
EPiC Series in Computing

Volume 91, 2023, Pages 42–55

Proceedings of 38th International Conference on Computers and Their Applications


# 3D Objects Detection and Recognition from Color and LiDAR Data for Autonomous Driving

Lian Kang and Pierre Payeur
University of Ottawa, Ottawa, ON, Canada
lkang018@uottawa.ca, ppayeur@uottawa.ca

**Abstract**

In recent years, autonomous driving vehicles are attracting growing commercial and scientific attention. How to detect and recognize objects in a complex real-world road environment represents one of the most important problems facing autonomous vehicles and their ability to make decisions on the road and in real time. While color imaging remains a rich source of information, LiDAR scanners can collect high quality data under different lighting conditions and can provide high-range and high-precision spatial information. Expanding object detection by processing simultaneously data collected by a color camera and a LiDAR scanner brings new capabilities to the field of autonomous driving. In this paper, a 3D object detector is proposed with focal loss and Euler angle regression to optimize the detector's performance. It uses a bird's-eye view map generated from a LiDAR point cloud and RGB images as input. Results show that the proposed 3D object detector reaches a speed over 46 frames per second and an average precision over 90%. In addition, a more compact detector is also proposed that processes the same input data three times faster with only slightly lower accuracy.

## 1 Introduction

Autonomous driving vehicles have been part of people's vision of the future. With the rise of artificial intelligence in recent years, autonomous driving received significant investment from companies. One of the critical tasks involved in autonomous driving is to detect passing pedestrians, vehicles, and other objects on the road, so that the vehicle can make corresponding driving decisions and ensure public safety. These requirements lead to the development of efficient object detection and recognition technologies.

The development of deep learning methods brought unprecedented progress in the field of objects detection and recognition. Recent research shows that bringing depth information into the detection model can increase the detector's performance and supersede the traditional 2D mapping for

autonomous driving. Compared to stereo cameras and active depth sensors, light detection and ranging (LiDAR) scanners performance is not significantly affected by ambient lighting conditions, and as a result they can provide consistent high-precision spatial information. Hence, they became increasingly popular for 3D object detection tasks in outdoor environments.

While the point cloud collected by LiDAR scanners can reflect the shape and posture information of the objects in the real world, it does not carry texture or color information. Therefore, the point cloud collected by a LiDAR scanner often needs to be preprocessed and combined with RGB images for the detector to fully leverage 3D location, as well as color and texture information.

The main contributions of this research include the merge of Euler angle's regression to DarkNet-53 [1] convolutional neural networks for defining a 3D object detector to classify and localize cars, pedestrians, and cyclists from LiDAR point clouds and RGB images in real-world road scenes. To reduce calculation and memory usage during training and testing, the LiDAR point cloud is converted into a bird's-eye view (BEV) map using coordinate systems transformation and height thresholding. Finally, the proposed architecture is adapted to merge a focal loss [2] and a generalized intersection over union (GIoU) loss [3] with the objective to handle biased data and optimize the proposed model. The solution is trained, and its object recognition performance is tested on real-world data provided by the KITTI vision benchmark suite [4].

# 2 Technical Background and Literature Review

## 2.1 LiDAR Technology

To provide reliable navigation for autonomous vehicles, as well as vision support for safe driving and decision-making, on-board sensors are needed to provide highly accurate and informative data about the environment as well as precise object positioning. Compared to using RGB cameras to take pictures, LiDAR technology is not sensitive to variations in lighting conditions and can work over day and night, even with glare and shadows. The LiDAR scanner used to collect point clouds for the KITTI dataset [4] and considered in this research is the Velodyne's HDL-64E. It is a 64 channels multi-beam mechanical LiDAR scanner, that continuously rotates the head to achieve dynamic 3D scanning. It covers a 360° horizontal and 26.9° vertical field of view [4]. Although the data provided by LiDAR reports on accurate 3D location, it does not contain color information. Therefore, in this work, both color images provided by a collocated RGB camera and 3D point clouds provided by the LiDAR are used.

## 2.2 3D Object Detectors using LiDAR Point Cloud

3D object detection and recognition require not only the traditional RGB or grayscale image, but also depth information, that is, the position coordinates of each pixel in space. For this reason, 3D detection tasks usually require larger and more complex training and testing datasets, which leads to higher requirements in data processing and computing capabilities.

3D object detectors can be divided into two-stage or single-stage solutions based on the method they use. Popular two-stage detectors include MV3D-Net [5] and AVOD [6]. MV3D-Net uses both RGB images and a LiDAR point cloud as input. The model combines a 3D object proposal network and a region-based fusion network to efficiently generate 3D candidate boxes over a bird's-eye view (BEV) and a front view, both extracted from the point cloud. AVOD and MV3D-Net hold many similarities. The main difference is that MV3D-Net uses a VGG16 based network for feature extraction, while AVOD uses feature pyramid networks (FPN), which can prevent the decrease of the resolution in the feature maps and keep low-level and high-level information. In this way, AVOD can significantly improve the detection accuracy on small objects.

One-stage detectors such as PIXOR [7] and PointPillars [8] also contributed to the development of 3D object detection. PIXOR discretizes the point cloud of a scene with equally spaced units and encodes the reflectivity in a similar manner to obtain a regular representation. Then, a fully convolutional network (FCN) is used to estimate the position and heading angle of the target with respect to the sensor. Its detection speed can reach over 28 frames per second (FPS). Unlike the way that PIXOR manually designs features, PointPillar directly stacks the points that fall into each grid and calls it a 'Pillar'. It then maps the learned feature vector back to the grid coordinates to get data similar to the image.

In summary, LiDAR point cloud-based 3D object detection algorithms mainly vary in the data preprocessing method and detector structures. For preprocessing the input data, compared with voxel-based methods or directly using point cloud, generating different 2D viewpoints from the original point cloud would be more efficient and more practical for applications without high computational power hardware. Furthermore, fusing the LiDAR point cloud and RGB images can provide the detector with both the 3D information and color features, which can improve the detector's performance. The optimization methods of the models include but are not limited to the fusion of various models, improvement of the loss function, and adjustment of the model depth. The proposed methodology in this work addresses both the data preprocessing and the optimization of the models.

# 3 Point Cloud Preprocessing

A 3D point cloud is the default representation of the data collected by a LiDAR scanner. There are two major ways to process a 3D point cloud: one is directly processing a 3D representation; another is to convert the 3D mapping into its equivalent 2D representation. In recent work using the LiDAR point cloud for object detection and recognition [9, 10, 11], researchers directly train the detector on the 3D point clouds [12] with the consequence that convolution operations take more time and memory compared to when information is encoded in 2D. Alternatively, some models like MV3D-Net [5] opt for converting the point cloud to a front view and a BEV map, which proves more efficient than processing the 3D point cloud directly, while not lowering the accuracy.

The BEV map is a graphical representation of the point clouds from a bird's-eye view. It is obtained by projecting the discrete LiDAR point cloud on a plane perpendicular to the height direction. Therefore, a BEV map forms an equivalent representation of the 3D location information contained in the LiDAR point cloud. The front forward view and color information is provided from corresponding RGB images. Therefore, all the required information can be obtained by combining the BEV maps converted from LiDAR point clouds and the RGB images. This section explains how the LiDAR point clouds and RGB images are preprocessed and converted into BEV maps.

1. Data Registration

The 3D point cloud and RGB images are obtained from different sensors and must be registered before the two sets of data are used together as input. With the help of the calibration matrices from the dataset, the registration of the LiDAR point clouds and RGB images is performed using coordinate transformation to align LiDAR axes and origin to the RGB coordinates.

2. Mapping 3D Points within Region of Interest to 2D Pixels

In the dataset considered, the point cloud of each scene represents approximately 1.9 MB, which increases computation time for both training and testing and reduces detection efficiency. Therefore, it is useful to focus on a region of interest (ROI) in the point cloud. To balance the model's efficiency while covering all the annotated target objects in the corresponding RGB image, the ROI over the point cloud is manually set as a rectangle that spans 40m on either side of the LiDAR scanner, and 80m in front of it.

The point cloud data collected from the LiDAR scanner are 3D points with real (x, y, z) values that carry depth information. The registered points within the ROI are then mapped into integer values that represent the pixel location on the discretized bird's-eye view (BEV) map.

3. Recording the Height and Intensity Information

After the coordinates (x, y) that represent each pixel in the BEV map are obtained, the height information represented by the values on the Z-axis and the intensity information represented by the 4th values contained in the source point cloud are extracted.

Inspired by PIXOR [7], a vertical ROI on the Z-axis is selected to support a height thresholding operation that is applied to preserve only data from the point cloud that are within the selected height of the ROI. Next, the height coordinates (on Z-axis) within the ROI are rescaled into ]0, 255[, and the height coordinates outside the ROI are forced to 0 or 255. Finally, the height values of the points in the point cloud that are mapped into the same 2D pixel position on the BEV map are cumulated and recorded to the height channel of the BEV map. Compared with using the maximum height of each pixel position [13], the cumulation method appears to be less affected by the changes in elevation of the objects due to the slope of the road. Unlike MV3D-Net [5] that manually selects multiple ranges on the Z-axis and accumulate the values of the points within the ranges to generate multiple height channels for each 2D point in the BEV map, the proposed method performs a single height thresholding operation to create one height channel. This contributes to make the detection process more efficient.

Similarly, the intensity information contained in the point cloud as the 4th value for each 3D point contained within the selected height ROI and falling within the same 2D pixel position on the BEV map is accumulated and recorded to the intensity channel. The resulting preprocessed point cloud data is used as part of the input for the proposed 3D object detector.

# 4 Model Architecture

The proposed method for 3D object detection combines a BEV map generated from a LiDAR point cloud and the associated RGB image information as a single 5-channel (height, intensity, R, G, B) input for every 2D pixel coordinate in the BEV map.

The BEV map part of the input represents the bird's-eye view over the detection range, with a cumulated height channel and cumulated intensity channel. The RGB image is resized with bilinear interpolation and padded with $[R, G, B] = [128, 128, 128]$ to match the size of the BEV map. It forms the RGB part of the input, which represents the front forward view as found in autonomous driving, with three different color channels (R, G, B). Doing so preserves both the 3D information collected by the LiDAR scanner in the distribution of feature points over the 2D BEV map and the color information collected by the RGB camera.

The output of the proposed model represents the detection and recognition confidence over a number of object classes, with prediction matrices at three different scales. The prediction matrices are used to draw B-Boxes around detected target objects and to label their respective classification.

Given the importance of making quick decisions in autonomous driving, any improvement in object detection speed while maintaining high object detection accuracy is valuable. With this in mind, a single-stage detection model is proposed in this paper.

As shown in Figure 1, the backbone of the proposed object detection model is based on DarkNet-53 [1], modified to include the preprocessing stage of the BEV maps and the RGB images described in Section 3. The detection head is based on the YOLOv3 anchor regression method, modified by adding BEV variables and rotation angle regression. Inspired by complex YOLO [14], the rotation angle encoding uses Euler representation.

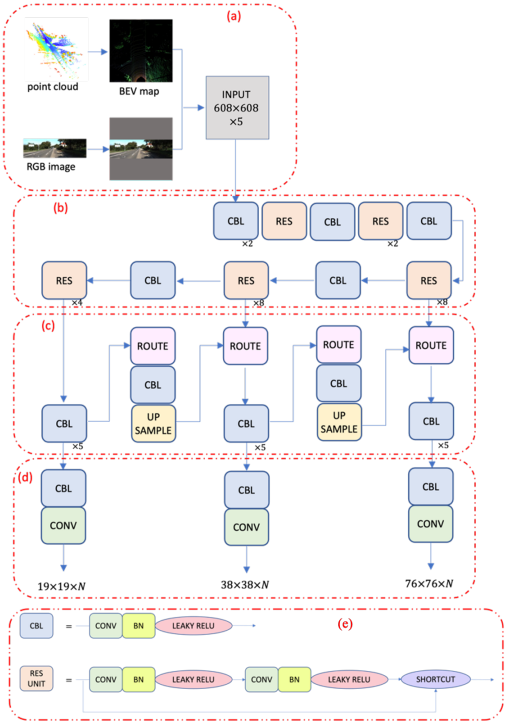


**Figure 1:** Proposed 3D object detection model structure: (a) preprocessing converts a 3D point cloud into a 5-channel 2D BEV map with rescaling of the corresponding RGB image, (b) structure of the backbone of the proposed model, (c) structure of feature pyramid network (FPN), (d) detection head which outputs prediction matrices at three different scales, and (e) respective structure of CBL (top) and res unit (bottom).

## 4.1 Detection Head with Euler Angle Regression

Through the backbone convolutional neural networks and the FPN, feature maps at three different scales are extracted from the input of the proposed model. As shown in Figure 1d, a detection head is used to generate the detection and recognition results based on these feature maps.

Within the detection head, each feature map will be divided into grid cells. For each grid cell, there are three anchors at different scales. Anchors are priors for bounding boxes (B-Box). They are equivalent to a reference frame for the predicted B-Box. Based on this reference, the predicted B-Box generated by the detection head only needs to be fine-tuned based on this anchor. For every anchor, there is a prediction matrix that contains the parameters used for regression during training and the detection result. The output prediction matrix of the proposed detection head contains the B-Box prediction matrix for both the RGB front forward view and the BEV, a confidence score, and classification scores over the 3 targeted object classes, namely car, pedestrian, or cyclist.

To adapt to the different viewpoints of the predicted output, based on Yolov3 [1], the B-Box prediction matrix for the proposed detection head is modified. It is divided into two parts: one for the front forward view, another one for the BEV, as shown in Figure 2. The prediction matrix contains 14 parameters (i.e., N = 14 in Figure 1d). The confidence score $p_0$ indicates the confidence that the predicted B-Box contains an object. If this predicted B-Box corresponds to the background, then this value should be 0. The classification scores $p_1, p_2, p_3$ represent the possibility that the category of the predicted B-Box falls into 'car', 'pedestrian' or 'cyclist' respectively. For the final output, only the B-Box with $p_0$ higher than a detection threshold will be kept, and the classification shows max $(p_1, p_2, p_3)$.

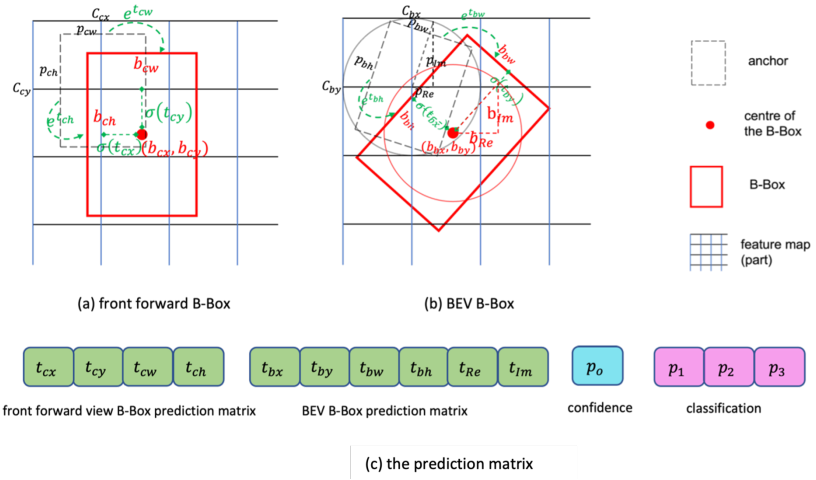


**Figure 2:** Converting the prediction matrix into B-Box: (a) front forward view B-Box, (b) BEV B-Box, (c) prediction matrix of the detector with 14 parameters.

Objects of primary interest in the context of autonomous driving, such as cars, pedestrians, and cyclists can generally be considered as standing or moving on the ground. Therefore, the front forward view bounding box surrounding a detected object on the RGB image can be represented by 4 parameters, that are the center coordinates, $t_{cx}, t_{cy}$, and the width and height, $t_{cw}, t_{ch}$, of the B-Box as shown in Figure 2a.

As shown in Figure 2b, different from the B-Box on RGB images, the BEV B-Box might not be parallel to the BEV map's coordinate axes. To predict the relative rotation of the B-Box, as inspired by complex YOLO [14], a Euler representation of the rotation angle is added to the prediction matrix of the proposed model. Hence, the BEV prediction matrix obtained from the regression of the proposed model contains 6 variables: Aside from the offsets of the B-Box centre coordinate $t_{bx}, t_{by}$, and the scaling of the width and height $t_{bw}, t_{bh}$, there is also the Euler representation of the rotation angle of the B-Box, $t_{Im}$ and $t_{Re}$.

## 4.2 Loss Functions

The loss function used in the proposed model consists of a combination of classification loss, B-Box regression loss, and confidence loss. Compared to YOLOv3 [1], the regression loss uses Generalized Intersection over Union (GIoU) [3] instead of mean square error (MSE). Moreover, to optimize the performance of the detector, the Euler angle is added to the B-Box regression loss. For the classification loss, focal loss [2] is added to address the imbalance problem in the training dataset.

1. Regression Loss

To match with the Euler angle regression network, a combination of GIoU [3] and Euler angle regression is used for the B-Box regression. The GIoU of the predicted B-Box and ground truth B-Box is computed as:

$$GIoU = \frac{I}{B^g \cup B^p} - \frac{A^c - (B^g \cup B^p)}{A^c} \quad (1)$$

Where $B^g$, $B^p$ are the ground truth B-Box and the predicted B-Box respectively. $I$ is the intersection of the ground truth and predicted B-Boxes, and $B^g \cup B^p$ is the union of the two B-Boxes. $A^c$ represents the smallest convex shape that encloses both $B^g$ and $B^p$. $A^c - (B^g \cup B^p)$ represents the area that is

inside $A^c$ but outside ($B^g \cup B^p$). The GIoU loss for the front forward view is represented by $L_{GIoU} = 1 - GIoU$.

Since the B-Box prediction matrix contains 4 variables for the front forward view and 6 variables for the BEV, the B-Box regression loss is divided into two parts: GIoU loss ($L_{GIoU}$) for the front forward view prediction matrix, and Euler GIoU ($L^E_{GIoU}$) for the BEV prediction matrix.

2. Classification Loss

In YOLOv3 [1], cross entropy loss [15] is used for classification. Ideally, a non-biased training dataset helps the model to learn the features for multi-class object detection and recognition, and cross entropy loss would be suitable. However, among the three classes considered in this research (car, pedestrian, cyclist), the training data provided by KITTI [4] contains 82% of the total objects in the class "car" and less than 5% of cyclist, which represents a bias. Inspired by [2], focal loss is used to replace cross entropy loss, which represents a first usage of focal loss on BEV maps to the best of our knowledge.

$$L^F_{cla} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{i,j}^{obj} \sum_{\mathrm{c}\in\mathrm{classes}} [\hat{p}_c(1-p_c)^{\gamma}\log(p_c) + (1-\hat{p}_c)\hat{p}_c{}^{\gamma}\log(1-p_c)] \quad (2)$$

$$I_{i,j}^{obj} = \begin{cases} 1, & \text{if object} \in \text{the } j^{\text{th}} \text{ anchor} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Where $\gamma$ is the relaxation parameter. The higher value of γ, the more "focus" will be given to misclassified examples, and the less loss will be propagated from examples. $S^2$ represents the number of grid cells, which is equal to the size of the feature map. In the proposed experiment, $S^2$ has three sizes: $19 \times 19$, $38 \times 38$, $76 \times 76$. $B$ represents the B-Box. $\mathrm{I}_{i,j}^{obj}$ is a binary value that indicates whether the $j^{th}$ B-Box of the $i^{th}$ grid cell's GIoU value is larger than the GIoU threshold. $p_c$ and $\hat{p}_c$ are the ground truth and the prediction classification score for class c.

3. Confidence Loss

The confidence loss is used to measure the objectiveness of the B-Box. The proposed model uses focal loss for confidence loss, as follows:

$$L_{con} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i(1-C_i)^{\gamma}\log(C_i) + \left(1-\hat{C}_i\right)\hat{C}_i^{\gamma}\log(1-C_i)\right] + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} \left[\hat{C}_i(1-C_i)^{\gamma}\log(C_i) + \left(1-\hat{C}_i\right)\hat{C}_i^{\gamma}\log(1-C_i)\right] \quad (4)$$

$$I_{i,j}^{obj} = \begin{cases} 1, & \text{if object} \in \text{the } j^{\text{th}} \text{ anchor} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

$$I_{i,j}^{noobj} = \begin{cases} 1, & \text{if the } j^{\text{th}} \text{ anchor is background} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$\hat{C}_i = \hat{p}_i(c) \times (GIoU + GIoU_E) \quad (7)$$

$$C_i = \begin{cases} 1, & \text{if object} \in \text{the } j^{\text{th}} \text{ anchor} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

If an object is detected in the B-Box, the confidence loss is $\sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i(1-C_i)^{\gamma}\log(C_i) + \left(1-\hat{C}_i\right)\hat{C}_i^{\gamma}\log(1-C_i)\right]$. $\hat{C}_i$ is the confidence score of the $j^{th}$ prediction B-box in $i^{th}$ grid cell, and $C_i$ is the ground truth, that is whether the B-Box contains an object.

In a real-life situation, most B-Box do not contain any object. This causes an imbalance problem where the background or negative samples are more frequently detected by the model than the objects or

some positive samples. To address this, the confidence loss is weighted down by a factor $\lambda_{noobj}$, which if no object is detected in the box (detected background only), the confidence loss will be $\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathrm{I}_{ij}^{noobj} \left[\hat{C}_i(1-C_i)^{\gamma} \log(C_i) + \left(1-\hat{C}_i\right)\hat{C}_i^{\gamma} \log(1-C_i)\right]$, where $I_{ij}^{noobj}$ is the complement of $I_{ij}^{obj}$ and $\lambda_{noobj}$ weighs the loss down.

In summary, the loss function of the proposed detector combines the two GIoU regression losses ($L_{GIoU}$ on the front forward view and the Euler angle loss $L_{GIoU}^{E}$ on the BEV view), the focal classification loss ($L_{cla}^{F}$), and the confidence loss ($L_{con}$). The combination of all components leads to the general loss function:

$$L = \alpha_1 L_{cla}^{F} + \alpha_2 L_{GIoU} + \alpha_3 L_{GIoU}^{E} + \alpha_4 L_{con} \tag{9}$$

Where $\alpha_1$, $\alpha_2$, $\alpha_3$, $\alpha_4$ are the weights for each part of the loss function, which are optimized based on experimental results.

# 5 Experimental Results

The performance of the proposed model is compared with other popular 3D object detectors that also use KITTI evaluation metrics with KITTI LiDAR data as input. The testing results show that the proposed model achieves a framerate of 46.4 frames per second (FPS) using a single NVIDIA Tesla V100 GPU while maintaining a high detection accuracy, computed as mean average precision (mAP).

**Table 1:** Comparative evaluation of existing 3D detectors and the proposed detector on KITTI dataset.

| | Method | Data | Speed (FPS) | mAP | | |
|---|---|---|---|---|---|---|
| | | | | Easy | Moderate | Hard |
| Two Stages | MV3D-Net [5] | LiDAR + RGB | 2.7 | 86.49 | 78.98 | 72.23 |
| | AVOD [6] | LiDAR + RGB | 10 | 89.74 | 84.81 | 78.12 |
| | F-PointNet [16] | LiDAR + RGB | 5.7 | 91.16 | 84.61 | 74.77 |
| | F-ConvNet [17] | LiDAR + RGB | 1.9 | 91.44 | 85.84 | 76.11 |
| | Fast Point R-CNN [18] | LiDAR | 15.3 | 90.87 | **87.71** | 80.51 |
| | MMF [19] | LiDAR + RGB | 12.2 | 86.81 | 76.75 | 68.41 |
| | STD [20] | LiDAR | 10 | 89.93 | 86.20 | 79.42 |
| Single Stage | VoxelNet [9] | LiDAR | 4.2 | 87.95 | 78.39 | 71.29 |
| | SECOND [10] | LiDAR | 19.7 | 89.33 | 82.87 | 78.51 |
| | PointPillars [8] | LiDAR | 41.9 | 90.07 | 86.56 | **82.81** |
| | SA-SSD [11] | LiDAR | 24.4 | 88.75 | 79.79 | 74.16 |
| | PIXOR [7] | LiDAR + RGB | 2.8 | 86.78 | 80.75 | 76.77 |
| | **Proposed detector** | **LiDAR + RGB** | **46.4** | **94.71** | 87.33 | 81.52 |

On cases annotated as “easy” according to the KITTI evaluation metrics, the proposed full detector performs best among the listed detectors in Table 1, leading the second best by over 3%. Compared with other detectors that use both the LiDAR point cloud and RGB images, such as MV3D-Net [5], AVOD [6], PIXOR [7], MMF [19], F-PointNet [16] and F-ConvNet [17], the proposed detector shows higher mAP on moderate and hard samples. As shown in Table 1, some models that use only a LiDAR point cloud as input reach better detection results on hard tasks compared with models that combine a

LiDAR point cloud and a RGB image. Sample experimental results are presented in the appendix (Figures 3 and 4) where predicted bounding boxes are displayed over the front forward RGB image (upper part) and the corresponding BEV map (lower part) for detected objects belonging to the three classes. Although combining with RGB images may weaken the model's ability to adapt to occlusions compared with using only a LiDAR point cloud as input, the proposed model demonstrates good detection results on testing scenes with occlusion.

# 6 Mini 3D Object Detector

With the continuous development of deep convolutional neural networks and the wide range of applications in this field, in addition to pursuing always higher accuracy, researchers also want to achieve higher detection speed. Ideally, a detector should be "compact", that is, the memory requirements and the amount of calculation involved in the detection must be limited as the availability of advanced hardware support is low for applications such as autonomous vehicles. These constraints motivate the development of deep convolutional neural networks that are better suited for widespread deployment on embedded devices. Therefore, although the detection speed of the full-scale detector proposed in Section 5 reaches up to 46.4 FPS, which is 15 times faster than PIXOR [7] and 3 times faster than MMF [19], we still want to explore the possibility of designing a lightweight network that uses fewer feature matrices to execute the same 3D object detection and recognition tasks.

## 6.1 Mini Detector Structure

The proposed mini detector merges the structure of tiny-YOLO [21] and the proposed full-scale detector to generate feature maps at 2 different scales. The mini detector's detection speed can reach up to over 3 times that of the proposed full-scale detector, therefore is more adaptable on mobile devices, while reaching a compromise on detection performance.

The main difference of the mini detector compared to its full-scale version is the backbone and FPN. The mini detector uses a DarkNet-19 [22] based backbone, modified to the input of the BEV map and corresponding RGB image. Compared to the 53-layers backbone of the full-scale detector, the mini detector's backbone only has 19 layers, that is about 1/3 the depth. For the mini detector only two different scales of feature maps are generated and passed to the detection head, compared to three in the full-scale version, to reduce the calculation. The detection head then converts the feature maps into prediction result. Otherwise, the detection head uses the same design as in the proposed full model and the same loss function, eq. (9), for training.

## 6.2 Experimental Results with the Mini Detector

For a fair comparison of performance with the two proposed detectors, the mini detector is trained and tested on the same software and hardware environment as the full detector. Moreover, the training and testing dataset remains the same. Table 2 presents the detection and recognition results of both the mini detector and the full detector on 1500 pairs of the LiDAR point cloud and the corresponding RGB images.

**Table 2:** Detection speed, precision, recall, AP and F1 estimated on each class and mAP on 1500 test samples with all three target classes for the mini detector compared with the full detector.

| | Class | Mini detector | Full detector |
|---|---|---|---|
| Speed (FPS) | | **158.97** | **46.4** |
| Precision | Car | 0.8922 | 0.9065 |
| | Pedestrian | 0.4783 | 0.6389 |
| | Cyclist | 0.6874 | 0.7951 |
| Recall | Car | 0.9572 | 0.9868 |
| | Pedestrian | 0.6231 | 0.9317 |
| | Cyclist | 0.8772 | 0.9524 |
| AP | Car | 0.9371 | 0.9794 |
| | Pedestrian | 0.6908 | 0.8272 |
| | Cyclist | 0.8544 | 0.9013 |
| F1 | Car | 0.9247 | 0.945 |
| | Pedestrian | 0.3838 | 0.758 |
| | Cyclist | 0.7832 | 0.8667 |
| mAP | | **0.8274** | **0.9026** |

As shown in Table 2, the mini detector achieves a good performance on detecting cars with AP higher than 0.9, but lower reliability on “pedestrian” and “cyclist” targets. This originates from the imbalance in the training dataset that contains no more than 20% of positive samples for the pedestrian and cyclist classes. Although FPN and focal loss [2] are used to reduce the impact of data imbalance, with fewer layers and less features extracted in the mini model, testing results suffer more from the imbalance than with the full detector. Conversely, the speed of the mini detector is 3.4 times faster than that of the full detector when tested in the same environment, while a 7.5% mAP reduction is observed.

# 7 Conclusion

Recent research shows an outstanding performance of convolutional neural networks in the field of computer vision and its potential for 3D image processing. Capitalizing on high-quality 3D point clouds collected by a LiDAR scanner and fusing the point clouds with corresponding RGB image contents, two deep learning architectures are proposed to further leverage the power of convolutional neural networks for 3D object recognition to assist in autonomous driving.

The input data of the proposed detectors contains a BEV map generated from the LiDAR point cloud along with information from a registered and rescaled RGB image that provides a front forward view. In the design of the proposed detector model, GIoU loss [3] and DarkNet-53 from a single stage detector YOLOv3 [1] are merged. The detection speed of the proposed full-scale model reaches up to 46.4 FPS in a single GPU-based implementation, with mAP over 90%.

To explore faster and lighter detection models better suited for real time and vehicle embedded implementations, a mini detector is also proposed by combining a compact 19-layer network model with key concepts of the proposed full-scale detector, which reaches mAP over 82%. Experiments demonstrate that compared to the full-scale detector, the mini detector takes about 2/3 of the training time and 1/3 of the testing time. A reasonable trade-off between processing time and accuracy can therefore be achieved for time-critical applications.

# Appendix: Experimental Results Visualization

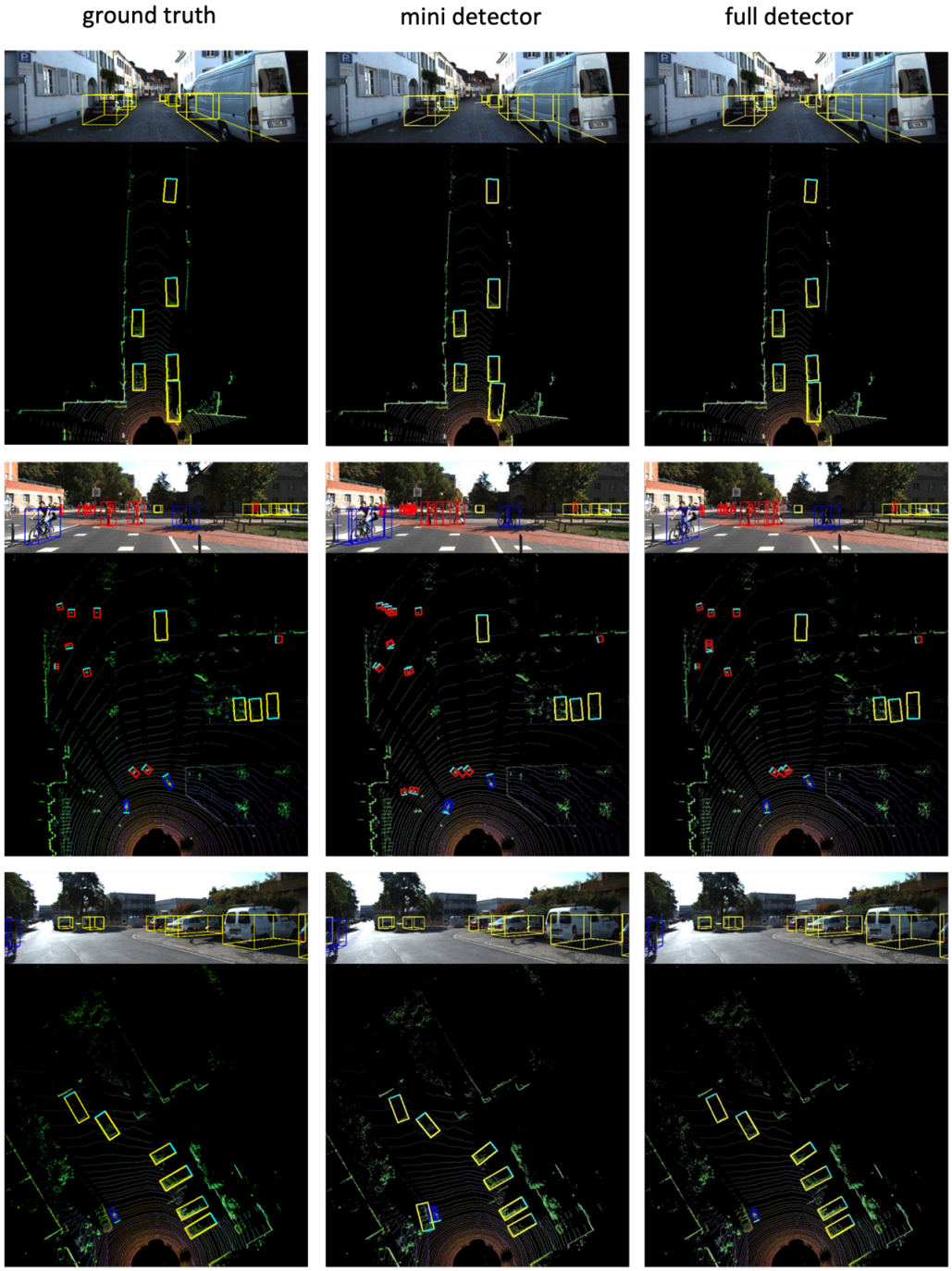


**Figure 3:** Samples cases comparing the ground truth (left), with detections from the mini detector (center), and from the full-scale detector (right), with B-Boxes (yellow = car; red = pedestrian; blue = cyclist) superimposed over front forward RGB image (upper part) and over corresponding BEV map (lower part).

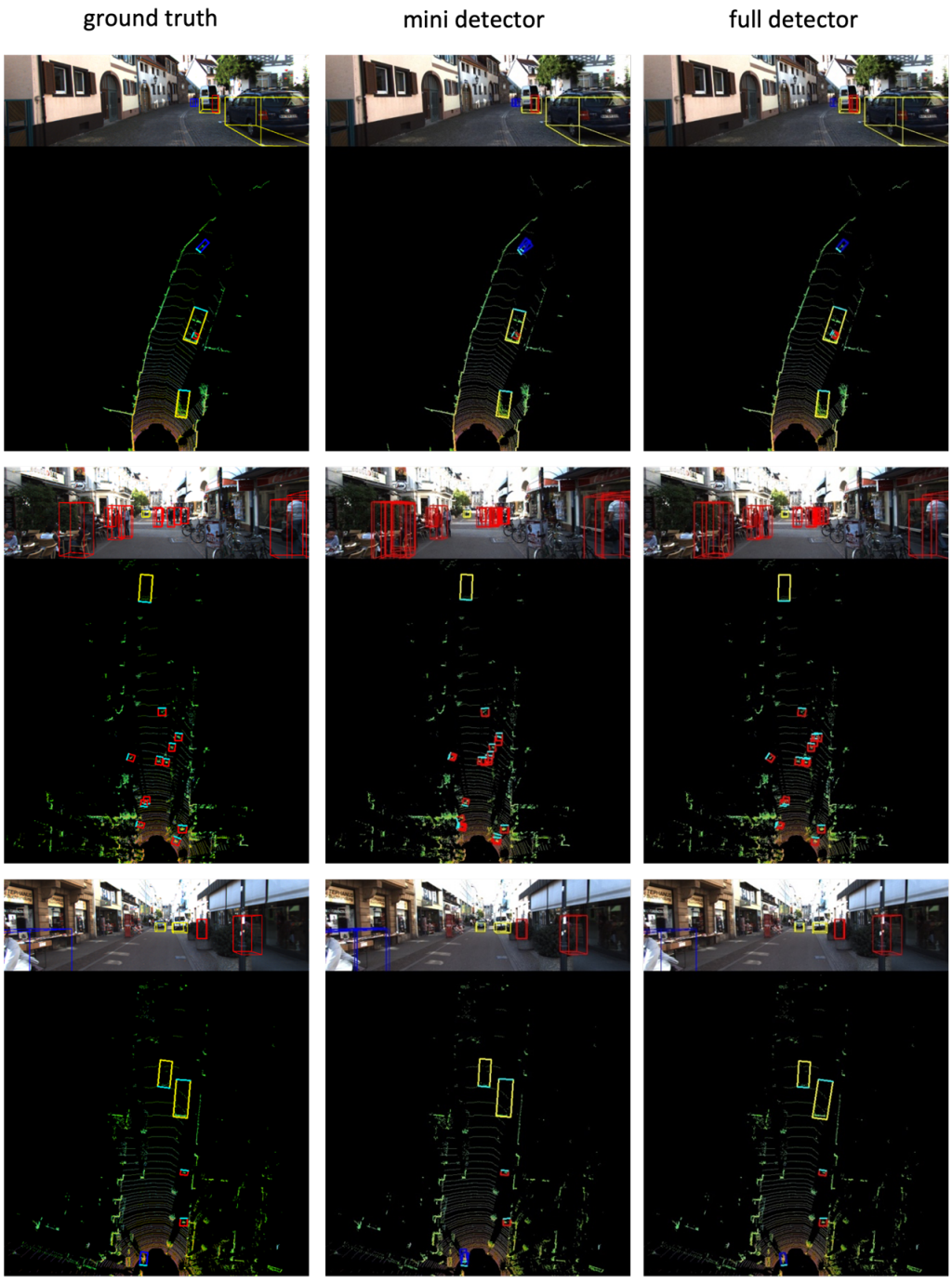


**Figure 4:** Additional samples cases comparing the ground truth (left), with detections from the mini detector (center), and from the full-scale detector (right), with B-Boxes (yellow = car; red = pedestrian; blue = cyclist) superimposed over front forward RGB image (upper part) and over corresponding BEV map (lower part).

# References


[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[2] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection," in Proceedings of the IEEE International Conference on Computer Vision, 2017.

[3] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[4] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[5] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3D object detection network for autonomous driving," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in Proceedings of 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018.

[7] B. Yang, W. Luo and R. Urtasun, "Pixor: Real-time 3D object detection from point clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[9] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[10] Y. Yan, Y. Mao and B. Li, "Second: Sparsely embedded convolutional detection," Sensors, 18(10), 3337, 2018.

[11] C. He, H. Zeng, J. Huang, X.-S. Hua and L. Zhang, "Structure aware single-stage 3D object detection from point cloud," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.

[12] V. A. Sindagi, Y. Zhou and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3D object detection," in Proceedings of the IEEE International Conference on Robotics and Automation, 2019.

[13] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia and A. De La Escalera, "Birdnet: a 3D object detection framework from lidar information," in 21st International Conference on Intelligent Transportation Systems, 2018.

[14] 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 Proceedings of the European Conference on Computer Vision Workshops, 2018.

[15] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in Advances in Neural Information Processing Systems, 2018.

[16] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018.

[17] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019.

[18] Y. Chen, S. Liu, X. Shen and J. Jia, "Fast point R-CNN," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019.

[19] M. Liang, B. Yang, Y. Chen, R. Hu and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[20] Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019.

[21] K. C. Saranya, A. Thangavelu, A. Chidambaram, S. Arumugam and S. Govindraj, "Cyclist detection using tiny YOLO v2," in Soft Computing for Problem Solving, 2020.

[22] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.