3D Object Detection
CS230 Project Report

Suvasis Mukherjee
Stanford University
(suvasism@stanford.edu)

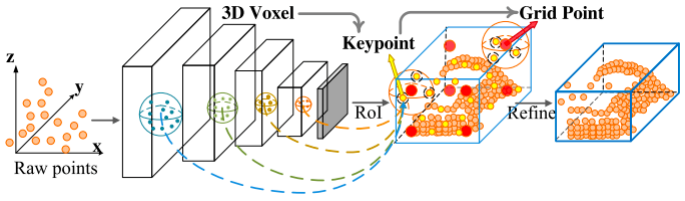
I. *Abstract*

3D object detection systems have become a core component in most of the recent autonomous vehicles systems. Autonomous vehicles are required to precisely detect and track the surrounding objects in real-time to achieve safe driving. Among the various sensors used in recent autonomous vehicles system, LiDAR, which measures distance between objects and the ego-vehicle has become a crucial sensor for 3D object detection systems. PV-RCNN [1] is a high-performing pointcloud based 3D object detector. However, PV-RCNN's Two-stage detectors cannot adequately identify changing object scales, differing point-cloud density and clutter issues. I propose to introduce some changes in the module based on 2D deformable convolution networks that can adaptively gather instance-specific features from locations where informative content exists. The results are based on KITTI dataset [2].

II. *Introduction*

3D Object Detection for self-driving cars is very challenging due to the enormous amount of data involved and real-time nature of the problem where the bar for the accuracy and performance of the model is very high. LiDAR is an important sensor in 3D detection system. It captures 3D scene information as sparse and irregular pointclouds, which provide vital cues for 3D scene perception and understanding. State of the art detector, PV-RCNN [1] understands 3D features from irregular cloud points. PV-RCNN achieves high performance 3D object detection by integrating point-voxel networks to learn 3D features from irregular pointcloud. There are 2 categories of 3D object detection methods based of point cloud representation, grid based (voxel based features) and point based (point based features) methods. Grid based method transforms irregular pointclouds to regular representation like birdview maps which can be processed by 2D or 3D CNN. PointNet [3] extracts discriminative features from raw point cloud for 3D detection. The disadvantage of grid based method is it lacks fine grained localization accuracy. This deficiency is addressed by point based method which preserves accurate location information. PV-RCNN combines both these feature learning schemes in 2 steps, voxel to keypoint scene encoding and keypoint to grid ROI feature abstraction. To understand keypoint, voxel-wise feature learning scheme requires a large number of voxels to generate accurate proposal, this is computationally inefficient. Hence, Furthest Point sampling (FPS) algorithm is used to select a small set of keypoints. The features of these keypoints are aggregated by grouping neighboring voxel-wise features via PointNet [3] set abstraction for summarizing multi-scale point cloud information. Voxel-to-keypoint is a good way to encode the whole scene.

Fig. 1. PV-RCNN framework.



Point-to-grid ROI feature abstraction aggregates the scene keypoint features to ROI grids for proposal confidence prediction and location refinement.
PV-RCNNs is efficient because it randomly sampled keypoints which capture multi-scale features for proposal refinement while retaining fine-grained localization information. However, in some situations random sampling is not effective in case of pedestrians or traffic poles as they are very hard to distinguish in point clouds. In this case, it would be better to align the keypoints towards the most discriminative areas, so that principal features for a pedestrian can be highlighted. Similarly, the scales for cars, pedestrians and cyclists are very different. While multi-scale feature aggregation is advantageous for image features, the non-uniform density of point clouds makes it hard to detect them with a single model. The goal of the project is to aggregates and focus on the most salient features at different scales and pick up on unevenly distributed contextual information.

III. *Literature Review*

Prior to emergence of modern Convolutional Neural Networks (CNNs) architectures and large-scale image datasets such as ImageNet [5], object detection was traditionally done with hand-crafted features, such as HoG [3] or Haar [9]. Object detection through CNNs was first done by using external segmented objects [6], and later by using the Region Proposals Networks (RPNs) [11].

For 3D Object Detection, there have been three approaches. The first approach as in [7] and [8] (among many others) is

to estimate the spatial location of the objects based solely on monocular visual information A second approach has been to use both camera and LiDAR as complementary data sources and combine the information as in [16]. In this data fusion approach, RPN was used in [17] where regions of interest and classification are computed on the image space, and final location is performed over the LiDAR data.

The third approach to 3D Object Detection uses point cloud data to compute object detections in 3D, either using information from stereo cameras or LiDARs. Earlier examples of this approach are [9] and [10]. A more recent approach for processing the point cloud spatial data has been to convert the 3D space into a voxel grid and apply 3D convolutions [11] [12] [13]. More recently, a more novel approach in the point cloud category is to use 2D CNNs on LiDAR point cloud of the front view [14] or a bird's eye view (BEV) [15].

The Bird's Eye View (BEV) projection of the LiDAR data can be processed by either single- [29], [31] or two-stage [21], [32] image detectors. Even processing of more compressed (binary) representations of BEV projections have been attempted by MODet [33].

The compactness of the BEV data enables the above methods to perform high speed processing for real-time applications such as autonomous driving. However, due to the sparsity of data and simplicity of the process, there is some information loss, especially the height coordinate.

The Birdnet+ [34] alleviates these limitations by using a two-stage object detector and ad-doc regression branches. Thus eliminating a post processing stage. Birdnet+ provides a mechanism for 3D box regression using only BEV images as input. We based my project on this paper and to improve the performance of the model. Since BirdsNet+ model lacks precision of the bounding box as it combines the localization and box regression together and classifies, we also studied how to improve a 3D object detection performance by predicting more precise 3D bounding box coordinates [36]. The objects (car, cycle, pedestrian etc.), i.e, box regression module and non-objects (localization) are classified independently. This change in the object detection pipeline helps classify box regression more precisely and improves the quality of the boxes. It applies spatial transformation to each point cloud points to improve precision.

3D Object Detection on LiDAR data is a rich growing area of research, and new developments are happening rapidly, One very interesting model is developed for 3D Video Object Detection using graph-based message passing and spatiotemporal transformation attention [35]. In this project, an end-to-end online 3D video object detector is built that operates on point cloud sequences. An Attentive Spatiotemporal Transformer GRU (AST-GRU) aggregates the spatiotemporal information to emphasize the foreground objects, and Temporal Transformer Attention (TTA) module aligns the dynamic objects. These attention-aware spatiotemporal aggregation mechanisms capture the video coherence in consecutive point cloud frame, which yields an end-to-end online solution for the LiDAR-based 3D video object detection.

IV. *Data Set and Features*

I performed experiments and benchmarking on KITTI Dataset[2].The KITTI dataset has been recorded from a moving platform while driving in and around Karlsruhe, Germany . It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system There are 4 different types of files in the dataset: camera_2 image (.png), camera_2 label (.txt), calibration (.txt), and velodyne point cloud (.bin).

For each frame, there is one of these files with the same name but different information. The image files are regular .png files. The label files contain the bounding box information for objects in 2D and 3D in text. Each row of the file is one object and contains 15 values (for classes like car, pedestrian etc.). The 2D bounding boxes are in terms of pixels in the camera image . The 3D bounding boxes are in 2 coordinates. The size (height, weight, and length) are in the object coordinate, and the center of the bounding box is in the camera coordinate. The point cloud file represents the location of the point and its reflectance in the lidar coordinate.

PV-RCNN uses images as input to the model. I acquired the publicly available KITTI object detection data set [2]. It consists of 7481 training images and 7518 test images, comprising a total of 7518 labeled objects. The training samples are generally divided into the train split (3,712 samples) and the val split (3,769 samples).

PV-RCNN experiments focused on three categories – car, pedestrian, and cyclist.

.

V. *Methods*

Technical Approach

A. Overall PV-RCNN architecture

Unlike RGB images, LiDAR point clouds are 3D, unstructured and include the coordinates in meters and the intensity value from the reflected beam, these are the features of the input points.

PV-RCNN uses keypoints to integrate two point cloud feature learning strategies namely, point-cloud set abstraction and voxel-based sparse convolution. It also bridges the RPN and RCNN stage. It uses 3D Voxel CNN for feature encoding and proposal generation. All the input points in the pointcloud are first converted into small voxels where the features of the non-empty voxels are directly calculated as the mean of point-wise features of all inside points. The network utilizes a series of $3 \times 3 \times 3$ 3D sparse convolution to gradually convert the point clouds into feature volumes with 1×,2×, 4×, 8× downsampled sizes. Such sparse feature volumes at each

level could be viewed as a set of voxel-wise feature vectors. 3D proposal generation: the encoded 8× downsampled 3D feature volumes are converted into 2D bird-view high-quality 3D proposals are generated following the anchor-based approaches. PV-RCNN stack the 3D feature volume along the Z axis to obtain the L 8 × W 8 bird-view feature maps. Each class has 2× L 8 × W 8 3D anchor boxes which adopt the average 3D object sizes of this class, and two anchors of

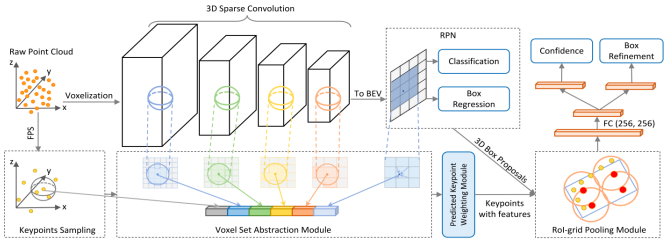
$$0^0, 90^0$$

orientations are evaluated for each pixel of the bird-view feature maps.

PV-RCNN uses a two-stage approach to combine the two algorithms described above. voxel-to-keypoint scene encoding step proposes proposals. voxel method extracts feature for the entire scene using point FPS(further point sampling) and retrieves features of multi-scale voxel. This actually only uses the characteristics of voxel, but it is expressed in key-point. The second stage is'keypoint-to-grid RoI feature abstraction': In this step, RoI-grid pooling module combines the characteristics of keypoints and RoI-grid points of the previous step Fusion. Then, the aggregated features of all RoI-grid points are used together for subsequent proposal optimization.

The figure (fig.2) below shows the PC-RCNN overall architecture- The raw pointclouds are first voxelized to feed into the 3D sparse convolution based encoder to learn multi-scale semantic features and generate 3D object proposals. Then the learned voxel-wise feature volumes at multiple neural layers are summarized into a smallset of keypoints via the novel voxel set abstraction module. Finally the keypoint features are aggregated to the RoI-grid points to learn proposal specific features for fine-grained proposal refinement and confidence prediction.

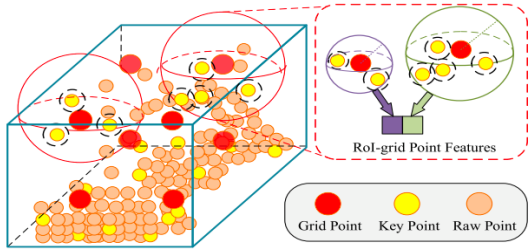
Fig. 2. PV-RCNN over all architecture.



3D sparse convolution processing of voxel has obtained relatively accurate proposals, but there are many The characteristics of the scaled keypoint needs refinement. PV-RCNN proposes keypoint-to-grid RoI feature abstraction module at this stage. as follows:

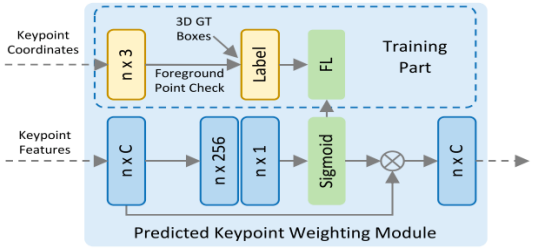
To group keypoint features for proposal refinement, a RoI-grid pooling via set abstraction is employed. It aggregates the

Fig. 3.



keypoint features to the RoI-grid points. As shown in the figure (fig 3), the red dot represents the grid point, the light yellow represents the KeyPoint, and the dark yellow represents the original point.

Using the grid points as the center, set a variable radius to aggregate the features of nearby KeyPoints, regularize the features of KeyPoints, and turn them into Voxel-type features again. There are two advantages to doing this: 1. When extracting Voxel features, you can collect the information of the object boundary outside the frame, which is good for the refinement of the recommended frame. 2. Greatly reduce 0 value features and reduce the sparsity of feature space. PV-RCNN samples 6×6×6 grid points in each proposal. First determine the neighbors within a radius of each grid-point, and then use a pointnet module to integrate the features For the characteristics of grid point, feature fusion means of multiple scales will be used here. After getting all the grid-point point features, PV-RCNN uses two layers of MLP to get the characteristics of 256-dimensional proposals.



B.Proposed Architectural modification

Technique applied in Mesh-RCNN [4] can be used to deform keypoints in PV-RCNN. Adaptive deformation module adaptively aligns keypoints towards the most content rich and discriminative features. The n sampled keypoints (shown in yellow in Fig. 2) have a 3D position v_i and a feature vector f_i corresponding to any 4 Conv layers as shown in fig2. This module will compute updated features $f_i^{'}$

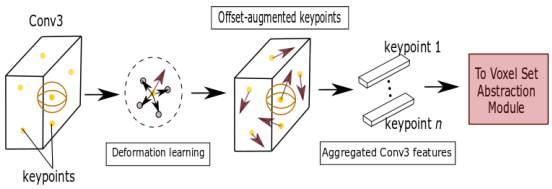
$$f_i^{'} = (1/n)RELU(\sum_{j \in N(i)} W_{offset}(f_i - f_j).(v_i - v_j))$$

where N_i gives the i-th keypoint's neighbors in the point-cloud and W_{offset} is a learned weight matrix. The new deformed keypoint positions as

$$v_i^{'} = v_i + tanh(W_{align}[f_i^{'}])$$

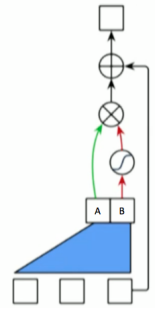
where W_{align} is a learned weight matrix [4].

Fig. 4. deformation module.



To compute the features for the deformed keypoints using PointNet++ similar to the PV-RCNN pipeline. Ref fig.2, the sample keypoints passed through gated CNN [40] (not shown in the fig. 2) to voxel set abstraction module. Gating block,

Fig. 5. Gated Linear Unit with residual connection.



Dauphin et al. [40], uses a mechanism called a "gated linear unit" (GLU), which involves element-wise multiplying A by sigmoid(B),

$$A \otimes sigmoid(B)$$

. Here, B contains the 'gates' that control what information from A is passed up to the next layer in the hierarchy. Conceptually, the gating mechanism allows selection of the keypoint features that are uncluttered and only refined keypoints are passed to voxel set abstraction module, and provides a mechanism to learn and pass along just the relevant keypoint features. Similar to ReLU, the gating mechanism also provides the layer with non-linear capabilities, while providing a linear path for the gradient during backpropagation (thereby diminishing the vanishing gradient problem). Using this concept, Given a keypoint feature f_i, the gated feature is obtained as function g = sigmoid($W_{gate}f_i + b_{gate}$) and modified feature of a keypoint is f_i^g = g $\otimes$W$_{f_c}f_i$

Where the W_{f_c}, W_{gate} and b_{gate} are learned from data. This function is applied to all the sample keypoint features and the gated features are passed on to the voxel set abstraction module.

C. Loss function

3D Proposal Refinement and Confidence Prediction The author uses the 3D Intersection-over-Union (IoU) proposed by the predecessors in the confidence prediction branche. The confidence goal for the k_{th} ROI is the following formula:

$$y_k = min(1, max(0, 2IoU_k - 0.5))$$

where IoU_k is the IoU of the k_{th} RoI w.r.t. its ground-truth box.

This formula represents the GT corresponding to the k_{th} ROI, so the LOSS function of the confidence prediction branch uses cross-entropy loss:

$$L_{iou} = -y_k log(y_k^{hat}) - (1 - y_k)(1 - log(y_k^{hat}))$$

where y_k^{hat} is the predicted score by the network.

Training loss RPN loss

$$L_{rpn} = L_{cls} + \beta(\sum_{r \in x,y,z,l,h,w,\theta} L_{smooth}(\Delta_{r^a}^{hat}, \Delta_{r^a}))$$

PV-RCNN framework is trained end-to end with the region proposal loss L_{rpn}, keypoint segmentation loss L_{seg} and the proposal refinement loss L_{rcnn}.

where the anchor classification loss L_{cls} is calculated with focal loss with default hyper-parameters and smooth L1 loss is utilized for anchor box regression with the predicted

Keypoint seg loss is the weight loss of the key points in the front background refinement loss is defined as follows:

$$L_{rann} = L_{iou} + \sum_{r \in x,y,z,l,h,w,\theta} L_{smooth}(\Delta_{r^a}^{hat}, \Delta_{r^a})$$

The overall training loss are then the sum of these three losses with equal loss weights. Further training loss details are provided in the supplementary file.

VI. *Results from Experiments*

The 3D object detection benchmark of KITTI [3] contains 7481 training samples. Following [1], we divide them into 3712 training samples and 3769 validation samples. Report results are on the validation (val) split of KITTI.

Network Architecture. As shown in Fig. 2, the 3D voxel CNN has four levels with feature dimensions 16,32,64,64, respectively. Their two neighboring radii rk of each level in the VSA module are set as (0.4m,0.8m), (0.8m,1.2m), (1.2m,2.4m), (2.4m,4.8m), and the neighborhood radii of set abstraction for raw points are (0.4m,0.8m). For the proposed RoI-grid pooling operation, we uniformly sample 6×6×6 grid points in each 3D proposal and the two neighboring radii ˜ r of each grid point are (0.8m,1.6m). For the KITTI dataset, the

detection range is within [0,70.4]m for the X axis, [40,40]m for the Y axis and [3,1]m for the Z axis, which is voxelized with the voxel size (0.05m,0.05m,0.1m) in each axis.

Experiment Summary

Training and Inference Details: PV-RCNN framework is trained with the ADAM optimizer. Trained the entire network with the batch size 4, learning rate 0.01 for 15 epochs on 1 AWS Tesla GPUs, which takes around 20 hours. Batch size more than 4 run into out of memory issue. 15 epoch took ab 20hrs.

The following table summarizes experimentation. moderate level of KITTI val split with AP calculated calculated by 40 recall positions. For the proposal refinement stage, PV-RCNN randomly sample 128 proposals with 1:1 ratio for positive and negative proposals, where a proposal is considered as a positive proposal for box refinement branch if it has at least 0.55 3D IoU with the ground-truth boxes, otherwise it is treated as a negative proposal, this is the default setup and this has not been changed for the test. All results are evaluated for IoU threshold 0.7 for cars and 0.5 for pedestrian and cyclists, this is the default setup and it hasn't been changed in this test.

All the values are in %

experiment	Car	Pedestrian	Cyclist	Comment
PV-RCNN	82.3	66.6	53.3	baseline
modified-PV-RCNN	82.37	66.8	53.3	test1
modified-PV-RCNN	82.33	66.8	53.5	test2
modified-PV-RCNN	82.36	66.9	54.3	test3

baseline - PV-RCNN baseline rerun on the environment described above.
test1 - modified PV-RCNN only conv3 and conv4 features are deformed
test2 - modified PV-RCNN conv1, conv2, conv3 and conv4 features are defornmed
test3 - conv3 and conv4 features are deformed and also the keypoints gating turned on as described in section V (B).

Comparison with state-of-the-art methods : The proposed deformation of keypoints encoding in, test1, with respect to conv3 and conv4 improves performance for pedestrian and car as pompared to PV-RCNN baseline. In, test2, conv1, conv2,conv3 and conv4 keypoints encoding have been considered, this has improved car, pedestrian and cyclist as pompared to PV-RCNN baseline. Finally, in test3, deformed keypoints encoding are considered for conv3 and conv4 and also the sample keypoints were gated for further refinement as discussed in section V(B), this has resulted in slight improvement in pedestrian performance and also car.

VII. *Challenges*

To Trained the entire network, I tried AWS instance with 4 Tesla M-60 GPUs. However, the PV-RCNN code is does not support multi-gpu on AWS instance [41]. So tried to modify the train.py, the training did not work in multi-GPU setup and started throwing GPU out of memory error. I spent a lot of time to fix the issue and also reached out to PV-RCNN author, but the issue couldn't be fixed. So I used 1 GPU AWS instance with batch-size 4 and 15 epochs, this took about 20hrs for each run. A number of more tests could have been conducted for for easy, moderate and hard setup for various combinations of deformations. Due to compute issues, I did only 3 set of tests as discussed.

VIII. *Future Enhancements*

Improve AWS integration of PV-RCNN.
Improve the keypoint encoding to detect smaller objects like small dog, human in sitting position and to detect small objects lying on road. This will require more training data.

IX. *Team Member Contribution*

I am the only contributor to this project.

X. *Github*

Modified PV − RCNN
https://github.com/suvasis/cs230

REFERENCES

[1] https://arxiv.org/pdf/1912.13192.pdf
PV-RCNN:Point-Voxel Feature SetAbstraction for 3D Object Detection Shaoshuai Shi1 Chaoxu Guo2,3 Li Jiang4 Zhe Wang2 Jianping Shi2 Xiaogang Wang1 Hongsheng Li1 1CUHK-SenseTime Joint Laboratory, The Chinese University of Hong Kong 2 Sense Time Research 3NLPR, CASIA 4CSE, CUHK

[2] http://www.cvlibs.net/datasets/kitti/raw_data.php
KITTI dataset

[3] PointNet
Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification andsegmentation.InProceedingsoftheIEEEConferenceon Computer Vision and Pattern Recognition, pages 652–660, 2017.

[4] http://arxiv.org/abs/1906.02739
Gkioxari, G., Malik, J., Johnson, J.: Mesh R-CNN. CoRR abs/1906.02739 (2019),

[5] https://arxiv.org/pdf/1703.00854.pdf P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, 2001, pp. I–511–I–518 vol.1.

[6] https://github.com/facebookresearch/detectron2/blob/master/configs/COCO-PanopticSegmentation/panoptic_fpn_R_101_3x.yaml
O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, Dec 2015.

[7] https://github.com/facebookresearch/Detectron
R. Girshick, "Fast R-CNN," in Proc. IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448. [7] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 2147–2156.

[8] https://github.com/facebookresearch/Detectron
R. Girshick, "Fast R-CNN," in Proc. IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448. [7] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 2147–2156.

[9] https://pdfs.semanticscholar.org/3a9a/f3088deb54500b600be211b4c63afcdc6a7e.pdf Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 1903–1911.

[10] http://dl.acm.org/citation.cfm?id=2969239.2969287
X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 424–432. [Online].

[11] http://link.springer.com/10.1007/978-3-319-16631-5
P. Babahajiani, L. Fan, and M. Gabbouj, "Object Recognition in 3D Point Cloud of Urban Street Scene," in Asian Conference on Computer Vision (ACCV Workshops), 2015, pp. 177–190. [Online].

[12] http://arxiv.org/abs/1609.06666
M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks," arXiv, 2016. [Online].

[13] http://arxiv.org/abs/1711.06396
Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," arXiv:1711.06396 [cs.CV], 2017.

[14] http://arxiv.org/abs/1609.06666
B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in Proceedings of Robotics: Science and Systems, AnnArbor, Michigan, June 2016.

[15] http://arxiv.org/abs/1609.06666
S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro, "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network," in 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Oct 2017, pp. 102–109.

[16] http://arxiv.org/abs/1609.06666
F. Garcia, D. Martin, A. de la Escalera, and J. M. Armingol, "Sensor fusion methodology for vehicle detection," IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 1, pp. 123–133, Spring 2017.

[17] http://arxiv.org/abs/1609.06666
C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum Point Nets for 3D Object Detection from RGB-D Data," arXiv:1711.08488 [cs.CV], 2017

[18] http://arxiv.org/abs/1609.06666
J. Beltr´an, C. Guindel, F. M. Moreno, D. Cruzado, F. Garc´ıa, and A. de la Escalera, "BirdNet: a 3D object detection framework from LiDAR information," in Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 3517–3523.

[19] http://arxiv.org/abs/1609.06666
Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," Sensors, vol. 18, no. 10, p. 3337, 2018.

[20] http://arxiv.org/abs/1609.06666
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 Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12 697–12 705.

[21] http://arxiv.org/abs/1609.06666
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 European Conference on Computer Vision (ECCV) Workshops, 2018, pp. 197–209.

[22] http://arxiv.org/abs/1609.06666
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 International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 3530–3535.

[23] http://arxiv.org/abs/1609.06666
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. International Conference on 3D Vision (3DV), 2019, pp. 214– 221.

[24] http://arxiv.org/abs/1609.06666
A. Barrera, C. Guindel, J. Beltrán, F. García, BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View, 2020

[25] http://arxiv.org/abs/1609.06666
Xu, D., Anguelov, D., and Jain, A., 2017, "Pointfusion: Deep Sensor Fusion for 3D Bounding Box Estimation," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, June 18–23, pp. 244–253.

[26] https://arxiv.org/pdf/1506.01497.pdf
Faster R-CNN: Towards Real-Time ObjectDetection with Region Proposal Networks

REAL WORLD EXAMPLES

[27] https://github.com/facebookresearch/Detectron
R. Girshick, I. Radosavovic, G. Gkioxari, Detectron, 2018.

[28] http://cocodataset.org
COCO: Common Objects in Context.

[29] http://www.cvlibs.net/publications/Geiger2013IJRR.pdf
Kitti dataset understanding

[30] https://arxiv.org/pdf/1910.04853.pdf
Improving a Quality of 3D Object Detection by Spatial Transformation-Mechanism.

[31] http://www.cvlibs.net/publications/Geiger2013IJRR.pdf
Vision meets Robotics: The KITTI Dataset

[32] https://arxiv.org/pdf/1805.01195.pdf
BirdNet: a 3D Object Detection Frameworkfrom LiDAR information

[33] https://ieeexplore.ieee.org/abstract/document/8885850
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. International Conference on 3D Vision (3DV), 2019, pp. 214– 221.

[34] https://arxiv.org/abs/2003.04188
A. Barrera, C. Guindel, J. Beltrán, F. García, BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View, 2020

[35] https://arxiv.org/abs/2004.01389
J. Yin, J. Shen, C. Guan, D. Zhou, R. Yang, LiDAR-based Online Video Object Detection with Graph-based Message Passing and Spatiotemporal Transformer Attention, 2020. Available: [36] Xu, D., Anguelov, D., and Jain, A., 2017, "Pointfusion: Deep Sensor Fusion for 3D Bounding Box Estimation," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, June 18–23, pp. 244–253.

[36] https://arxiv.org/pdf/1910.04853.pdf
Improving a Quality of 3D Object Detection by Spatial Transformation-Mechanism.

[37] https://github.com/facebookresearch/detectron2,2019
Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2,"

[38] https://github.com/beltransen/lidar_bev
J. Beltrán, F. García, lidar-bev

[39] https://github.com/tomas789/kitti2bag
Kitti2bag,

[40] http://arxiv.org/abs/1612.08083
Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. CoRR abs/1612.08083 (2016),

[41] https://aws.amazon.com/blogs/machine-learning/
multi-gpu-distributed-deep-learning-training-at-scale-on-aws-with-ubuntu18-dlami-efa-on-p3dn-instances-and-amazon-fsx-for-lustre/
multi-GPU AWS and pytorch

[42] https://arxiv.org/pdf/2008.08766.pdf
deformed pv- rcnn