from an ImageNet [41] pre-trained model, which proved effective despite the domain gap. Nevertheless, we let all the convolutional layers be altered during training, unlike the standard practice in Faster R-CNN where the first two layers, aimed to extract general-purpose features, are frozen at fine-tuning. At the new detection branches, we use a Xavier initialization with normal distribution [42].

Pixels in the input BEV images are not mean-centered since the average values will be, anyway, close to zero due to the data sparsity. As usual, random horizontal flipping is applied as augmentation. At inference time, non-maximum suppression (NMS) is applied to the output of the detector to filter out redundant results. A modified version considering the rotated boxes is used, with an IoU threshold of 0.3.

V. EXPERIMENTS AND RESULTS

A comprehensive set of experiments has been carried out to validate our approach's performance and offer in-depth insight into the soundness of different solutions adopted throughout the framework. The bulk of the experimentation lies on the well-known KITTI object detection benchmark [6], although results in the more recent nuScenes dataset [7] are also provided, enabling the assessment of the method within an actual 360 ° setup.

A. EXPERIMENTAL SETUP

Our implementation is built on top of the Detectron2 open-source framework [43], based on PyTorch. Thus, we take advantage of the various optimizations introduced in this framework over the original Faster R-CNN implementation [3]. We performed our experiments in a computer equipped with an Intel Core i9-7900X CPU, 32 GB RAM, and an NVIDIA Titan Xp GPU. Unless otherwise stated, models were trained through stochastic gradient descent (SGD) with a batch size of 4 and a learning rate of 0.01, subject to step-wise decay.

The experimental analysis reported in this section can be divided into two main parts: on the one hand, a series of ablation studies aimed to evaluate the effect of different framework components; on the other hand, a complete evaluation of the approach's overall performance. The former uses the portion of the KITTI object detection benchmark for which labels are publicly available and divides it into training/validation sets according to the split in [36]. The latter additionally employs the official testing set to establish a fair comparison with other methods in the literature. The nuScenes dataset is considered in a second stage to further assess the universality of the approach.

Evaluation based on the KITTI dataset follows the rules of the BEV detection and 3D detection tasks in the benchmark, which currently rely on the following definition of average precision (AP) for a given category (c):

$$AP_c = \frac{1}{40}\sum_{r\in R} p_{\text{interp}}(r), \tag{12}$$

This AP_c is computed from the precision-recall curve as the average value, over 40 recall values $r \in R$, of the interpolated precision p_{interp}, derived from the precision p as

$$p_{\text{interp}}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}), \tag{13}$$

The assignment between detections and labels is performed according to IoU thresholds computed in 2D (in the BEV detection task) or 3D (in the 3D detection task). For the three categories defined in the evaluation, *Car*, *Pedestrian*, and *Cyclist*, this threshold is set to 70%, 50%, and 50%, respectively.

On the other hand, nuScenes evaluation follows the lines of the official detection task, which uses a mean average precision (mAP) computed in a significantly different way: detections are assigned to ground-truth instances according to the 2D BEV distance between centers, d, and the AP is computed by integrating the precision-recall curve for values over 0.1; then, APs are averaged over the whole set of categories, C, and four different values of $d \in D$, with $D = \{0.5, 1, 2, 4\}$:

$$mAP = \frac{1}{|C|\,|D|} \sum_{c \in C} \sum_{d \in D} AP_{c,d}, \tag{14}$$

Reportedly [7], this variant is less sensitive to small localization errors in the detections and decouples detection from object size and orientation estimation.

Furthermore, the nuScenes benchmark introduces a series of true positive metrics aimed to describe the detection quality in terms of translation, scale, and orientation, among others. We will adopt here those relevant to our task, which are listed below:

- Average translation error (ATE), representing the Euclidean distance between centers in BEV in meters.
- Average scale error (ASE), computed as $1 - IOU$ after aligning centers and orientation.
- Average orientation error (AOE), which is the smallest yaw angle difference between prediction and ground-truth in radians.

B. ABLATION STUDIES

This section will introduce a set of ablation studies to explore the effect of various alternatives for the input representation and the design of the detector's architecture. As stated before, the train/validation subsets of the publicly available KITTI dataset are used for model training and evaluation. All models were trained for 20k iterations.

1) BEV INPUT REPRESENTATION

At this point, we further investigate two determining aspects of the input representation that are directly related to the method's performance: the size and resolution of the BEV image and the set of features encoded in each of its cells.

As mentioned in Section III, the choice of the grid map extension is subject to a trade-off between the maximum detection distance and the processing time per frame, even though the upper bound is determined by the LiDAR scanner resolution and layer distribution.

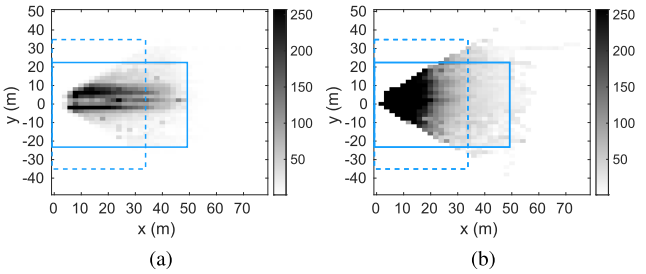


FIGURE 7. Spatial distribution of labeled objects in the KITTI dataset (Hard difficulty), represented in grid maps with 2 × 2 m cells: (a) Total number of objects. (b) Mean number of 3D points per object. In both cases, rectangles in blue represent the area spanned by the former 35 × 70 m BEV representation (dashed line), and the proposed BEV 50 × 45 m representation (solid line).

Although the choice depends primarily on the application, we have investigated the spatial distribution of object annotations in the KITTI dataset to optimize the size of the BEV representation used in the experiments. Fig. 7 represents both the number of objects (located by their centroid) and the average number of LiDAR points by which they are represented in the region in front of the vehicle, where KITTI annotations are available.

The area covered by the 35×70 m BEV representation used in [8], [9] is depicted as a dashed-line rectangle. As can be seen, that crop misses a significant portion of objects present in the $x \in [35, 50]$ range, which are still represented by enough LiDAR points, while devoting many data to render the scarcely populated areas at both sides. Overall, roughly 80% of object centroids (Hard difficulty) are contained in this region, placing an upper bound on detection performance. On the contrary, we propose the 50×45 m region delimited by the solid-line rectangle, which contains most objects (approx. 98% centroids) while using fewer data.

Therefore, in our experiments on the KITTI dataset, we use a 1000 × 900 BEV image representing a 50 × 45 m area with a 0.05 m cell resolution. In the general case, the optimal size of the BEV image should be selected according to the particularities of the sensor and the application requirements.

Apart from the BEV representation size, information available for inference is determined by the features encoded in each cell. The baseline described in Section III offers a compact and descriptive representation made of three channels, namely maximum height (H_{max}), mean intensity (I), and density (D); however, we studied other viable alternatives through a set of ablation experiments on the composition of the BEV image, with a threefold objective. First, we aim to isolate the effect of each of these channels on the detection performance. Second, we investigated the effect of two additional features: minimum height, H_{min}, and occupancy, O. The former encodes the height of the lowest point in a cell, normalized over a $[H_{\text{bottom}}, H_{\text{top}}]$ range (with $H_{\text{bottom}} = -3$ m and $H_{\text{top}} = 3$ m) with respect to the

TABLE 1. BEV and 3D detection performance (AP BEV % and AP 3D %) on the KITTI validation set (Moderate difficulty) for different sets of channels.

BEV input		Car		Pedestrian		Cyclist	
Features (and # vertical slices)	N_{ch}	BEV	3D	BEV	3D	BEV	3D
$I(1), D(1), H_{max}$ (baseline)	3	82.99	68.71	64.07	54.01	52.25	50.23
$I(1)$	1	65.17	32.60	45.15	28.29	38.12	28.35
$I(3)$	3	67.02	43.32	59.47	51.38	40.56	37.12
$I(12)$	12	68.01	49.51	61.63	54.58	42.02	39.10
$D(1)$	1	81.13	46.15	53.82	37.42	42.28	32.62
$D(3)$	3	80.82	64.19	60.50	53.25	48.89	42.83
$D(12)$	12	81.54	67.44	58.98	51.48	49.45	44.07
$O(1)$	1	78.75	48.19	48.60	32.36	39.06	29.91
$O(3)$	3	79.37	58.88	57.64	46.26	48.11	42.02
$O(12)$	12	83.83	70.34	62.59	55.51	46.13	46.67
H_{max}	1	81.96	69.85	58.92	50.57	45.42	42.99
H_{max}, H_{min}	2	81.52	68.81	56.30	48.65	50.15	48.51
$D(1), O(1), H_{max}$	3	81.86	68.74	57.06	50.12	47.07	43.69
$I(1), D(1), H_{max}, H_{min}$	4	81.18	67.80	63.79	57.06	51.33	48.81
$I(3), D(3), H_{max}$	7	82.03	68.26	62.43	55.03	50.58	48.51
$I(1), D(1), O(12), H_{max}, H_{min}$	16	83.20	69.59	60.68	54.83	50.20	47.07
$I(12), D(3), H_{max}$	16	84.59	70.45	60.76	53.26	53.68	51.09

hypothetical flat ground plane, whereas the latter is a binary feature that only indicates the mere presence of LiDAR points in the cell, as in [27], [44]. Finally, we explored the possibility of discretizing the space along the z-axis and representing the features of each of the vertical slices through a different BEV channel. This approach involves the division of the 3 m space above the floor that is considered in the computation of each cell into shorter subpillars. We tested two possibilities: three divisions of 1 m in height each, and 12 divisions of 25 cm each. It should be noted that this approach is suitable for I, D and O, but makes little sense for H_{max} and H_{min}, which are intended to contain the extreme values of the z coordinate within the whole pillar.

BEV and 3D detection results on the KITTI validation set for different combinations of these alternatives are shown in Table 1, where the number of vertical slices used by each feature is denoted in parenthesis. Results correspond to the Moderate difficulty, as defined in the KITTI benchmark.

Naturally, an increase in the total count of channels, indicated by the N_{ch} column, entails greater model complexity. Average processing times for each value of N_{ch} are shown in Table 2.

TABLE 2. Average runtime per frame (ms) vs. number of features included in the BEV input (N_{ch}).

N_{ch}	1	3	4	7	12	16
Runtime (ms)	112	115	116	129	132	142

According to the single-channel experiments, the H_{max} feature is significantly more relevant than I, D, and O; however, the baseline representation ($I(1)$, $D(1)$, H_{max}) still offers +3.2 mAP 3D over the single-channel H_{max} alternative.

On the other hand, the binary occupancy (O) feature shows remarkable representation capabilities, outperforming I when analyzed separately. However, the combination with $D(1)$ and H_{max} still yields a detection performance below the baseline. Meanwhile, H_{min} proves an interesting addition in some cases, providing complementary information to H_{max}. In fact, the encoding composed by the baseline channels plus H_{min} gives a modest improvement of +0.2 mAP 3D.

Finally, the division of the space in vertical slices largely improves the representation capabilities of the features when used separately. Particularly noteworthy is the $O(12)$ case, whose performance is equivalent to the baseline. Other combinations with a higher channel count (i.e., 7 or 16 channels) also increase the detection performance, but the magnitude of the offset (+0.6 mAP 3D in the best case, $I(12)$, $D(3)$, H_{max}) hardly justifies the complexity rise brought about by the new channels.

From now on, experiments focus on the 3-channel baseline configuration, which has been proven a sweet trade-off between detection and model complexity.

2) FEATURE EXTRACTOR

Based as it is on Faster R-CNN, our method allows different convolutional backbones to fulfill the role of feature extractor. Besides, features for both the RPN and the second detection stage can be sampled at different levels. Table 3 shows a comparison of the detection performance for two different ResNet backbones, ResNet-50 and ResNet-101, and two of the sampling strategies mentioned in Section IV-A: extracting features from the C3 layer (with 8× downsampling) and using an FPN that integrates information from three different levels.

These results show that the ResNet-50 with FPN outperforms the C3 version by a large margin in the Pedestrian category, made of smaller objects, while offering similar

TABLE 3. BEV and 3D detection performance (AP BEV % and AP 3D %) and runtime (ms) on the KITTI validation set (Moderate difficulty) for different ResNet backbones.

Backbone	Car		Pedestrian		Cyclist		T (ms)
	BEV	3D	BEV	3D	BEV	3D	
R-50-C3	83.9	69.1	53.7	44.4	51.3	47.9	58
R-50-FPN	83.0	68.7	64.1	54.0	52.3	50.2	116
R-101-FPN	82.3	67.8	60.9	51.4	52.9	50.5	136

TABLE 4. BEV and 3D detection performance (AP BEV % and AP 3D %) on the KITTI validation set (Moderate difficulty) for different sets of parameters estimated by the CNN (vs. manually estimated). 2D involves x, y, l, w, and θ_{bin}. BEV input resolution was 1400 × 700, with 0.05 m cell size.

CNN output			Car		Pedestrian		Cyclist	
2D	h, z	$\Delta\theta$	BEV	3D	BEV	3D	BEV	3D
*			63.0	35.4	58.4	48.0	42.7	38.4
✓			64.4	40.6	62.2	51.2	44.0	37.6
✓	✓		63.8	51.3	62.8	52.7	41.9	40.0
✓	✓	✓	66.3	55.6	61.7	52.4	44.9	42.6

* Axis-aligned 2D bounding box with an orientation estimate (from [8]).

performance in the other categories. On the other hand, the ResNet-101 does not significantly improve the 50-layer equivalent despite the difference in their representation capabilities. Therefore, the ResNet-50 with FPN proves the optimal choice for the application.

3) DETECTION BRANCHES

In [8], 3D detection was performed by a two-step approach in which a BEV-based CNN was responsible for providing axis-aligned detections that were later converted to 3D cuboids through a hand-crafted refinement procedure. This process involved the transformation of axis-aligned 2D detections (each with an estimate of its yaw angle) into rotated 2D boxes according to geometrical constraints, as well as providing an estimation of the vertical location and height of the obstacles based on a grid-based approximation of the ground position and a fixed height value per category.

The BirdNet+ framework [9], in which the current work is focused, replaced this whole manual refinement stage by embedding the inference of all the 3D cuboid parameters into the CNN. Table 4 shows the differences in BEV/3D detection performance when transitioning gradually from the original paradigm, where the CNN estimated only the parameters of the axis-aligned 2D boxes and a discrete orientation, to the full BirdNet+ scheme, where all the parameters are inferred by the network, including a refinement of the yaw angle based on the regression of a residual term $\Delta\theta$. Here, the original 1400×700 BEV input from [8] has been used for the sake of consistency.

The results demonstrate the suitability of the end-to-end approach over the alternative refinement steps. The largest improvement in 3D detection (+4.9 mAP 3D) occurs when z and h are introduced into the inference, suggesting the importance of an accurate estimation of the vertical coordinate in contrast to the usual flat ground and fixed height assumptions.

C. PERFORMANCE ASSESSMENT

The second group of experiments aims to assess the performance of the approach compared with other state-of-the-art methods. These tests were performed on both the KITTI object detection benchmark and the nuScenes dataset; the former allows fair comparison with numerous works in the literature, whereas the latter enables the evaluation of the proposed method as a 360° 3D detection framework.

1) KITTI BENCHMARK

Ablation studies in the previous section have already provided abundant evidence of the approach's detection performance on the KITTI validation subset under different configurations. In this section, the goal is twofold: first, the sensitivity of the approach to the distance from the sensor and the IoU threshold used in the evaluation will be studied; second, results on the official KITTI testing set will be presented, enabling fair comparison with other approaches.

Given the limited resolution of current scanners, LiDAR-only methods suffer from a lack of accuracy in long-range object detection. Table 5 provides an overview of the effect of the distance from the LiDAR device in the BEV and 3D detection performance on the KITTI dataset. It should be noted that the distance intervals are defined in terms of the Euclidean distance separating the sensor from the obstacles.

TABLE 5. BEV and 3D detection performance (AP BEV % and AP 3D %) on the KITTI validation subset (Moderate difficulty) regarding the distance from the LiDAR (in meters). The proportion of objects per category in the validation subset is included.

Dist.	% instances	Car		Pedestrian		Cyclist	
	Car/Ped./Cyc.	BEV	3D	BEV	3D	BEV	3D
0–10	13 / 30 / 13	90.7	86.8	78.7	73.1	72.9	71.2
10–20	28 / 43 / 26	93.9	83.6	55.2	43.8	78.4	76.4
20–30	26 / 20 / 27	85.5	71.0	35.3	25.9	54.5	51.9
30–40	19 / 5 / 19	69.5	49.6	13.5	12.4	46.6	40.2
40–50	14 / 2 / 15	45.2	20.5	1.4	1.1	20.4	17.8

Results prove that, with the Velodyne HDL-64 featured by the KITTI benchmark, the proposed method yields reasonable accuracy over a reasonably wide range of distances. As expected, AP for the Pedestrian category drops dramatically beyond 20 m–30 m due to their smaller size. However, the decline is smoother for *Car* and *Cyclist*, where detection performance is decent in the most densely populated areas, up to the 40 m–50 m range. In general, detection accuracy depends strongly on the number of grid cells representing an object in the BEV, as shown in Fig. 8, where the detection performance is represented as a function of the number of cells spanned by an object on average for each of the Euclidean distance ranges in Table 5. Hence, it is reasonable

TABLE 6. BEV and 3D detection performance (AP BEV % and AP 3D %) of different approaches on the KITTI testing set, including methods using LiDAR (L) and RGB data (I). LiDAR data can be used as bird's eye view (BEV), range view (RV) or voxelized (vox.).

Category	Method	Data	AP 3D (%)			AP BEV (%)			T (ms)
			Easy	Mod.	Hard	Easy	Mod.	Hard	
Car	MODet [27]	L (BEV)	-	-	-	90.80	87.56	82.69	50
	PIXOR++ [45]	L (BEV)	-	-	-	93.28	86.01	80.11	35
	AVOD-FPN [37]*	I + L (BEV)	83.07	71.76	65.73	90.99	84.82	79.62	100
	MV3D (L) [36]	L (BEV+RV)	68.35	54.54	49.16	86.49	78.98	72.23	240
	C-YOLO [39]	I + L (vox.)	55.93	47.34	42.60	77.24	68.96	64.95	60
	TopNet-Ret. [26]	L (BEV)	-	-	-	80.16	68.16	63.43	52
	BirdNet [8]	L (BEV)	40.99	27.26	25.32	84.17	59.83	57.35	110
	BirdNet+	L (BEV)	76.15	64.04	59.79	87.43	81.85	75.36	115
Pedestrian	AVOD-FPN [37]*	I + L (BEV)	50.46	42.27	39.04	58.49	50.32	46.98	100
	C-YOLO [39]	I + L (vox.)	17.60	13.96	12.70	21.42	18.26	17.06	60
	TopNet-Ret. [26]	L (BEV)	-	-	-	18.04	14.57	12.48	52
	BirdNet [8]	L (BEV)	22.04	17.08	15.82	28.20	23.06	21.65	110
	BirdNet+	L (BEV)	41.55	35.06	32.93	48.9	42.87	40.59	115
Cyclist	AVOD-FPN [37]*	I + L	63.76	50.55	44.93	69.39	57.12	51.09	100
	TopNet-Ret. [26]	L (BEV)	-	-	-	47.48	36.83	33.58	52
	C-YOLO [39]	I + L (vox.)	24.27	18.53	17.31	32.00	25.43	22.88	60
	BirdNet [8]	L (BEV)	43.98	30.25	27.21	58.64	41.56	36.94	110
	BirdNet+	L (BEV)	65.67	53.84	49.06	70.84	59.58	54.2	115

* AVOD makes use of two separate models: one for *Car* and another for *Pedestrian* and *Cyclist* detection.

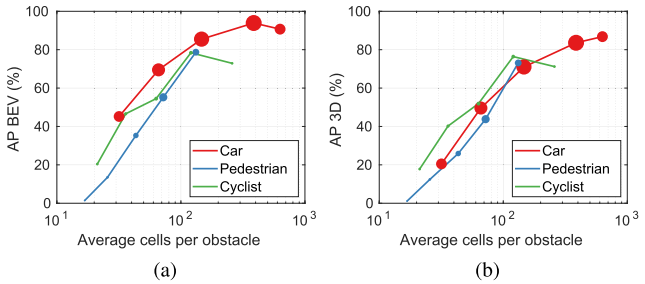


FIGURE 8. Detection performance vs. average number of cells spanned by an object on the KITTI validation subset (Moderate difficulty): (a) AP BEV (%). (b) AP 3D (%). The radius of each marker is proportional to the number of objects.

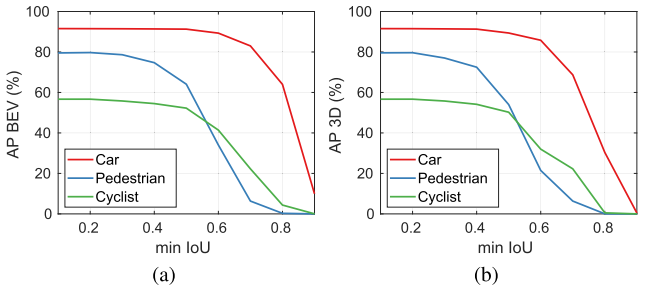


FIGURE 9. Detection performance vs. IoU threshold on the KITTI validation subset (Moderate difficulty): (a) AP BEV (%). (b) AP 3D (%).

to assume that pedestrian and cyclist detection performance could be significantly improved using a higher resolution LiDAR scanner.

Experiments up to this point have used the strict IoU thresholds established in the KITTI benchmark for the association between detections and ground-truth annotations, as stated before. However, Fig. 9 offers a complementary perspective by providing the AP BEV and AP 3D stats for each category in a wide range of IoU thresholds. The results evidence that a significant number of detections across all the categories are slightly mislocalized, thus not being considered under the official criteria even though they achieve decent overlaps (mostly around 0.3–0.4) with the ground-truth labels. The effect is especially notable in the Pedestrian category, where both AP stats (BEV and 3D) exceed 75% when the threshold is lowered to 0.3.

Results on the KITTI testing set (official benchmark) are presented in Table 6, together with those obtained by other comparable methods. In this case, a model trained on the full KITTI training set over 40k iterations was employed. As can be seen, our proposed framework outperforms other LiDAR-only approaches in the literature and yields a detection performance close to top-performing fusion pipelines, all at a framerate around 10 FPS.

It is worth mentioning that our method offers a key advantage that is relatively uncommon in related work: it can perform simultaneous detection of all the object categories with a single forward pass, as both the BEV representation and model training have been tuned to deal with the interclass variations.

Some examples of detection results on the KITTI dataset are depicted in Fig. 10. Our method can identify objects partially occluded (as in Fig. 10b) or difficult to spot (e.g., the pedestrians behind the tram stop in Fig. 10c). Distant agents can also be detected as long as they fall within the area covered in the BEV representation, as in Fig. 10c and 10d.

TABLE 7. Detection performance on the nuScenes validation set, using only one LiDAR sweep (1 sw.), using one LiDAR sweep but ignoring labels with less than 10 points (>10 p.) and using the past 10 LiDAR sweeps (10 sweeps).

Category	AP (%)			ATE (m)			ASE (%)			AOE (rad)*		
	1 sw.	>10 p.	10 sw.	1 sw.	>10 p.	10 sw.	1 sw.	>10 p.	10 sw.	1 sw.	>10 p.	10 sw.
Car	59.9	86.5	67.3	0.244	0.223	0.248	15.3	14.7	15.1	0.175	0.111	0.104
Truck	37.3	47.5	44.4	0.416	0.407	0.398	20.2	19.8	19.5	0.157	0.123	0.108
Bus	49.5	52.9	48.4	0.442	0.440	0.500	19.1	19.0	19.4	0.323	0.324	0.181
Trailer	25.2	30.1	32.3	0.703	0.695	0.609	23.0	22.9	22.3	0.666	0.659	0.772
Const. veh.	7.1	11.7	12.4	0.739	0.722	0.733	45.3	45.2	45.7	1.327	1.338	1.075
Pedestrian	43.4	57.8	48.6	0.220	0.218	0.346	28.7	27.0	28.4	1.325	1.294	1.187
Motorcycle	23.6	51.8	31.4	0.251	0.214	0.270	24.5	24.1	25.1	0.899	0.867	0.455
Bicycle	6.0	16.1	10.0	0.332	0.295	0.350	28.4	26.9	28.0	0.686	0.653	0.579
Traffic cone	11.3	36.4	18.9	0.312	0.260	0.238	35.9	32.4	33.5	-	-	-
Barrier	44.0	50.5	51.5	0.433	0.401	0.344	29.4	29.1	29.7	0.037	0.035	0.030
Mean	30.7	44.1	36.5	0.409	0.387	0.403	26.9	26.1	26.6	0.621	0.600	0.499

* Following the nuScenes benchmark, AOE is evaluated at 360° for all classes except barriers where it is only evaluated at 180°.

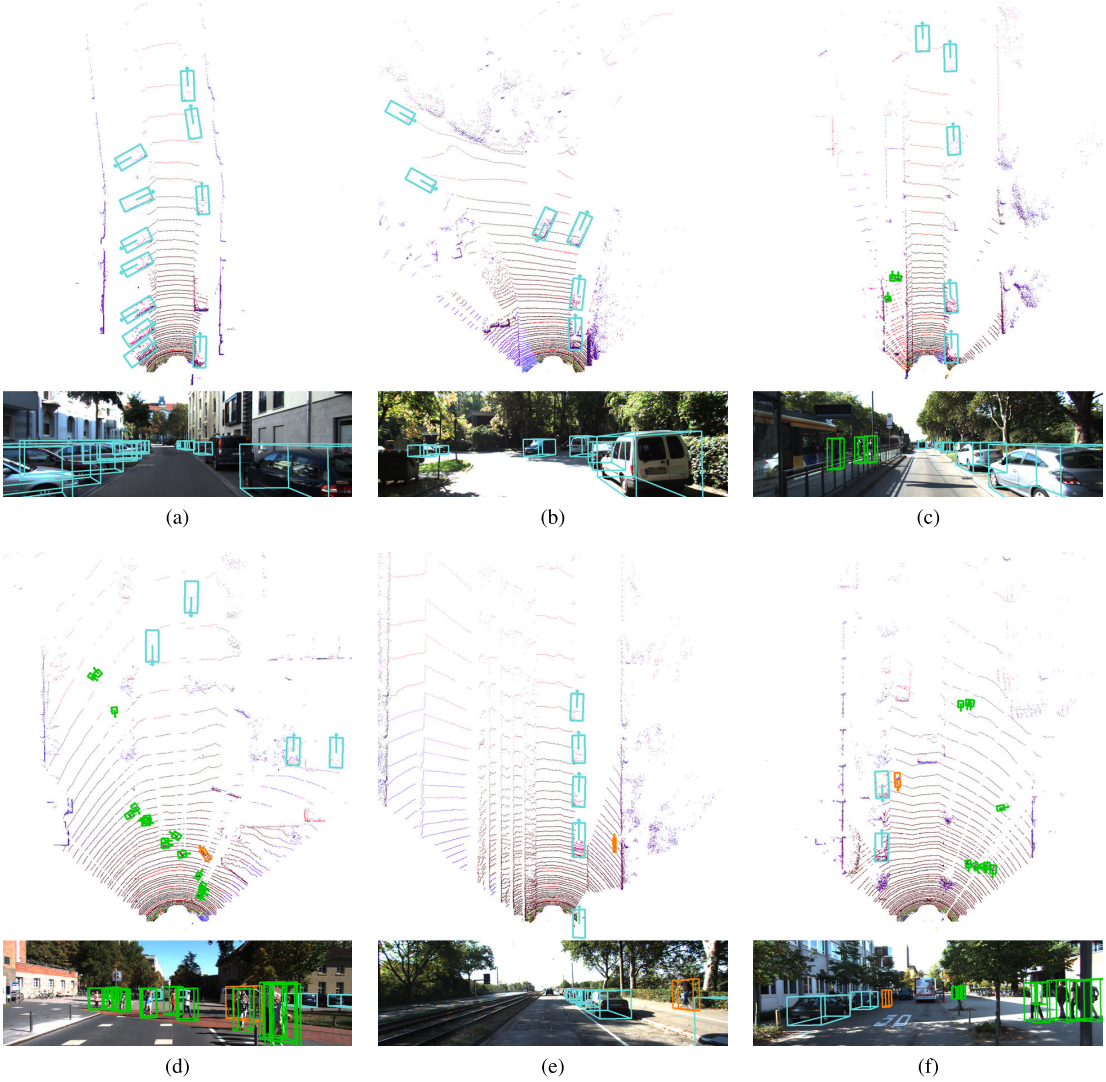


FIGURE 10. Results on the KITTI testing set. BEV detections are shown on the top row, while 3D boxes projected onto the front camera are displayed on the bottom row.

As apparent from the results, cars are generally well described by the resulting 3D bounding boxes and detected with a very high recall; note that other vehicle categories such as trams or vans are not considered. On the other hand, the multiclass capabilities are proven in Fig. 10d-10f, where pedestrians (including children) and cyclists are correctly identified even

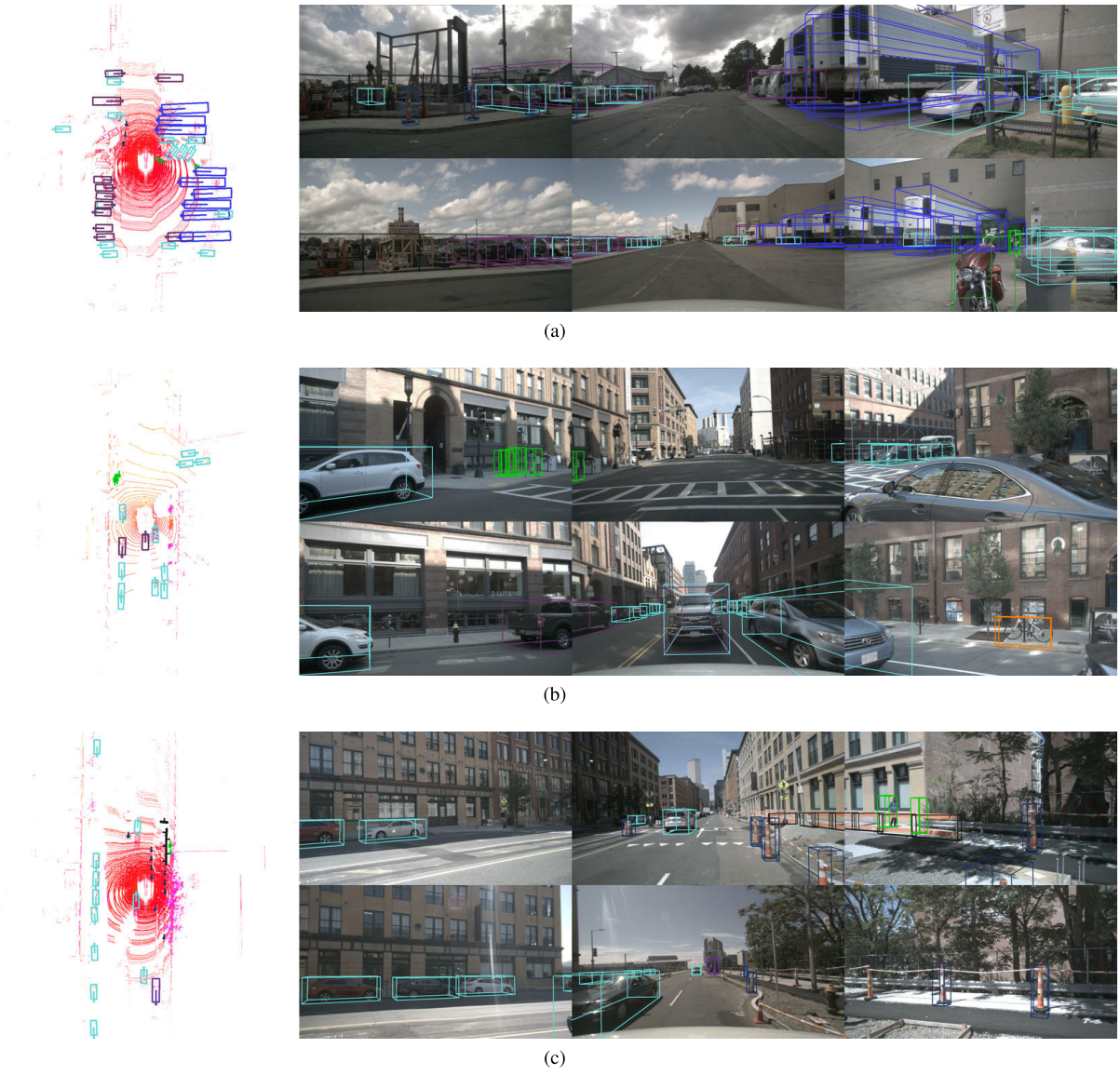


FIGURE 11. Results on the nuScenes test set. Detections in the BEV are shown on the left. The RGB images correspond to the front-left, front, front-right, rear-left, behind, and rear-right cameras, respectively.

when they are over a sidewalk. In general, the results show that our method performs well on the challenging scenes of the KITTI test dataset.

2) nuScenes DATASET

The nuScenes dataset, released in 2019 [7], offers a comprehensive set of 3D annotations representing obstacles in a 360° range around the vehicle. However, the low resolution of the Velodyne HDL-32 used to retrieve the data poses a serious challenge for training and validation of LiDAR-based detection methods. A significant fraction of labeled objects are represented by a few LiDAR points, thus making it difficult (or, in some instances, impossible) to detect them using only this modality. Moreover, the nuScenes detection challenge considers ten different semantic categories, as diverse as *construction vehicle*, *motorcycle*, or *traffic cone*.

We have applied our BirdNet+ method on the nuScenes dataset, with some adaptations. First, as we aim to perform 360° 3D detection, the BEV was modified to represent 51 m in all directions from the ego-car (located now at the center of the BEV). Due to memory constraints, cell size was increased to 10 cm so that the resolution of the BEV image becomes 1020×1020. As with KITTI, this representation comprises most of the objects considered in the evaluation.

Notably, the nuScenes' LiDAR scanner is placed with a non-negligible rotation with respect to the ground plane that was compensated using the calibration parameters provided for each frame. We also increased the maximum height considered in the BEV ($H_{\max}$) to 4 meters to account for the

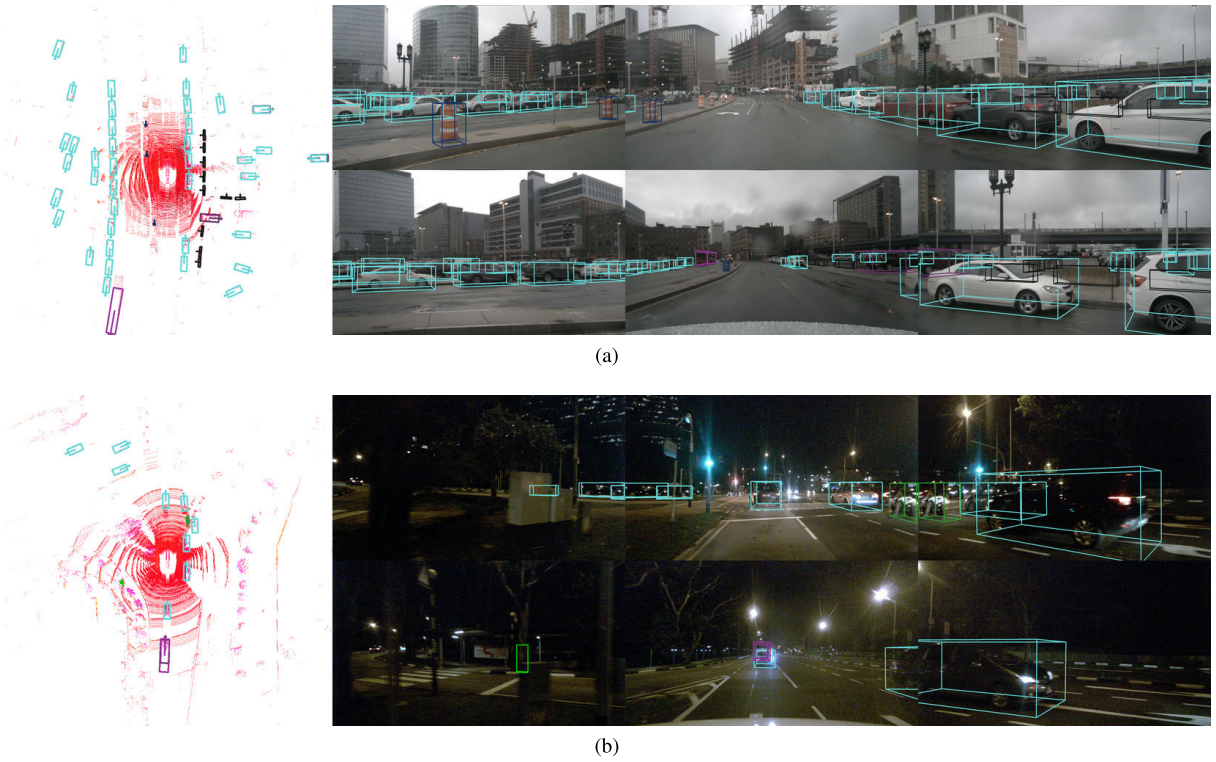


FIGURE 12. Results of BirdNet+ with rainy and night conditions on the nuScenes test set. Detections in the BEV are shown on the left. The RGB images correspond to the front-left, front, front-right, rear-left, behind, and rear-right cameras, respectively.

new large-size categories (e.g., trucks). Hardly visible objects represented by less than either five LiDAR points or four BEV cells are discarded for training.

We perform two kinds of experiments on the nuScenes validation set. First, we adopt a single-frame alternative that considers only the LiDAR data in the keyframes used for evaluation. Here, we additionally explore the effect of removing the most challenging instances (those with less than 10 points) from the evaluation. Later, as encouraged by the nuScenes benchmark, we aggregate the past 10 LiDAR sweeps (0.5 s) for each keyframe. To consider this accumulation in the construction of the density channel, we compute local $N_{\max}$ maps for each of the sweeps as described in Section III-B and add them all up after expressing them in the keyframe coordinates so that the value in each cell is normalized by the actual maximum number of points. Models were trained for 200k iterations, with all other parameters remaining the same as in the KITTI configuration.

Table 7 shows the detection performance on the validation set for the alternatives discussed above. Results for a single sweep show significant differences between classes, with some categories such as *car* or, remarkably, *pedestrian* achieving AP values over or near 50%, whereas others, such as *construction vehicle* or *bicycle*, are more problematic, probably due to their high intra-class variability.

The effect of removing poorly represented objects (>10 p.) is very significant (+13.4 mAP points), further confirming the considerable difficulty that this benchmark poses for LiDAR-only methods.

Finally, when the past 10 LiDAR sweeps are considered, AP values experience an overall increase that affects both small (*bicycle*, *traffic cone*) and large objects (*car*, *truck*). Besides, the orientation error (AOE) is significantly reduced. ATE and ASE remain virtually unchanged, which shows that true positives are equally well detected as in the case of a single sweep (even though there are more of them). This approach is standard practice in the nuScenes benchmark to overcome the lack of resolution of the LiDAR scanner [7], so we adopt this model as our reference.

Results on the official nuScenes testing benchmark were also obtained with a model trained on the trainval set for 240k iterations. Per-category stats are shown in Table 8. The results are consistent with those obtained in the validation set but surpass them by +2.7 mAP, probably due to the higher amount of training data, which enhances the generalization capabilities of the model.

Additionally, Table 9 provides a comparison with other real-time methods which take BEV images as their unique input. As can be observed, the performance of the presented network is on par with other LiDAR-based state-of-the-art approaches, offering a well-balanced accuracy in location, size, and orientation estimation.

Fig. 11 depicts some examples of detection results on the nuScenes dataset. The selected cases represent challenging

TABLE 8. Detection performance on the nuScenes test set (10 LiDAR sweeps).

Category	AP (%)	ATE (m)	ASE (%)	AOE* (rad)
Car	67.7	0.239	14.6	0.105
Truck	43.6	0.389	19.1	0.096
Bus	39.7	0.518	18.2	0.190
Trailer	47.2	0.561	17.6	0.853
Const. vehicle	16.3	0.685	35.6	1.068
Pedestrian	48.7	0.329	30.2	1.099
Motorcycle	28.9	0.266	24.7	0.521
Bicycle	11.0	0.363	29.5	0.648
Traffic cone	28.0	0.233	35.9	-
Barrier	60.5	0.347	24.7	0.027
Mean	39.2	0.393	25.0	0.512

* Following the nuScenes benchmark, AOE is evaluated at 360° for all classes except barriers where it is only evaluated at 180°.

TABLE 9. Detection performance of different LiDAR-based approaches on the nuScenes testing set.

Method	mAP (%)	mATE (m)	mASE (%)	mAOE (rad)
PointPillars [28]	30.5	0.517	29.0	0.500
SARPNET [46]	32.4	0.400	24.9	0.763
IntoFocus [47]	39.5	0.363	26.5	1.132
PointPillars+ [48]	40.1	0.392	26.9	0.476
Ours	39.2	0.393	25.0	0.512

* Following the nuScenes benchmark, AOE is evaluated at 360° for all classes except barriers where it is only evaluated at 180°.

scenarios, such as Fig. 11a, where cars, trailers, and trucks, parked at a different ground level on both sides of the road, appear next to several cases of minority classes (pedestrian, motorcycle, and traffic cones). Fig. 11b illustrates the extensibility of our method to those keyframes where only one sweep of the LiDAR is available. Furthermore, together with Fig. 11c, they demonstrate the adaptability of our method to different sizes and shapes of vehicles (e.g., bicycles and trucks), even with distant agents. Besides, cones and barriers are adequately detected and localized as long as they are reasonably high. Despite some rare confusions (e.g., detection of a double pedestrian due to the accumulation of multiple sweeps in Fig. 11c), the proposed method proves convenient for scenarios with multiple kinds of objects.

Finally, in Fig. 12, we further confirm the adaptability of our method to diverse weather conditions and night scenarios where camera methods often fail to perform well. Our method shows high robustness and can accurately estimate the location and dimensions of multiple agents such as cars, motorcycles, traffic cones, and buses, even with decreased visibility.

VI. CONCLUSION

A complete 3D detection framework aimed at on-board perception has been presented in this paper. First of all, the method proves the feasibility of employing a bird's eye view representation of LiDAR data to perform 3D object detection. This structure enables the use of meta-architectures originally intended for detection in RGB images, which have been extensively validated in the literature and can achieve high accuracy levels in the range of distances where LiDAR data is dense enough.

In this regard, a novel encoding for the BEV representation, agnostic to the LiDAR device in use, has been introduced to ensure data homogeneity throughout the detection range. This encoding has proven effective in expressing different kinds of objects found in traffic environments.

This BEV constitutes the input of a 3D detection approach that adopts a series of novel detection heads responsible for estimating all the parameters representing the pose, dimensions, and semantic classification of an object. In conjunction with an RPN trained to generate candidate proposals directly from the LiDAR BEV, these branches enable the end-to-end identification of the relevant objects in the surroundings of a vehicle.

Combining these two contributions makes it possible to perform 3D detection over a set of multiple classes with a single model and a single BEV representation. The resulting framework complies with the real-time constraints of on-board perception systems and can provide reliable detections using LiDAR data exclusively. The validity and applicability of the approach have been assessed in a comprehensive set of experiments on the challenging KITTI and nuScenes benchmarks. Besides, the source code of the implementation has been released to promote reproducibility and further research.

Nevertheless, the proposed method still presents some limitations. On the one hand, the detection performance on small objects (e.g., traffic cones) is highly dependent on the bird's eye view resolution: if the space is discretized into overly coarse cells, the features of the less voluminous elements may be blurred. On the other hand, the uneven number of annotations among the different categories in existing datasets negatively affects the RPN, as the number of suitable candidate proposals for underrepresented objects is significantly lower than those available for the most common categories, leading to biased performance.

In order to address these issues, further future developments will be considered. First, features from the raw point cloud data may be exploited in the second detection stage, where box parameters are inferred from object candidates. In that way, the proposed BEV representation would be used to estimate an initial set of proposals, while the inference of the final 3D boxes would take advantage of geometrical information currently lost by the input voxelization of the top-view projection. Second, the current region proposal network may be endowed with a class-aware similarity attention mechanism, as in [49], instead of the current foreground-background classification that severely penalizes the underrepresented categories. This will not only mitigate the class imbalance on the variety of object candidates, but will also

help the RPN to filter out objects of non-labeled categories, reducing the number of false-positive proposals.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

REFERENCES

[1] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3D object detection leveraging accurate proposals and shape reconstruction," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, 2019, pp. 11867–11876.

[2] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for LiDAR point clouds in autonomous driving: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021.

[3] 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, Jun. 2017.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 779–788.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 652–660.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 3354–3361.

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 11621–11631.

[8] 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 *Proc. IEEE Intell. Transp. Syst. Conf.*, Nov. 2018, pp. 3517–3523.

[9] A. Barrera, C. Guindel, J. Beltrán, and F. García, "BirdNet+: End-to-end 3D object detection in LiDAR bird's eye view," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Rhodes, Greece, 2020, pp. 2985–2990.

[10] S. Royo and M. Ballesta-Garcia, "An overview of LiDAR imaging systems for autonomous vehicles," *Appl. Sci.*, vol. 9, no. 19, p. 4093, Sep. 2019.

[11] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, Aug. 2020.

[12] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, "SE-SSD: Self-ensembling single-stage object detector from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14494–14503.

[13] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 918–927.

[14] H. Zhang, D. Yang, E. Yurtsever, K. A. Redmill, and O. Ozguner, "Faraway-frustum: Dealing with lidar sparsity for 3D object detection using fusion," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Indianapolis, IN, USA, Sep. 2021, pp. 2646–2652.

[15] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018, *arXiv:1812.05276*.

[16] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 770–779.

[17] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 1951–1960.

[18] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4490–4499.

[19] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[20] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, "Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds," *Sensors*, vol. 20, no. 3, p. 704, Jan. 2020.

[21] Z. Liu, Z. Zhao, T. Huang, R. Hu, and X. Bai, "TANet: Robust 3D object detection from point clouds with triple attention," in *Proc. AAAI Conf. Artifi. Intel.*, New York, NY, USA, 2020, vol. 34, no. 7, pp. 11677–11684.

[22] G. Wang, J. Wu, B. Tian, S. Teng, L. Chen, and D. Cao, "CenterNet3D: An anchor free object detector for point cloud," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 15, 2021, doi: 10.1109/TITS.2021.3118698.

[23] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3D object detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3164–3173.

[24] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-YOLO: An Euler-region-proposal for real-time 3D object detection on point clouds," in *Proc. Eur. Conf. Comput. Vis. Workshops*, Munich, Germany, 2018, pp. 197–209.

[25] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 7652–7660.

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

[27] Y. Zhang, Z. Xiang, C. Qiao, and S. Chen, "Accurate and real-time object detection based on Bird's eye view on 3D point clouds," in *Proc. Int. Conf. 3D Vis. (3DV)*, Québec City, QC, Canada, Sep. 2019, pp. 214–221.

[28] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, 2019, pp. 12697–12705.

[29] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 12677–12686.

[30] Z. Liang, M. Zhang, Z. Zhang, X. Zhao, and S. Pu, "RangeRCNN: Towards fast and accurate 3D object detection with range image representation," 2020, *arXiv:2009.00206*.

[31] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 3296–3305.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2015, pp. 1–14.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.

[34] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2117–2125.

[35] J. Ma, W. Shao, H. Ye, L. Wang, and H. Wang, "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Trans. Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov. 2018.

[36] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 6526–6534.

[37] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Madrid, Spain, 2018, pp. 5750–5757.

[38] C. Guindel, D. Martin, and J. M. Armingol, "Fast joint object detection and viewpoint estimation for traffic scene understanding," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 4, pp. 74–86, Sep. 2018.

[39] M. Simon, K. Amende, A. Kraus, J. Honer, T. Sämann, H. Kaulbersch, S. Milz, and H. M. Gross, "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Salt Lake City, UT, USA, 2019, pp. 1–10.

[40] N. Cheng, X. Li, S. Lei, and P. Li, "BVNet: A 3D end-to-end model based on point cloud," in *Proc. Int. Symp. Vis. Comput.*, San Diego, CA, USA, 2020, pp. 422–435.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, 2009, pp. 248–255.
[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, Sardinia, Italy, 2010, pp. 249–256.
[43] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. (2019). *Detectron2*. [Online]. Available: https://github.com/facebookresearch/detectron2
[44] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 1513–1518.
[45] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proc. Conf. Robot. Learn.*, Zürich, Switzerland, 2018, pp. 146–155.
[46] Y. Ye, H. Chen, C. Zhang, X. Hao, and Z. Zhang, "SARPNET: Shape attention regional proposal network for LiDAR-based 3D object detection," *Neurocomputing*, vol. 379, pp. 53–63, Dec. 2020.
[47] J. Wang, S. Lan, M. Gao, and L. S. Davis, "InfoFocus: 3D object detection for autonomous driving with dynamic information modeling," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 405–420.
[48] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 4604–4612.
[49] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai, "Few-shot object detection with attention-RPN and multi-relation detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 4013–4022.

ALEJANDRO BARRERA received the M.Sc. degree in robotics and automation from the Carlos III University of Madrid, Spain, in 2018, where he is currently pursuing the Ph.D. degree in electrical engineering, electronics and automation. His research interests include perception systems based on deep learning algorithms for 3D scene understanding and domain adaptation techniques between simulation and real environments.

JORGE BELTRÁN received the M.Sc. degree in robotics and automation from the Universidad Carlos III de Madrid, Spain, in 2016, where he is currently pursuing the Ph.D. degree in electrical, electronic and automatic engineering. His research interests include the field of perception systems for autonomous vehicles, focusing on multi-modal sensor calibration, sensor fusion, and 3D object detection using deep neural networks.

CARLOS GUINDEL received the Ph.D. degree in electrical engineering, electronics and automation from the University Carlos III of Madrid, Spain, in 2019. He is currently a Postdoctoral Researcher with the University Carlos III of Madrid. His research interests include the application of deep learning techniques to the intelligent transportation systems field covering topics, such as object detection, pose estimation, and sensor fusion.

JOSÉ ANTONIO IGLESIAS (Member, IEEE) received the B.S. degree in computer science from Valladolid University, in 2002, and the Ph.D. degree in computer science from the Carlos III University of Madrid (UC3M), Spain, in 2010. He is currently an Associate Professor at UC3M. He has published more than 50 peer-reviewed papers (Best Paper Award 2009 SSCI—ESDIS) and 20 journal articles. He received the Extraordinary Ph.D. Thesis Award in his Ph.D. degree. He is the Chair of the IEEE CIS Task Force (TF) on Adaptive and Evolving Fuzzy Systems and the Vice Chair of the IEEE TF on Autonomous Learning, Neural Network Technical Committee. He is a member of the editorial boards of three international scientific journals.

FERNANDO GARCÍA (Member, IEEE) received the Ph.D. degree in electrical, electronic and automatic engineering from the Universidad Carlos III de Madrid, in 2012. He currently works as an Associate Professor with the Universidad Carlos III de Madrid. His research interests include perception and data fusion, mainly applied to vehicles and robotics. He has been a member of the Board of Governors of the IEEE-ITS Society, since 2017, and the vice chair of the Spanish chapter, for the period 2021–2022.