2023 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
Tomar, Portugal – April 26-27, 2023

Applying 3D Object Detection from Self-Driving Cars to Mobile Robots: A Survey and Experiments

Maciej K. Wozniak*
KTH Royal Institute of Technology
Stockholm, Sweden
maciejw@kth.se

Viktor Kårefjärd*
KTH Royal Institute of Technology
Stockholm, Sweden
karef@kth.se

Mattias Hansson*
KTH Royal Institute of Technology
Stockholm, Sweden
matthans@kth.se

Marko Thiel
Hamburg University of Technology
Hamburg, Germany
marko.thiel@tuhh.de

Patric Jensfelt
KTH Royal Institute of Technology
Stockholm, Sweden
patric@kth.se

***Abstract*—3D object detection is crucial for the safety and reliability of mobile robots. Mobile robots must understand dynamic environments to operate safely and successfully carry out their tasks. However, most of the open-source datasets and methods are built for autonomous driving. In this paper, we present a detailed review of available 3D object detection methods, focusing on the ones that could be easily adapted and used on mobile robots. We show that the methods do not perform well if used *off-the-shelf* on mobile robots or are too computationally expensive to run on mobile robotic platforms. Therefore, we propose a domain adaptation approach to use publicly available data to retrain the perception modules of mobile robots, resulting in higher performance. Finally, we run the tests on the real-world robot and provide data for testing our approach.**

***Index Terms*—perception, mobile robots, object detection**

I. Introduction

The area of object detection, vital for autonomous vehicles and mobile robotic applications, concerns locating and classifying different objects in a given frame [1]. For these applications, it is critical to understand both dynamic and static objects, such as pedestrians, cyclists, and cars [2]. In 2D computer vision, detected objects are usually represented by a bounding box (BB), a minimal box, typically axis-aligned that encloses the entire object [3]. Object detection in 3D is a more challenging problem. The added dimension means that the bounding box must include depth and full 3D coordinates. Moreover, the orientation of a bounding box, usually ignored in 2D, is more important when moving to three dimensions. Since most of the objects are not axis-aligned, failing to consider the rotation of the bounding box will result in a small Intersection over Union (IoU) between the predicted and ground-truth (GT) bounding box.

Several constraints must be considered when it comes to object detection in the area of mobile robotics. Due to safety, bandwidth, and privacy concerns, it may be essential to process data directly on mobile platforms such as drones, autonomous vehicles, and delivery robots [4]. Processing the data on the mobile platform means that the efficiency of detection algorithms is of great importance so that these methods can run on the limited computational resources available on the platform. Sensitive data, such as images of people and license plates, should be processed on the platform rather than stored on a remote server, thus preventing additional security risks.

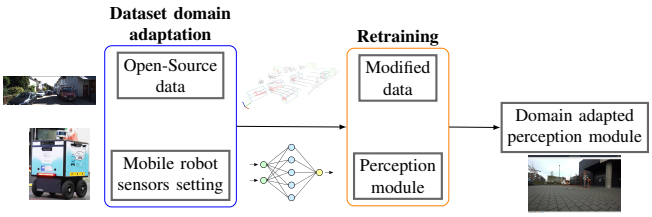


Fig. 1: Domain adaptation pipeline. Starting with modifying the open-source dataset to fit the robotic platform sensors' characteristics, retraining the models on that data, and finally testing it on a real-world robot.

Another important constraint is the cost and purpose of different mobile robots. Due to the high cost of high-resolution LiDARs, smaller mobile robots often use less expensive 16-channel LiDARs instead of more expensive 64 and 32-channel variants used in self-driving cars [5].

This paper surveys 3D object detection deep learning methods to highlight their strengths, weaknesses, and relevance to mobile robots. We tested chosen methods on two standard autonomous driving datasets to obtain relative performance and speed. Next, we build a domain adaptation pipeline, showed in Fig. 1, to adapt the methods to our mobile robotic platform Laura [6]. We modify the *KITTI* [7] dataset to simulate the 16-channel LiDAR available on our robot platform and used it to retrain the models in order to examine how a point clouds sparsity and range limit affects deep learning model performance in terms of accuracy and speed.

* Equal contribution

Our main contributions are (i) a detailed survey of available methods for 3D object detection from the autonomous driving area applied to mobile robots, (ii) creating a pipeline for adapting the autonomous driving datasets for mobile robots, (iii) retraining and testing the solutions on a real-world mobile robot platform, and (iv) providing data and code for testing our domain adaptation solutions[1].

II. Related Work

In recent years, deep learning methods have outperformed handmade feature descriptors, thus in this section, we will focus on them.

A. 2D Object Detection Using Deep Learning

Deep learning approaches use a deep neural network for feature extraction, removing the need for handcrafted feature detection [8]. There are two main deep learning approaches for 2D object detection.

The first group of methods generates a set of candidate boxes. Each candidate box is then classified after passing through a Convolutional Neural Network (CNN). Such approaches lead to high detection and positioning accuracy. They are referred to as regional-based methods. Examples include Region Based CNN (RCNN) [9], Fast-RCNN [10] or SPPNet [11].

The second group consists of one-stage frameworks. Methods in this group define the detection task as an end-to-end regression problem. A grid is used to represent an image. For a particular detection, the center of the target falls within a grid cell. That cell also handles the prediction of the object. By treating the problem this way, both position and classification are processed in a single stage. Single stage approach often achieves faster inference times than two-stage approaches. Examples include You Only Look Once (YOLO) [12], or Single Shot MultiBox Detector (SSD) [13].

Recently, transformers have been adapted from language to computer vision applications [14]. Early work started with Vision Transformer (ViT) [14] for object classification and Detection Transformer (DETR) [15] for object detection.

Transformers remove the need for hand-crafted modules such as a Region Proposal Network (RPN) [16]. DETR [15] uses a CNN backbone for feature extraction and positional encodings as an input to a transformer encoder-decoder. The encoder-decoder directly outputs bounding box proposals that benefit from the self-attention of transformers to differentiate between instances of detected objects. However, transformers often need large computational and memory resources; thus, they are currently not usable on most mobile robot hardware.

B. Rgb-based 3D Object Detection

While 2D object detection in images has matured a lot, 3D detection from 2D images remains a challenging problem. The task of monocular 3D detection is arduous because of the missing depth information [17].

[1]https://github.com/maxiuw/3DobjectDetectionAndTracking/.

There is a multitude of different methods used in this area. Some methods rely on monodepth estimation to obtain a depth map to perform 3D detection in a multi-–stage approach [18], while other more recent works focus on learning 3D bounding box representations in a single–stage manner [19], [20].

SMOKE [20] was one of the first end-to-end trainable single-stage detectors in this area. This method relies on a key-point estimator similar to [21], where the center of each object is represented by the 3D center of a detected object projected onto the image frame. The key-points are then regressed to obtain the full 3D bounding box, rather than using predefined anchors such as bounding boxes of the chosen size. Key-point classification and box regression is trained together in a single stage.

FCOS3D [19] is a state-of-the-art method within monocular 3D object detection. This method adapts the 2D detector FCOS [22] to obtain an anchor-free representation of 3D bounding boxes. FCOS3D [19] relies on feature extraction using 2D detectors, a Feature Pyramid Network (FPN) [23] for multi-scale representation, detection heads at multiple scales followed by anchor-free box regression. Similar to SMOKE [20], FCOS3D is trained in a single-stage approach but achieves higher accuracy, highlighting the potential of single-stage monocular 3D detection.

C. RGB-D based 3D object detection

RGB-D sensors combine RGB images with a corresponding depth map. The depth map can be combined with RGB images to aid 3D object detection [24]. There are different approaches to 3D object detection using RGB-D images [25].

2.5D approaches primarily utilize the depth image as a fourth channel for object detection using 2D CNNs [26]. A method that uses this approach is 3D-SSD [25], where features are extracted from the RGB and depth image. The features are then fused to predict 3D anchor boxes jointly using both domains.

3D approaches have the benefit of being able to learn 3D features, which may contain more information for the 3D object detection task, which focuses on detecting objects, their poses, and their classes. A benefit of using RGB-D is that the RGB image can be used to reduce the search space in the point cloud. One breakthrough method using this is Frustum PointNets [27]. Their method consists of extracting a 3D frustum from 2D object detection on the RGB image using a 2D CNN. After that, they perform 3D instance segmentation using PointNet [28] and 3D bounding box estimation on the points belonging to the correct segmentation. Since segmented points are used for detection, there is no need for voxelization as used in most LiDAR methods [29], [30]. Although Frustum PointNets is a promising approach, it struggles with semi-occluded objects that belong to the same 3D frustum since they overlap in the same segmentation.

D. LiDAR-based 3D Object Detection

LiDAR sensors generate data in three dimensions, represented as clouds of reflection points. Because the point cloud

tends to be relatively sparse, the applications do not naturally follow the fast-advancing literature of computer vision [31]. Traditional 2D convolution pipelines operate on feature-rich 2D images [32]. Therefore, much work in LiDAR-based detection focuses on how to represent this sparse, three dimensional data to suit convolution-based detection [33].

Another challenge in 3D detection is the speed of 3D convolution compared to their 2D counterpart [34]. The extra dimension adds a lot of computational complexity since the convolution kernel has to slide over a three dimensional grid.

Early work featured 3D convolution or variants of projection to reduce the data dimensionality. An approach that has recently become popular is to project the data into the ground plane to create a bird's eye view (BEV) [35], [36]. One of the BEV advantages is the nearly complete lack of occlusions. On the contrary, a perspective-based method like Frustum PointNets [27] can face difficulties when, e.g., two cars are to be positioned in the same bounding box. Since two cars stacked on top of one another vertically are not common, the BEV representation can be considered an advantage.

However, the BEV approaches [35], [36] still depend on fixed encoding to generate the BEV map, resulting in information loss along the vertical axis.

An improvement was proposed by authors of VoxelNet [29], the first application of PointNets [28] on LiDAR data for 3D object detection. VoxelNet introduced end-to-end learning in the area of 3D object detection where no features of interest are predefined, and the entire network is trained simultaneously through backpropagation. VoxelNet segments the environment into voxels with the use of PointNet [28]. Further, a 3D convolutions layer is used to adapt for the following 2D CNN detection. However, the inference speed of this type of method is too slow for a real-time implementation.

A refinement to the inference speed comes in the form of the Sparsely Embedded Convolutional Detection (SECOND) [37] as a successor of VoxelNet. The method changes how the CNN handles sparsely populated point clouds, resulting in an increased inference speed.

PointPillars [30] circumvents 3D convolution by using three main components: Pillar Feature Network, a 2D CNN, and a single shot detector (SSD) head detection. The end-to-end PointPillars method selects sub-regions in the point cloud data referred to as pillars (vertical columns) and creates a stacked pillars tensor with a corresponding pillar index tensor. PointNet [28] is used to learn a depiction of the point cloud in these pillars. This approach allows the encoder to learn a set of features. These features are then sent back to the *detection head* (CNN network) as a pseudo-image. It predicts the location of 3D bounding boxes for the detected objects.

E. 3D Object Detection Fusing RGB and LiDAR

There are multiple ways to improve object detection by combining images and point clouds. Some methods, such as Frustum PointNets [27] use the image to propose a region for 3D detection to perform box estimation. While Frustum PointNet uses both: point clouds and images, the features in the different domains are used separately. Consequently, multi-modal fusion methods emerged to better combine the features from RGB images and LiDAR point clouds.

MV3D [36] generates object proposals from a LiDAR BEV representation to create region proposals for 3D objects. The proposals are combined with feature detection in the BEV LiDAR projection, front view LiDAR, and RGB image through ROI (Region of Interest) pooling to fuse the features from the different views. The fused features are then jointly used to improve the performance of classification and bounding box regression.

Another method that combines RGB and LiDAR data is MVX-Net [38]. The method projects the point cloud or a voxelized point cloud onto the RGB image data using camera calibration information. The method then simultaneously uses the RGB image and the projected point features.

The approach used in MVX-Net [38] assigns more weight to detection made from LiDAR data, which can result in losing semantic information and features stored in the image. BEVFusion [39], on the other hand, unifies multi-modal features in a BEV representation instead. BEVFusion achieves good results in the semantic-oriented 3D scene segmentation in addition to 3D object detection. However, it is memory expensive to use and retrain.

F. Domain Adaptation

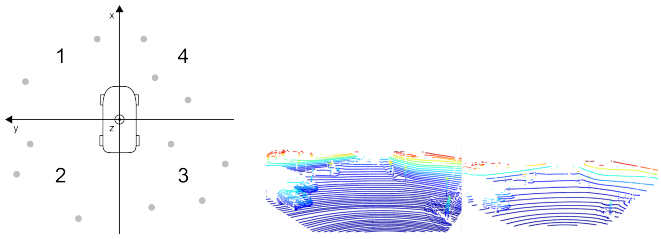
Domain adaptation is a sub-field of transfer learning that concerns performing the same task across different domains [40]. Domains are commonly referred to as source $\mathcal{D}^s$ and target domain $\mathcal{D}^t$ with the corresponding tasks $\mathcal{T}^s$ and $\mathcal{T}^t$. Domain adaptation considers the case where a model is trained on the source domain and is tested on the target domain to perform the same task $\mathcal{T}^s = \mathcal{T}^t$ under the assumption of domain differences $\mathcal{D}^s \neq \mathcal{D}^t$. There are two types of domain adaptation strategies: supervised [41] and unsupervised [42]. The unsupervised setting is of particular importance since it removes the need for labeled data in the target domain.

Wang et al. [43] showed the domain gap within 3D object detection between different autonomous driving datasets. The authors argue that the largest contributing factor was varying vehicle sizes across geographic areas. They proposed to improve generalization across datasets by a couple of simple techniques such as adjusting detection sizes according to vehicle size differences, statistical normalization, and fine-tuning detectors on a few samples from the target dataset.

Works by Saltori et al. [42] or Yang et. al [44] propose more sophisticated methods of domain adaptation for 3D detection by assigning pseudo-labels in target data and optimizing target performance in an unsupervised learning setting. Both of these methods rely on scaling as the main factor of learning domain invariance, along with scoring of pseudo-labels to determine good quality detections in the target domain.

III. METHODS

3D object detection methods and domain adaptation approaches presented in Section II are developed and tested for



(a) Illustration of a point cloud in a coordinate system relative to a capturing vehicle.

(b) The reduced KITTI point cloud (left), and the same point cloud with 16 LiDAR channels (right).

Fig. 2: Illustration of how *KITTI* dataset was adapted to our robot set-up.

autonomous driving scenarios. To bridge the gap between 3D object detection in mobile robots and self-driving cars, we propose a simple method for domain adaptation to the data and deep learning models retraining, so the models perform better (run faster and achieve higher precision) when deployed on mobile robots.

In Section IV we show that *off-the-shelf* solutions do not perform well on mobile robots. However, if the methods are retrained using data similar to the one obtained by the robot, we can achieve improved performance. Nevertheless, since each mobile robot is different, it would be extremely laborious to create and label enough data from each of them. We propose an approach that adjusts the open-source datasets, so that they resemble data obtained from our mobile robot, and use it to retrain the models. We hypothesize that this can improve performance in comparison to *off-the-shelf* models, trained on the original data.

For the experiments, we focused on *KITTI* since it uses a similar LiDAR sensor and it was collected in the same country as our robot data (Germany), allowing us to isolate the sensor differences by eliminating the geographic differences highlighted in [43].

The *KITTI* dataset does not include any information to which layer each data point in the point cloud belongs. To create our *Modified KITTI* dataset and remove 48 out of 64 layers, we exploited the order of the points in the point cloud data, see Fig. 2b. The points in a *KITTI* LiDAR scan are stored in the clockwise direction from the rotating LiDAR beam starting from the bottom layer and moving upwards. By going through the points in each scan one revolution has been completed where a shift in the sign of the y-axis coordinate occurs while the sign along the x-axis is positive, namely a shift between the first and fourth quadrant as seen in Fig. 2a.

In Algorithm 1, we showed how we store every fourth LiDAR layer to simulate the point cloud of a 16-channel LiDAR sensor.

For the final test, data sequences collected with our robot were converted into *KITTI* format, including transforming the labels from the LiDAR frame to the camera frame used by *KITTI*.

Algorithm 1 Layer removal

procedure LiDAR LAYER REMOVAL
  points ← coordinates of points in *point cloud*
  layer ← 0
  i ← 0
  previous quadrant ← quadrant of *points*[0]
loop:
  i ← *i* + 1
  quadrant ← quadrant of *points*[*i*]
  if *quadrant* > 1 **and** *previous quadrant* = 1 **then**
    layer ← *layer* + 1
  if *layer* MOD 4 = 0 **then**
    keep *point*
  previous quadrant ← *quadrant*

The data from the LiDAR used on our robot[2] is significantly sparser. At $30\,\mathrm{m}$ the distance between the layers will be around $1\,\mathrm{m}$, which is often too large to make any reasonable deductions. Additionally, the robot cannot drive faster than $6\,\mathrm{km/h}$, thus, it is more important to focus on detecting what is happening in its closest neighborhood. To address this, we investigated restricting the sensor range and the bounding boxes in the training and testing data to $50\,\mathrm{m}$ and $30\,\mathrm{m}$.

IV. Experiments

We tested how LiDAR resolution and max detection range affect various 3D object detection methods, tracking the methods' performance and speed. Selected models were retrained on the modified *KITTI* data with sparse LiDAR and different max detection ranges and tested on a subset of data collected by our mobile robot containing pedestrians, cars, and cyclists to test the models on unseen mobile robot data. Examples of the detections and GT labels can be seen in the supplementary video1.

All the LiDAR-based models were trained for 120 epochs, using Adam with decoupled weight decay (*AdamW*) optimizer, learning rate 0.003 with weight decay equal 0.001. In the case of MVX-NET, we decrease the learning rate to 1×10^{-7}, due to poor initial results.

To measure the performance of the chosen methods, we use average precision (AP) where $p(r)$ is a precision-recall curve and AP is the area under that curve; and mean average precision (mAP) where N is the number of classes, AP_i is the average precision for class i.

$$AP = \int_0^1 p(r)dr \qquad mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{1}$$

Reported results are from *Moderate* difficulty on *KITTI*[7]. We also report and compare runtime r of the methods in Hz.

We run the experiments and measure the speed of each model on a desktop computer with an Intel Core i7-6850K (6 cores, 12 threads) 4.00GHz CPU and an NVIDIA GeForce GTX 1080 (8 GB) GPU and our mobile robot, that uses Intel Core i7-7600U 2x2.80 GHz CPU and NVIDIA Volta GPU with 64 Tensor cores.

[2]Velodyne VLP 16, with 30° vertical angle

A. Off-the-shelf 3D Object Detection methods

Table I shows the tested methods and if they rely on RGB images, LiDAR point clouds, or both, and which dataset they were tested on.

We started by testing off-the-shelf models on *nuScenes* and *KITTI* datasets. In Table II, we compared FCOS3D [19], a mono camera-based method, and two LiDAR-based methods Centerpoint [30] and PointPillars [30] on the mini variant of the *nuScenes* dataset. Additionally, in Table III we compared methods on the *KITTI* dataset. We chose PointPillars [30] and SECOND [37] as LiDAR-based methods together with the mono camera-based SMOKE and the multi-modal fusion-based MVX-Net [38].

The results in Table II refer to mean Average Precision for pedestrians and cyclists, AP_P and AP_C respectively.

Since the *nuScenes* dataset features six cameras, the result in Table II is the combined task speed for all six of these inputs. A high number of cameras resulted in the per-camera frequency of $3.9Hz$ for the image-based FCOS3D [19].

In Table II, it can be observed how the LiDAR-based methods outperform the RGB-based method, FCOS3D [19], in all metrics and computational speed. The best performance was from CenterPoint [45], with the best detection scores and run-time. PointPillars [30] performed similarly in terms of AP for pedestrians and cyclists but was slower and with a lower mAP than CenterPoint.

Similar conclusions can be drawn from Table III where the tested RGB method, SMOKE [20], performs much worse than all other methods. Here one multi-modal method MVX-Net was also tested [38]. The method increases run-time because both image and LiDAR domains are used.

The tests in Table III had a faster run-time than in Table II. Even though *KITTI* has twice as many (64) LiDAR layers compared to *NuScenes* (32), the tests in *KITTI* only used the portion of the point-cloud in the forward direction of the car. Thus, a faster run-time is expected.

B. 3D Object Detection on Modified KITTI

Table IV shows the results of retrained methods evaluated and trained on the specified dataset configuration, reducing LiDAR layers and the LiDAR range. Example detections on modified *KITTI* can be observed in Fig. 3, where the retrained models produce fewer false positives in comparison to *off-the-shelf* ones, trained on the original *KITTI*.

The default models (without the asterisk) weights are pre-trained on unmodified *KITTI* and are tested on the modified *KITTI* with a specified range and reduced number LiDAR layers. Re-trained methods (with the asterisk) are trained and tested on the modified *KITTI* with a specified range limit and reduced number of LiDAR layers.

The results in Table IV indicate that the PointPillars [30] method does not improve when trained on the modified *KITTI* [7] dataset, whereas SECOND [37] and CenterPoint [45] improve significantly.

A decrease in performance in PointPillars can be caused by too few of points in the point cloud, resulting in a lower number of pillars being created, and therefore less accurate predictions.

SECOND performs the best when retrained on 16-layer and $30\,\mathrm{m}$ range limited LiDAR (mAP increase by 14%). Depending on the setting and the class, CenterPoint maP increases up to 30%. In MVX-Net, we observe the class and setting-specific increases (pedestrian and cyclists) or decreases (vehicles) in performance. These results may be the consequence of fusion methods being domain and sensor-specific.

In these experiments, we observed that while domain adaptation works well on most LiDAR-based models, the same approach does not directly transfer to improvement in fusion-based methods. We can also observe that models achieved the

TABLE I: Modality and dataset for tested methods.

	RGB	LiDAR	*nuScenes*	*KITTI*	*Our robot*
SMOKE	X			X	
FCOS3D	X		X		
CenterPoint		X	X	X	X
SECOND		X		X	X
PointPillars		X	X	X	X
MVX-NET	X	X		X	X

TABLE II: Results from testing, *nuScenes* mini dataset.

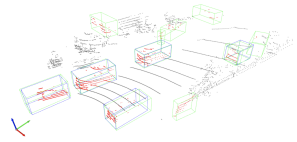
Method	mAP_{3D}	$AP_{3D,P}$	$AP_{3D,C}$	r [Hz]
Centerpoint	60.47	91.3	35.8	2.3
PointPillars	50.54	86.7	36.1	1.1
FCOS3D	38.00	47.8	23.9	0.65

TABLE III: Results from testing *KITTI*, Moderate.

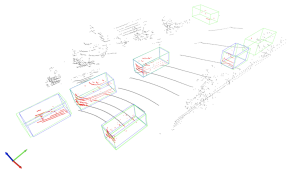
Method	mAP_{3D}	$AP_{3D,P}$	$AP_{3D,C}$	r [Hz]
SMOKE	3.50	3.58	2.49	17
PointPillars	64.4	51.0	62.8	23.6
SECOND	65.9	56.7	62.6	15.7
MVX-Net	62.9	55.7	54.6	7.8

TABLE IV: Results from testing 3D object detection (**V**ehicles, **P**edestrians, and **C**yclists) with range limits in 16-layers LiDAR. The best scores are highlighted in bold. *Retrained using modified *KITTI* dataset.

Method	Range limit [m]	mAP_{3D}	$AP_{3D,V}$	$AP_{3D,P}$	$AP_{3D,C}$
PointPillars	No limit ($\approx$ 70)	47.97	65.47	37.26	41.16
	50	48.95	65.61	38.83	42.42
	30	61.54	84.43	41.26	58.95
PointPillars*	No limit ($\approx$ 70)	48.49	67.49	35.37	42.74
	50	47.76	67.39	34.29	41.61
	30	57.56	83.75	37.60	51.11
CenterPoint	No limit ($\approx$ 70)	35.52	49.34	21.01	36.21
	50	36.06	49.57	21.24	37.39
	30	48.91	72.53	22.14	52.06
CenterPoint*	No limit ($\approx$ 70)	46.91	54.22	41.41	45.1
	50	51.15	60.94	46.40	46.09
	30	63.68	79.82	45.66	**65.57**
SECOND	No limit ($\approx$ 70)	42.78	54.32	38.47	35.54
	50	45.35	54.61	39.10	36.35
	30	57.82	78.03	42.15	53.27
SECOND*	No limit ($\approx$ 70)	50.50	65.75	43.93	41.80
	50	51.94	65.70	45.93	44.18
	30	**64.89**	**84.66**	50.31	59.69
MVX-Net	No limit ($\approx$ 70)	43.48	59.78	38.47	27.10
	50	44.11	60.89	43.77	27.69
	30	56.64	81.40	47.70	40.83
MVX-Net*	No limit ($\approx$ 70)	53.73	68.02	53.09	40.06
	50	54.30	67.22	**53.43**	42.25
	30	56.34	79.00	44.15	45.86

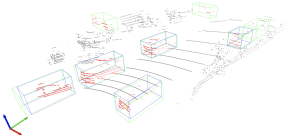


(a) PointPillars trained using standard *KITTI*.



(b) SECOND trained using *modified KITTI*.

(c) MVX-NET trained using standard *KITTI*, detections in image frame.



(d) MVX-NET trained using standard *KITTI*.

Fig. 3: Example detections of the best scoring models achieved on *modified KITTI*.

TABLE V: Results from testing on labeled robot's data AP for **V**ehicles, **P**edestrian and **C**yclists. The configurations indicate how the model was trained. The default is 64-layer LiDAR data with no range limit.

Method	Train. Config	$AP_{3D,V}$	$AP_{3D,P}$	$AP_{3D,C}$	r [Hz]
PointPillars	Default	65.38	46.43	27.33	30.5
	16 lay.	48.06	50.59	38.40	31.6
	16 lay. 50m	45.49	46.76	21.09	31.9
	16 lay. 30m	5.20	10.39	4.71	32.4
CenterPoint	Default	34.20	45.17	31.85	15.4
	16 lay.	53.99	66.16	67.47	15.3
	16 lay. 50m	68.25	70.16	67.73	15.5
	16 lay. 30m	48.93	66.26	49.00	15.7
SECOND	Default	53.81	44.74	66.54	21.5
	16 lay.	69.43	81.89	81.09	22.7
	16 lay. 50m	**72.83**	**86.19**	**95.57**	22.8
	16 lay. 30m	44.07	77.94	81.05	23.0
MVX-Net	Default	8.93	39.36	23.91	13.1
	16 lay.	8.55	29.12	29.41	13.0
	16 lay. 50m	3.25	43.41	21.11	13.2
	16 lay. 30m	1.25	32.51	28.13	13.5

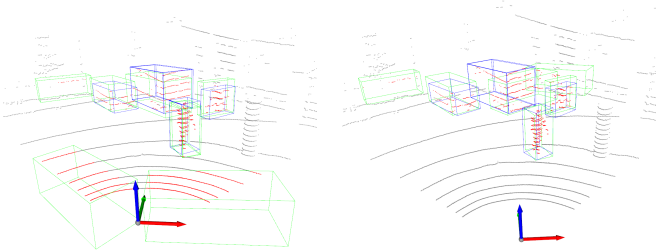
highest performance on the *vehicle* class. One of the causes is *label imbalance*, commonly pointed out in *KITTI*, meaning that the majority of the labeled points belong to the *vehicle* class, but vehicles are also bigger, whereas cyclists and humans are thinner and easier to miss. That results in much higher $AP_{3D,V}$ than AP_{3D} in other classes.

C. 3D Object Detection on Our Robot's Data

In Table V we can see that layers reduction and range limit, correspond to speed increase in each method. The best overall detection (for all three classes) was achieved by SECOND by retraining with $50\,\mathrm{m}$ range 16 LiDAR channels, followed by SECOND retrained using 16 channels without a range limit (Fig. 4).

We can see significant improvement in all three classes when testing retrained CenterPoint [45], however, the scores are not as high as in SECOND.

For PointPillars [30], we can observe improvement in pedestrian and cyclist detection when using the model retrained with 16-layer LiDAR without a range limit. However, similarly to



(a) Off-the-shelf SECOND (b) Domain-adapted SECOND

Fig. 4: Reduced number of false positives from domain-adapted SECOND model. Ground-truth bounding boxes are highlighted in blue, and detections are green in the point cloud.

the tests run on *KITTI* [7], with limited range (such as $30\,\mathrm{m}$), we can see a significant decrease in performance.

MVX-Net [38] achieved quite poor results. We speculate that this can be caused by wrong sensor calibration and the rigidity of fusion methods. It can be potentially improved by creating and training on a larger robot-specific dataset; however, that would again make the model specific to the robotic platform.

V. Discussion and Limitations

In this paper, we made a step towards bridging the gap between 3D object detection in self–driving cars and mobile robots.

We observed how the reduction of LiDAR data to 16-layers as compared to 64-layers resulted in an information loss that negatively affected performance. The examined multi-modal fusion method showed similar performance drops in the lack of high-resolution LiDAR data, even with the addition of a second modality. In most cases, retraining the methods using the lower resolution point cloud improved the performance; however, decreasing the distance resulted in an mAP drop for PointPillars (LiDAR-based) and MVX-Net (fusion). This highlights the adaptability of different methods toward sparse point clouds.

Finally, we showed a significant performance decrease on the target robotic platform unless the model is retrained on the sparse LiDAR dataset. Our robot uses a LiDAR with different azimuth angles and is collected at a different angle than the *KITTI* training set. These differences increase the generalization difficulty when transferring models to this new domain. We have seen that LiDAR-based methods generalized much better well toward unseen robot data than multimodal methods. We suspect that the models could achieve yet higher performance using more advanced domain adaptation or by retraining on a large robot-specific dataset. Additionally, we showed that with sparser LiDAR and limited distance, we can increase the speed of the methods (Table II vs. Table III with Table V).

References

[1] P. Jing, W. Zheng, and Q. Xu, "Vision-based mobile robot's environment outdoor perception," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019, pp. 1–5.

[2] B. Jung and G. S. Sukhatme, "Detecting moving objects using a single camera on a mobile robot in an outdoor environment," in *International conference on intelligent autonomous systems*, Citeseer, 2004, pp. 980–987.

[3] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[4] M. Thiel, S. Tjaden, M. Schrick, K. Rosenberger, and M. Grote, "Requirements for robots in combined passenger/freight transport," in *Hamburg International Conference of Logistics (HICL) 2021*, epubli, 2021, pp. 195–215.

[5] "Rsn: Range sparse net for efficient, accurate lidar 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5725–5734.

[6] M. Thiel, M. Grote, M. Schrick, and S. Tjaden, "Integrierte mobilität und automatisierung-transportroboter unterwegs im tabula shuttle/integrated mobility and automation-transport robots on the road in the tabula shuttle," *Automobiltechnische Zeitschrift*, vol. 124, no. 4, 2022.

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, 2013.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 1440–1448.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[13] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, Springer, 2016, pp. 21–37.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 213–229.

[16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[17] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *CVPRUNTIL2018*, 2016, pp. 2147–2156.

[18] B. Xu and Z. Chen, "Multi-level fusion based 3d object detection from monocular images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2345–2353.

[19] T. Wang, X. Zhu, J. Pang, and D. Lin, "Fcos3d: Fully convolutional one-stage monocular 3d object detection," in *Proceedings of the International Conference on Computer Vision*, 2021, pp. 913–922.

[20] Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 996–997.

[21] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[22] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the International Conference on Computer Vision*, 2019, pp. 9627–9636.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[24] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and dense lidar data," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 4112–4117.

[25] Q. Luo, H. Ma, L. Tang, Y. Wang, and R. Xiong, "3d-ssd: Learning hierarchical features from rgb-d images for amodal 3d object detection," *Neurocomputing*, vol. 378, pp. 364–374, 2020.

[26] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 821–826.

[27] 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/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[29] 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, pp. 4490–4499.

[30] 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, pp. 12 697–12 705.

[31] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[33] A. Khoche, M. K. Wozniak, D. Duberg, and P. Jensfelt, "Semantic 3d grid maps for autonomous driving," in *IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2022, pp. 2681–2688.

[34] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[35] 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 *IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018, pp. 3517–3523.

[36] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

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

[38] 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*, IEEE, 2019, pp. 7276–7282.

[39] Z. Liu, H. Tang, A. Amini, *et al.*, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," *arXiv preprint arXiv:2205.13542*, 2022.

[40] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[41] K. Lai and D. Fox, "Object recognition in 3d point clouds using web data and domain adaptation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1019–1037, 2010.

[42] C. Saltori, S. Lathuiliére, N. Sebe, E. Ricci, and F. Galasso, "Sf-uda 3d: Source-free unsupervised domain adaptation for lidar-based 3d object detection," in *Proceedings of the International Conference on 3D Vision (3DV)*, IEEE, 2020, pp. 771–780.

[43] Y. Wang, X. Chen, Y. You, *et al.*, "Train in germany, test in the usa: Making 3d object detectors generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 713–11 723.

[44] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 368–10 378.

[45] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 784–11 793.