École Polytechnique
Département d'Informatique

INF592 Research Internship Report

Voxel-based approach for object detection in point clouds

Realized by :
Duc-Thinh NGO

Supervised by : Maxime KIRGO
At : EDF R&D Paris-Saclay
Address : Boulevard Gaspard Monge, 91120 Palaiseau

Cycle d'ingénieur X2018

Contents

Acknowledgemens

Firstly, I would like to express my deepest thanks to my supervisor Maxime Kirgo, PhD. candidate at EDF R&D and Ecole Polytechnique. His constant guidance and necessary advices played a vital role during the internship period. His suggestions were crucial for making the execution of the report.

My special gratitude goes to Guillaume Thibault and Guillaume Terrasse, researchers at EDF R&D for their excellent support and encouragement in spite of being extraordinarily busy with their duties.

I would like to convey my appreciation and thanks to EDF for their generous support with powerful equipment and HPC clusters to carry out my experiments.

I appreciate this opportunity to work in RVVS team in EDF R&D Paris Saclay. I will strive to use the skills and knowledge gained in the best possible way, and I will continue to improve them to achieve my desired career goals.

Introduction

Over the past few years, deep learning has emerged as the primary effective approach for several problems in the field of computer vision. Among fundamental tasks in computer vision, several works attempted to solve the problem of object detection on images. In a few recent years, the increasing popularity of point cloud datasets acquired from specialized devices such as LiDAR scanners, as well as the development of autonomous vehicles, has attracted the research community to extend their work on 3D vision tasks.

In brief, point cloud is a set of points defined a 3D metric space. It is the simplest representation of 3D geometric information. However, due to the lack of a regular structure, it is often difficult to perform large-scale computations and extract geometric information from a point cloud. PointNet [1] is the pioneering work to propose a neural network architecture on point clouds. Other works attempt to embed point clouds into other regular data structure: such as 2D images [2] or voxel grid [3].

Object detection is a classical task in computer vision. It requires to localize and classify objects at the same time in an environment, such as 2D images or 3D scenes. R-CNN [4] is the first to exploit a deep learning approach to detect objects in 2D images. Nowadays, detectors can be classified into two groups: one-stage and two-stage detectors. A two-stage detector is a combination of (i) a region proposal network (RPN), which proposes regions of interest (RoI) from input data; and (ii) a refinement network that perfoms RoI pooling and process feature maps to classify and refine predicted RoIs. Whereas, an one-stage detector generates predictions by choosing the bounding box with the highest confidence score from a set of several boxes without any refinement step. On one hand, the one-stage architecture offers a memory and speed efficient model. On the other hand, the two-stage architecture offers a robust detector.

In this internship, we study the voxel-based approach to the object detection task on point cloud data. To this end, in the first chapter, we review existing methods. In the second chapter, an analysis on the dataset of EDF is performed. The third chapter introduces works and experiments we performed during the project. In the final chapter, we conclude and discuss on future work.

Chapter 1

Literature review

Introduction

For this review , we focus our scope on two prominent classes of methods for 3D point cloud processing: 1. point-based architectures that learn from raw point coordinates a function, represented by a deep neural network, providing per-point local features; and 2. voxel-based architectures that apply 3D convolution operators to a voxel grid constructed from the point cloud.. Despite the advantage in processing directly on point coordinates and features, point-based detection methods demands a careful architecture design to perform convolutions on irregular data. In contrast, voxel-based detection can leverage the success of 2D convolutional neural networks (CNNs) by translating architectures directly to the 3D setup. In addition, there exists works which exploit different approaches to resolve the task, such as: by multi-view projection [5], [6], or by graph neural network [7].

Beside the two main classes of algorithms which are discussed in section 1.1 and 1.2, the section 1.3 covers the methods that combine both.

1.1 Point-based detection

PointNet [1], [8] is a pioneering work for raw 3D point cloud processing. It first learns an affine transformation matrix to align point clouds to a canonical space before feature extraction. Fully connected networks (FCNs), which are shared between points, are performed directly on point coordinates in order to learn point-wise features. Max pooling, which is a symmetric function, is applied on the entire set of per-point features in order to retrieve the global features of the entire point cloud while making these features invariant to the permutation of points.

However, PointNet is limited at its point-based nature. While it has to learn a function for local aggregation, the convolution operator is mathematically well-defined and CNNs excel at enlarging the receptive field. More recently, some attempts have been made to construct a convolution on the irregular format of point clouds [9], [10].

In resolving the object detection task, there are works attempting to leverage point-based approach. Frustum-Pointnet [11] is the pioneering work which takes advantage of point-based architecture to detect objects. It utilizes RGBD images to generate proposals to group points in 3D frustums then applying PointNet for box predictions. Point-RCNN [12] processes raw point clouds and also follows a two-stage architectures: it first generates 3D RoIs direcly from applying PointNet on raw points, then uses features extracted from foreground points to perform box refinement and object recognition. Votenet [13] leverages a voting mechanism to generate

center points of objects then uses them for generating region proposals, refines predictions, and achieves a better performance.

1.2 Voxel-based detection

Although voxel-based detectors have the advantage in performing convolutions on sampled grid, works that exploit this approach need to address two main challenges: the quantization loss in the voxelization and the sparsity of data. Most of the architectures studied here either propose new methods to voxelize a point cloud and to encode geometric features in a voxel, or inherit from a state-of-the-art 2D detector architecture and present highly-engineered convolution operators.

DSS [14] is one of the first architectures to integrate the voxelization in a 3D detector. It first re-samples the point cloud into a grid of voxel and leverages the Truncated Signed Distance Function (TSDF) to encode the geometric features into voxels. The model then uses a joint amodal object recognition network which is composed of both 3D and 2D object recognition networks. However, due to to the sparsity of data, the dense 3D grid reduces the processing speed of the network and occupies a large amount of memory.

Later, a series of works introduces new methods to voxelize the point cloud and generate the detection. Voxelnet [15] addressed the sparsity of 3D voxel-based data by implementing a voxel feature encoder (VFE). The idea is to leverage a hash table to store voxel coordinates and to search for the voxel to which a point belongs, then concatenate point-wise features to form the features for a voxel. This facilitates the voxelization and ignores voxels with few points, but to perform convolution operators, it is required that sparse voxel-wise structures are reorganized to the dense grid. However, since the point cloud is a set of points that are uneven distributed in the space, a single setting for the voxel grid might lead to the quantitative loss of information during the voxelization. Realizing this shortcoming, Voxel-FPN [16] leverages a multi-scale voxelization module and a feature pyramid network (FPN) in the architecture.

Further, [17]–[19] propose new definitions for the convolution operator on sparse data. SECOND [20] and Minkowski Engine [21] further improve these concepts with high-level engineering implementations to boost their speed. Along with the innovation of the sparse convolution algorithm, the voxel-based approach becomes more and more popular.

SECOND [20] was one of the first works that integrate sparse convolution in their detection pipelines. It uses VFE layers to voxelize the point cloud into a sparse grid, then applies a sparse convolutional middle extractor to learn and aggregate information along the z-axis. The sparse 3D-spatial latent tensors are then reshaped into a 2D-spatial tensor by concatenating features along the z-axis. Finally, a 2D-spatial Region Proposal Network (RPN) generates the detection. This method combines the voxel-based and the bird's-eye-view-based (BEV) approaches, while leverages the sparse convolution to learn features along the z-axis.

GSDN [22] exploited Minkowski Engine for sparse convolution. In general, it leverages a sparse voxel FPN as its core architecture. In the decoder, at each upsample stage, a transposed convolution is applied to generate new voxels then unnecessary supports that do not contribute to the prediction are pruned out to save space. One important point to note is that GSDN predicts 3D boxes directly on sparse 3D-spatial tensors without encoding or projecting it on lower dimensional representation, such as 2D bird's-eye-view feature maps.

1.3 Fusion-based detection

On one hand, point-based detection excels in processing raw point clouds and can naturally handle sparse data. On the other hand, the voxel-based detectors are efficient in performing convolutions and enlarging receptive field. Recent studies are conducted to bridge the gap between these two approaches.

Fast Point R-CNN [23] leverages a two-stage detection pipeline in which, a RPN is a 2D convolutional network applied on a pseudo-image which is encoded along the z-axis from a raw point cloud. Later, a RefinerNet, which utilizes PointNet, extract features directly from the point cloud and aggregate with the features generated from the voxel RPN to classify and to refine the final 3D box regressions.

PV-RCNN [24] takes advantage of the voxel representation of SECOND [20] as its RPN and refers from point-based architecture of Point-RCNN [12] to design a module which can segments foreground/background keypoints, downsampled from raw point cloud, and aggregate features encoded in the voxel backbone. Finally, it performs a RoI pooling operator and leverages grid points, whose features are extracted from the point-based network, to refine the predictions. PV-RCNN++ [25] further improves the keypoint sampling method. Both methods are claimed to score a state-of-the-art results on large-scale driving scenes [26], [27].

In this internship, we study GSDN and PV-RCNN. While the former is a purely-voxel approach and is claimed to score the highest result in [28], the latter one is published with code which will facilitate the reproduction of results.

Chapter 2

Data processing and analysis

2.1 Reorganize and annotate data for object detection task

In this project, we will build a 3D detector for the dataset Chaufferie (Boiler Room) provided by EDF. This dataset composes of 67 scans acquired from a RGB LiDAR scanner, and has been pre-processed, denoised, annotated and ready for use. However, to prepare the data for the detection task, it still demands some extra work.

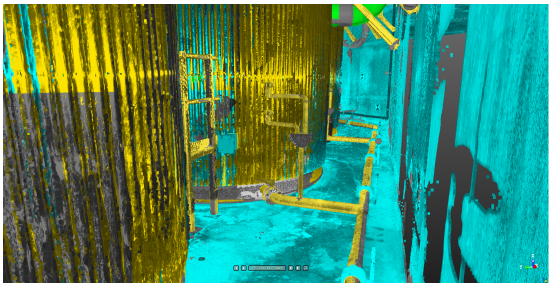
The dataset is stored as a RealWork project on Trimble, which combines two parts:

- A raw point cloud acquired from the LiDAR scanner which includes point coordinates accompanied by their features: RGB colors and intensity. There are 53 stations around the boiler room, each of which provides with a scan. Although being pre-processed, these point clouds are not annotated at all. (Figure 2.1c)
- A combination of meshes and 3D geometric models reconstructed from the raw LiDAR point clouds by EDF's engineers. Each mesh and model corresponds to an object. These, however, contain only point coordinates along with the name of the scan to which each point belongs. (Figure 2.1a and 2.1b)

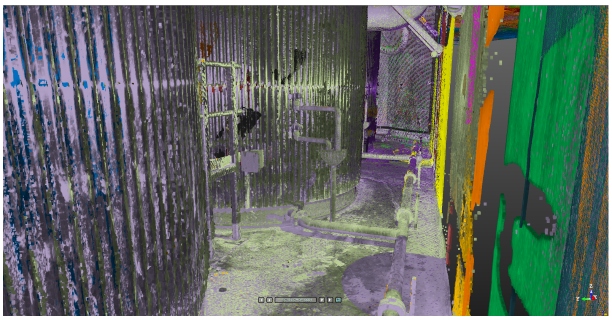
We decided to reorganize the dataset into a structure suitable for object detection task. Firstly, by using a Trimble's macro developed by EDF, we can extract raw point clouds into *.pts* files. Each file contains the point cloud corresponding to a scanner station, including point coordinates (x, y, z-axis) and their features: RGB values under the format of three 1-byte unsigned integers, intensity as an signed short integer. Each scan contains $14 - 50$ million points. Besides, the macro also extracts points from 3D models in *.asc* files and organizes into the hierarchy as following:

```
/
├── GC ..... Backgrounds
│   └── GC_SOLS ..... Ground
│       ├── OBJECT4638_SCAN_6.asc
│       └── OBJECT4638_SCAN_7.asc
└── RECONSTRUCTION_CHAUFFERIE ..... Reconstructed objects
    ├── ECHEL ..... Ladders
    │   └── ECHEL_001 ..... Ladder no.1
    │       ├── OBJECT3581_SCAN_26.asc
    │       ├── OBJECT3581_SCAN_31.asc
    │       └── OBJECT3582_SCAN_61.asc
    └── VANNE ..... Valves
        └── VANNE_002 ..... Valve no.2
            └── OBJECT386_SCAN_47.asc
```

(a) Reconstructed meshes



(b) Point clouds of reconstructed objects



(c) Registered raw LiDAR scans

Figure 2.1: A view of Chaufferie in 3 different modes provided by Trimble RealWorks

The name of each *.asc* file contains the information about that point cloud. For example, `OBJECT3581_SCAN_26.asc` in the directory `ECHEL_001` means that the point cloud belongs to the "object" 3581, reconstructed from the scan 26, and presents a part of the ladder 001. The "object" here indicates the primitive geometry as a part of an object, which we can ignore for now. Using this information, we concatenate point clouds of each object in the same scene into a segmented point cloud corresponding to each scan station. However, due to the aforementioned characteristics of the 3D models, the segmented point clouds do not contain any features of points beside their coordinates. In order to retrieve points features, we project the segmented point clouds onto the raw point clouds. For each point in the segmented point clouds, we search for its nearest neighbor in the raw points then extracts features from its nearest neighbor. In practice, we downsampled both the raw point cloud and the segmented point cloud using voxel-based downsample[1] with the voxel size of $5\,\mathrm{mm}$. We leverage a k-D tree search algorithm for nearest neighbor searching. Finally, we store the dataset in the *.ply* format. Figure 2.2 illustrates a segmented point cloud.

Annotating data for object detection task corresponds to generating 3D bounding boxes for objects in scenes. We decided to utilize the oriented bounding boxes that are aligned along the z-axis as labels. This is a commonly-used type of oriented bounding boxes in other datasets [26], [27], [29]. In order to infer a 3D oriented bounding box for a point cloud of an object, we project points on the xy plane, find the 2D oriented bounding box of thoses projected points. This can be done by computing the two eigenvectors of the covariance matrix of projected point coordinates. The 2D oriented bounding box is exactly the axis-aligned bounding box in the basis of these two vectors. Expanding this 2D oriented bounding box along the z-axis by the height of the object, we obtain the z-aligned oriented bounding box.

1. http://www.open3d.org/docs/0.6.0/python_api/open3d.geometry.voxel_down_sample.html

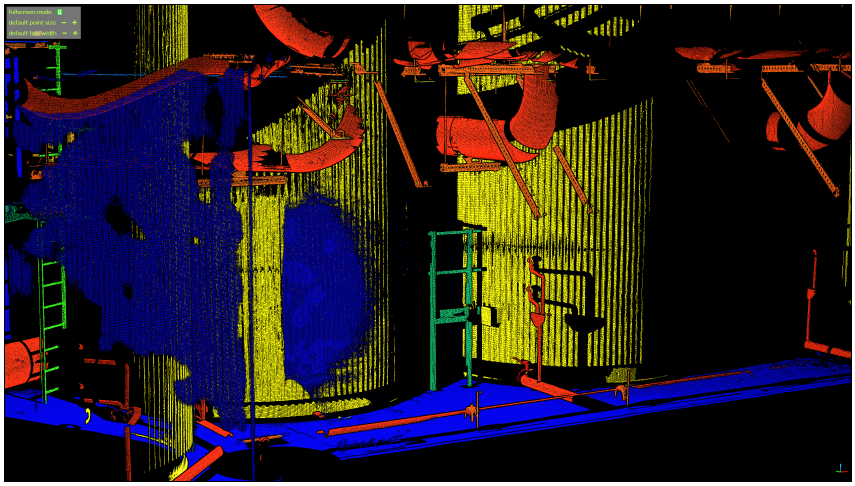


Figure 2.2: A segmented point cloud. Objects of different classes are illustrated by different colors.

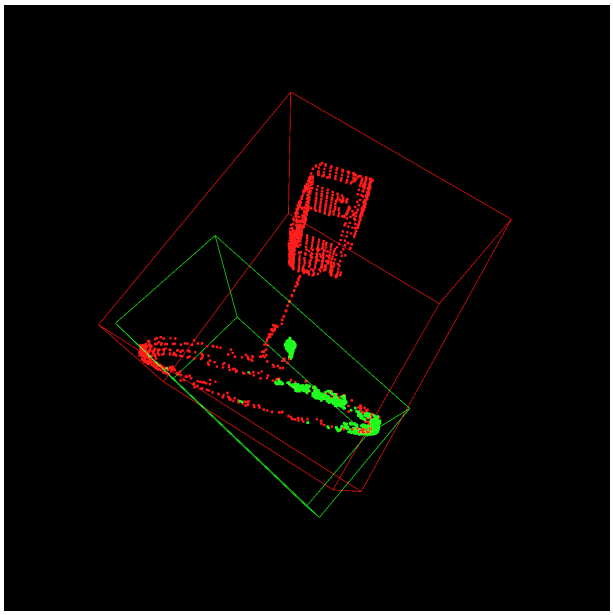


Figure 2.3: Comparison of normal and amodal bounding boxes. Red points and red box present for the full registered point cloud of the valve and its amodal box. Green points and green box present for the point cloud of valve from a single scan and its bounding box.

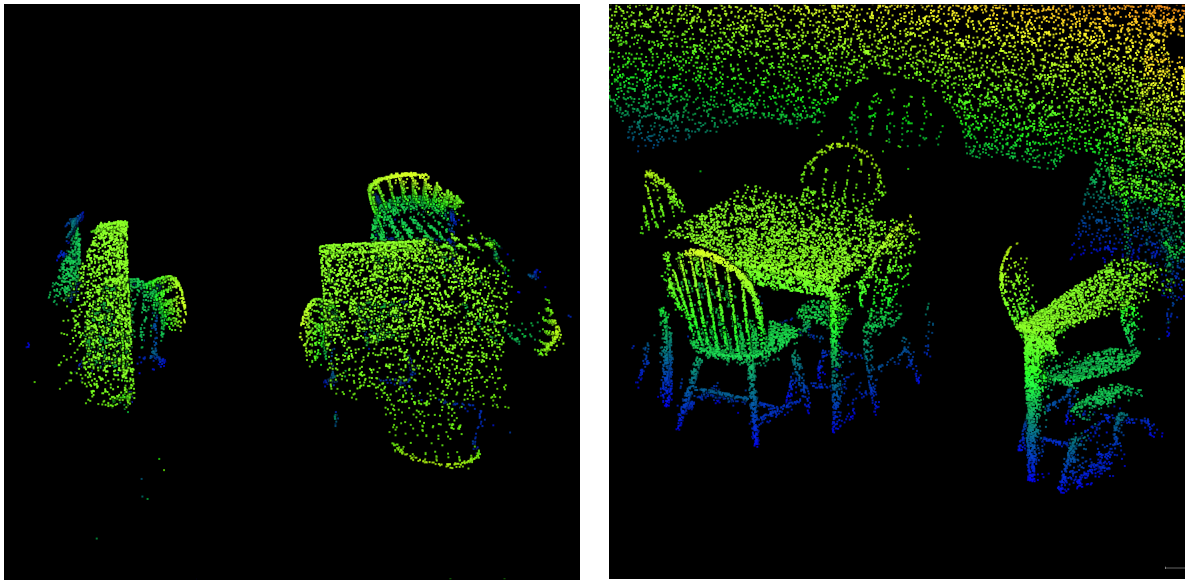


Figure 2.4: Objects in indoor scenes are close to each other. Therefore, without a careful design, the embedding of point clouds into multi-view or voxel may result in the disentanglement of several objects in a pixel/voxel (Point cloud is taken from a Sun-RGBD scene. Ground is removed on purpose for a better visualization.)

Furthermore, we have two options for generating a bounding box: generate a bounding box of only the part of object that appears in the scene, or generate an amodal bounding box (Figure 2.3). On one hand, the first option is suitable for large objects which tend to spread on multiple scenes such as ladders, gratings, pipes. On the other hand, the amodal bounding boxes are more suitable with the objects at a smaller scale, such as valves, technical boxes. The bounding boxes in the first option can be naturally inferred from the segmented point clouds. However, the amodal bounding boxes requires to re-utilize the point clouds extracted from the 3D models. We concatenated all points belonging to an object and compute its amodal bounding box. Considering the name of object in a scene, we label it with the amodal bounding box pre-computed.

2.2 Data analysis

In general, unlike large-scale outdoor scenes, indoor datasets, have two main differences:

1. The distribution of objects inside the scenes. It is noticed that in outdoor scenes, where only cars, bicycles and pedestrians are interested instances, objects are sparsely distributed. Whereas in indoor scenes, objects are usually close to each other and even occluded along the z-axis (Figure 2.4). The most frequent cases are where the lamps are placed on the surface of the night-stands, or chairs are hidden under tables. In other words, bounding boxes of objects in indoor scenes are usually overlapped on each other.
2. A large difference in the size of scenes and objects. In autonomous driving scenes, the field of vision often ranges to $70\,\mathrm{m}$ in the xy-axes and $3\,\mathrm{m}$ in height [26], [27]. While in the indoor datasets, a scene is usually limited in a room whose average area is about $22\,\mathrm{m}^2$ [28], or sizes range to $5\,\mathrm{m}$ in the xy-axes [29]. Moreover, objects in indoor scenes have a variety of sizes compared to objects in autonomous driving scenes where interested objects are only vehicles and pedestrians.

In spite of being a set of scans of a boiler room, Chaufferie differs from Scannet and Sun-RGBD. It has a vision field ranging to $32\,\mathrm{m}$ in the horizontal direction and $10\,\mathrm{m}$ in the vertical

direction. Each scan is not limited in a rectangular space either. However, the distribution of objects is similar in the case of indoor scenes since it has a diversity of objects placed close to each other. Above all, Chaufferie differs from all mentioned datasets in the positions of objects: some are not restricted to be placed on the ground.

Unfortunately, almost all of novel works support the voxel-based approach only evaluate on autonomous driving scenes [15], [16], [20], [23]–[25], [30]. GSDN [22] and DSS [14] are exceptions, which addressed to indoor datasets. In contrast, several point-based methods cover the indoor scene understanding [13], [31]. Therefore, in this project, we study the voxel-based methods but still keeps an eye on the point-based approaches. In addition, since Chaufferie has special characteristics comparing to existed datasets, we also evaluate our proposed methods on another indoor dataset, Sun-RGBD [29], which has been widely researched and is the closest to our application setup.

Chapter 3

Building models

In this section, we describe our studies on PV-RCNN and GSDN that are claimed to be state-of-the-arts on KITTI and Scannet respectively. First of all, we will provide with some basic modules that are commonly used in several voxel-based 3D detectors. In section 3.2, we will detail the architecture of PV-RCNN and how we modify and adapt it on Sun-RGBD and Chaufferie. In section 3.3, we will review GSDN's architecture and describe our own implementation.

3.1 Overview of 3D detectors

Pre and Post-processing

Point clouds before being fed to the detectors are always translated so that they center at the origin. While for Sun-RGBD, no further pre-processing is needed, point clouds in Chaufferie are cropped and sampled. This aims to resolve the great difference between the scale of scenes and of the objects of interest (such as valves and technical boxes). In training, scenes are randomly cropped so that at least an interested object remains in scenes. In inference time, in order to prevent the randomness and to provide a trusted metric, we tile Chaufferie scenes following two settings of grid. The evaluation of models will be performed on this tiled scenes. In training time, data is augmented using the techniques: rotation by a random angle, random flipping , random scaling factor uniformly sampled from $[0.95, 1.05]$. In outdoor datasets, a scene can also be augmented by adding randomly more objects from other scenes. However, due to the dense distribution of objects in an indoor scene, this method is not suitable.

After the predictions are generated by the detector, many generated bounding boxes for the same object overlap on each other. We filter these predictions by adopting a non-maximum suppression (NMS) and keep only the bounding boxes with the highest confidence score.

Anchor boxes

On each voxel (3D RPN) or pixel (2D RPN), we place a number of pre-defined boxes called anchor box. These anchors, centered at voxels/pixels, are used as a mean to assign classification and box regression targets simultaneously. An anchor is said to contain objects if its IoU (intersection over union) with a ground-truth object is greater than $\tau_{\text{foreground}}$. Conversely, it is set as a background anchor is its IoU with any ground-truth objects is smaller than $\tau_{\text{background}}$. In SSD [32], these boxes are called default boxes and are assigned to feature map cells at different scales. Moreover, beside giving the objectness score and box regression, it even predicts the class of object.

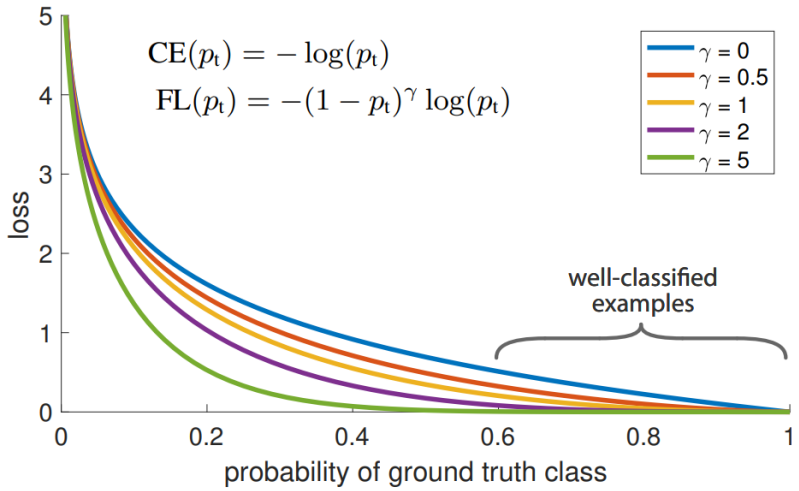


Figure 3.1: focal loss

Focal loss for object detection

[33] proposes focal loss function aiming to address the class imbalance in detectors. To simplify the problem, we starting from the cross-entropy loss for binary classification. The extending to multi-label is straightforward. The binary cross-entropy loss is defined as:

$$\mathbf{CE}(p, y) = -\log(p_t).$$

where $p_t = (1-y)(1-p) + yp$ with $y \in \{0, 1\}$ specifying the ground-truth class. To tackle with the class imbalance, where easy negative examples dominate, a focal loss is proposed by adding a modulating factor a the cross-entropy loss:

$$\mathbf{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t).$$

where $\alpha_t = (1-y)(1-\alpha) + y\alpha$ with $\alpha \in [0, 1]$. This α_t can be considered as an hyper-parameter for tuning. The comparison between focal loss with several γ is shown in Figure 3.1. In practice, we set $\gamma = 2$ and tune α. For PV-RCNN, α is set at 0.25 following the setting in [33]. While in GSDN, we set $\alpha = 0.8$. This is resulted from several attempts and the observation on the precision and recall scores obtained from the classifier. When both precision and recall scores are too low, smaller than 0.1, it means that the classifier does not return enough positive predictions due to the domination of negative examples. We increase α until observing that both precision and recall scores increase to a sufficient level. Particularly in GSDN, focal loss is used in sparsity prediction, serving the pruning module, hence we tune α to achieve a high precision (around 0.8) and a sufficiently high recall (around 0.2) so that it does not prune out too many interesting voxels.

Multi-task loss for 3D detectors

The classification loss includes the objectness loss and the loss of class prediction. For objectness loss, we leverage focal loss for resolving the imbalance between positive-negative anchors. For the loss of classifier, we utilize a focal-loss for multi-class classification or simply a weighted cross-entropy loss. The negative objectness class can also be considered as an additional class for the multi-label classification, allowing to get rid of objectness classifier [32].

The box regression loss is inherited from 2D detection frameworks [34] [32].

$$\mathbf{L}_{\text{reg}} = \sum_{i \in \text{Pos}}^{N} \left(\sum_{m \in \{cx,cy,cz,l,w,h\}} x_{ij} \text{Smooth}_{\text{L1}}(a_i^m - \hat{t}_j^m) + x_{ij} \text{Smooth}_{\text{L1}}(\sin(a_i^\alpha - t_j^\alpha)) \right)$$

where $x_{ij} \in \{0, 1\}$ indicates whether anchor i is matched to ground-truth box j and

$$\hat{t}_i^{cx} = (cx_t - cx_a)/l_a, \quad \hat{t}_i^{cy} = (cy_t - cy_a)/w_a, \quad \hat{t}_i^{cz} = (cz_t - cz_a)/h_a,$$

$$\hat{t}_i^{l} = \log(l_t/l_a), \quad \hat{t}_i^{w} = \log(w_t/w_a), \quad \hat{t}_i^{h} = \log(h_t/h_a).$$

In addition, for 3D oriented bounding boxes, [11] introduces a regularization term. Indeed, the center, size loss terms and heading angle error are separate terms, thus, this may lead to a situation where two terms are imbalanced and the localization loss is dominated by one of these twos. Therefore, beside training to optimize the center, size and heading angle loss terms, a corner loss term is added.

Therefore, multi-task loss function for 3D detectors is

$$\mathbf{L} = \frac{1}{N}(\lambda_{\text{reg}}\mathbf{L}_{\text{reg}} + \lambda_{\text{corner}}\mathbf{L}_{\text{corner}} + \lambda_{\text{objness}}\mathbf{L}_{\text{objectness}} + \lambda_{\text{cls}}\mathbf{L}_{\text{cls}}).$$

Teh total loss is normalized by the number of anchors matched to ground-truth boxes. In practice, we set $\lambda_{\text{reg}} = \lambda_{\text{corner}} = 0.5, \lambda_{\text{objness}} = \lambda_{\text{cls}} = 1$.

Evaluation

The models on both Sun-RGBD and Chaufferie are evaluated on PASCAL-VOC 2012 [35] benchmark. At first, we set an IoU threshold. Detections and ground-truths are matched if and only if their overlapping area is greater than the IoU threshold. One important rule: When several generated predictions are true on a single ground-truth object, the one with highest confidence score is considered as the only true positive prediction, the others are hence false positive. We plot a precision-recall curve, varying by threshold of confidence score. The average-precision is calculated as the area under the precision-recall curve. We use the 11-point interpolation to compute,

$$AP = \frac{1}{11} \sum_{r \in \{0.0, 0.1, ..., 1.0\}} p_{interp}(r),$$

where $p_{interp}(r) = max_{\tilde{r} \geq r} p(\tilde{r})$. The mean average-precision (mAP) is the mean of average precisions of all classes of objects.

3.2 PV-RCNN-based detectors

Architecture review - Vanilla PV-RCNN

PV-RCNN is a fusion between voxel and point-based detectors. It shares the similar idea with Fast Point R-CNN. Both are two-stage detectors. They both proposed a voxel-based backbone for the proposal generation. While in the second stage, point-based approaches were in use for the box refinement. However, 2 major novel modules differ PV-RCNN from the other fuse-based detectors:

- ☐ The 3D Voxel CNN, which utilizes sparse convolution, is adopted as a voxel backbone. By exploiting sparse convolution operators, the PV-RCNN backbone can naturally enlarge the receptive field with a deep CNN without overloading the hardware memory. Furthermore, it allows the model to extract more abstract abstract features in multi-scale

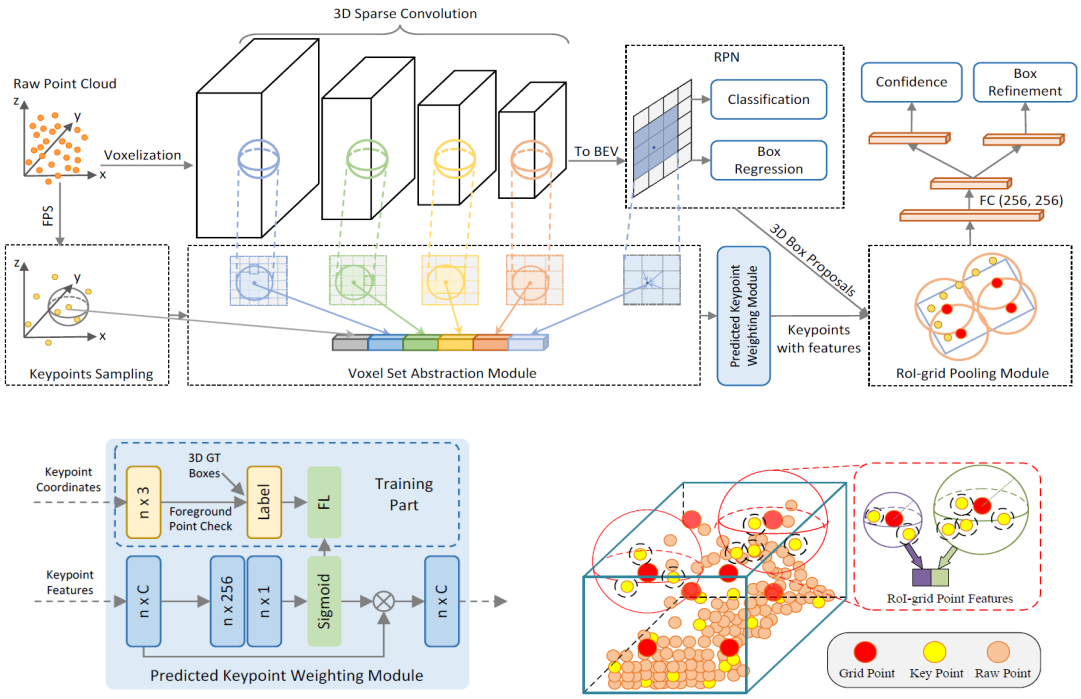


Figure 3.2: Illustration of the whole pipeline of PV-RCNN (**Up**), the Predicted Keypoint Weighting (**Left down**) and RoI-grid pooling module (**Right down**). Predicted Keypoint Weighting is a fully connected network which applies on the features of keypoitns. After going through an activation Sigmoid, it returns the probability of each point indicating whether it is foreground, i.e inside a ground-truth bounding box. At the RoI-grid pooling module, neighboring keypoints inside a radius r of a grid point are located; then a PointNet module is adopted to aggregate features for that grid point from neighboring keypoint features.

while getting rid of voxelizing raw point clouds with different settings of grid as in [16]. After obtaining a 3D-spatial feature map, the model reshapes it into a BEV feature map by stacking features along the z-axis. Another 2D backbone is applied to generate BEV RoI proposals.

- The two-stage point-based strategy to learn features of RoI-grid points. This involves the first voxel-to-keypoint encoding step and the second keypoint-to-grid RoI feature abstraction step. In the first step, a fixed number of keypoints are sampled from the raw point cloud using the Farthest Point Sampling (FPS) algorithm. A Voxel Set Abstraction module encodes multi-scale features obtained from the 3D voxel backbone into keypoint features. This is done by searching k nearest voxels surrounding a keypoint then concatenating their features. Furthermore, a Predicted Keypoint Weighting module learns to weight keypoint features based on an auxiliary module which predicts whether a keypoint is foreground or background. Finally, the keypoint features, which are inside a neighbor of a grid point by radius r, are aggregated by a PointNet [8] to generate RoI-grid point features. A FCN of 2 layers with 256 feature dimensions is learnt to process these RoI-grid point features and concatenate them into features of RoIs. Two other FCNs are adopted to regress confidence score and refine box predictions from each RoI features.

The model reports state-of-the-art results on both Waymo and KITTI datasets.

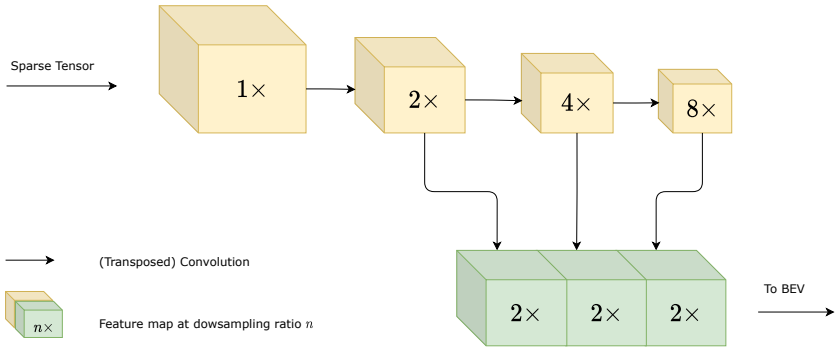


Figure 3.3: Illustration of 3D backbone inspired from VoxelNet [15]

Adaptation for indoor datasets

As mentioned in 2.2, there is a great gap between outdoor and indoor scenes. Since the PV-RCNN was only addressed to autonomous driving scenes, it demands a certain amount of work to adopt it for solving indoor detections. We will describe our work to improve it on Sun-RGBD and Chaufferie step-by-step.

The importance of feature map resolution

At first attempt, we leverage the implementation of PV-RCNN [36] and apply it directly on Sun-RGBD. There is no change in the model. The dataset loader is customized based on implemented dataset loaders. The first qualitative results show that, although the model can detect and localize objects, it gives prediction with low confidence scores. We found out that it was due to the feature maps were downsampled up to 8 times, which resulted in low resolution BEV feature maps. Low quality BEV feature maps lead to low quality region proposals. It was analysed in section 2.2 that indoor datasets differ from outdoor datasets in the terms of scale and object distribution. While a feature map of downsampling ratio $8\times$ encoded from an large-scale outdoor scene may still well represent the object positions, one encoded from an indoor scene may fail to preserve the localization information. After Due to the occlusion between several objects in a single voxel, which now $8^3\times$ larger than the sampled voxel in input. Consequently, the first modification we made to the PV-RCNN is in the 3D voxel backbone.

We refer to the RPN in [15], in which, a $2\times$ downsampled feature map is returned by the 3D backbone (Figure 3.3). This modification shows the improvement in the confidence scores (The overall results are presented in section 3.4). Therefore, we deduce that with a higher quality feature map, the models generates better detections. Later on, instead of only using a $2\times$ feature map, we leverage a U-net for producing a feature map with the resolution exact as the input voxel grid (Figure 3.4).

Design of a 3D U-net and removal of the BEV backbone

However, as we discussed in section 2.2, BEV projections and BEV region proposals are not suitable for Chaufferie dataset. Therefore, we get rid of the BEV CNN and replace BEV anchors with 3D anchors. On one hand, this anchor head allows the model to generate 3D region proposals, leading to a more precise proposal along the z-axis. This allows the detector to localize objects without any supposition about the z position of objects. On the other hand, since 3D anchors are placed on each voxel in a pre-defined dense grid, they occupy a large amount of memory and decelerate the whole training pipeline. In section 3.3, we will discuss

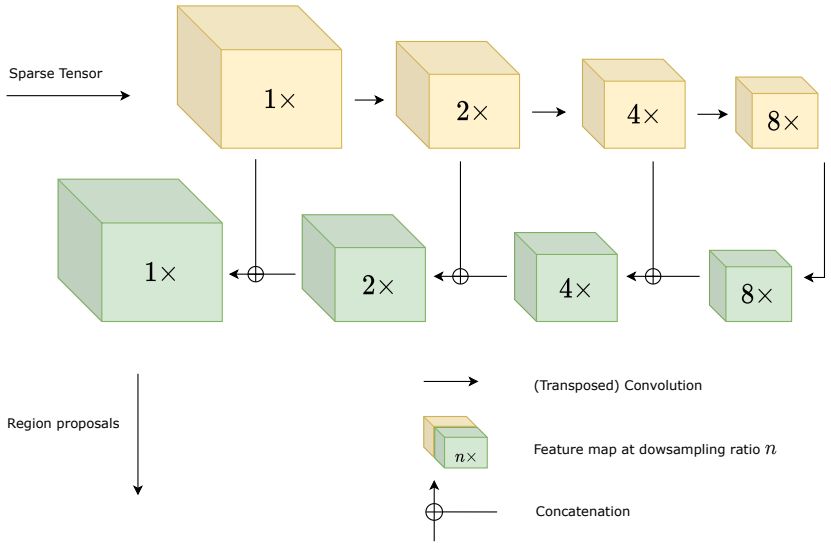


Figure 3.4: Illustration of the U-net based backbone. In the final feature map, before generating region proposals, PV-RCNN U-net Dense process features through a dense residual block. This allows voxels, which is empty before the dense convolution, retrieve features if they can contribute in the generation of RoIs.

about the possibility to prevent this bottleneck with a sparse 3D anchor head. In order to further increase the region proposal quality, we leverage a sparse U-net implemented in [37] as the backbone. Besides, a problem arises due to the sparsity of the raw point clouds and sampled voxel grids [13], [22], [31]. It is observed that, object centers, or simply points whose anchors overlap the most with objects, are in the empty space. Therefore, a backbone which leverages simply the sparse convolution or sub-manifold convolution may result in situation where voxels around the object centers are all empty, contain no valuable information. To this end, after converting the sparse tensors to dense tensors, we perform a series of dense convolutions on these dense tensors. A residual block [38] is added to the output of the U-net backbone, before the generation of proposals. From now, to better distinguish between two models, we call the former without dense residual block at the end of U-net is Sparse PV-RCNN U-net and the latter is Dense PV-RCNN U-net.

These two models are trained and evaluated on both Sun-RGBD and Chaufferie. The results are reported in section 3.4.

3.3 GSDN detector

Architecture review

GSDN inspires from the single-shot detector [32]. It leverages the sparse tensors and sparse convolution to construct a 3D voxel FPN. At each stage stage of the decoder, GSDN generates predictions given the feature maps at that level. GSDN proposes a novel decoder capable of expanding and pruning out voxels. At the end of each decoder stage, a generative sparse transposed convolution is applied to upsample the feature maps and generate new voxels as well. This differs from the U-net backbone in PV-RCNN that we adopted. It generates new voxels which may contribute in predictions, while in U-net inspired from [37], it only associates features to existed voxels, which limits the possibility to generate precise RoI (Figure 3.5). GSDN decoder hence generates more new interested voxels (Figure 3.6). These generated feature

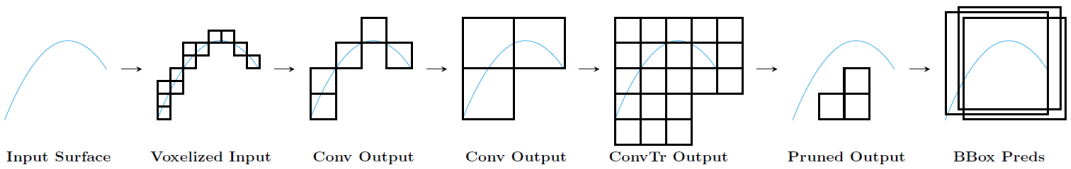


Figure 3.5: Illustration of the benefit of generative and pruning module. While submanifold convolution ignores all empty voxels, a transposed convolution will generate more voxels. This allows the model to generate more voxels. Pruning module will prune out all voxels which are not valuable for the predictions.

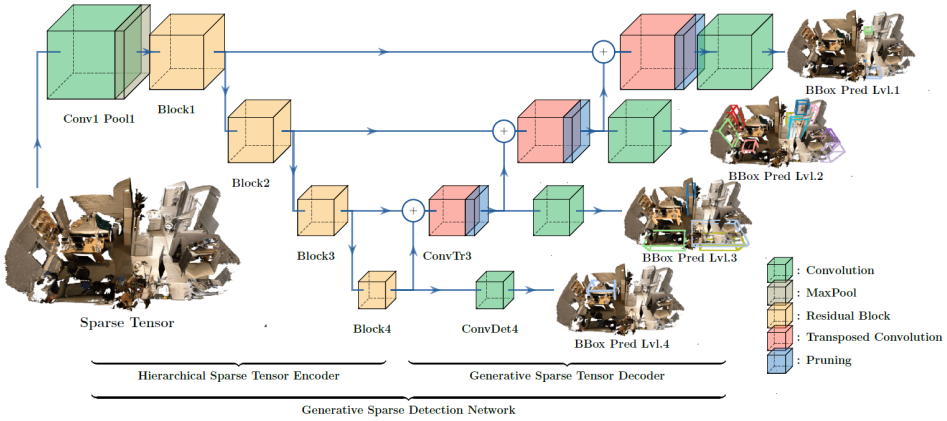


Figure 3.6: Pipeline of GSDN model.

maps are then aggregated (by concatenation or element-wise addition) with the feature maps in the encoder. Moreover, to resolve the cubic increase in the number of voxels, GSDN proposed a pruning module, guided by sparsity predictions. Beside giving sparsity predictions, feature maps are used for generating multi-scale detections.

Our implementation

Since GSDN does not provide source code, we re-implemented the model based on its description. Although having described their architecture, the authors do not explicit each modules in terms of: number of filters, kernel size of convolution and so on. Thus, we implemented it by supposing that GSDN follows the pattern of 2D FPN detailed in [39] with only a change in the decoder, a generative transposed convolution is used instead of upsampling. In general, FPN also has an encoder and a decoder as U-net. Both models can use ResNet as their encoders. However, FPN and U-net are different in their decoders. Instead of using concatenation to aggregate feature maps at the same level of the encoder and decoder, FPN uses a residual connection to aggregate two feature maps. Moreover, at each level of the decoder, FPN generates a feature map to generate multi-scale predictions. All feature maps on the decoder have the same number of channels.

In Figure 3.9, we illustrate an interpretation of GSDN, particularly the points after pruning module.

3.4 Results

For all experiments, we utilize a variant of Adam optimizer, AdamW [40] with the one cycle learning rate schedule [41] with the maximum learning rate is set at 0.05. NMS threshold is set at 0.1. Batch size 16 for PV-RCNN and 32 for GSDN.For PV-RCNN, voxel size on Sun-RGBD is set at $[0.065, 0.0975, 0.06]$, on Chaufferie is set at $[0.03, 0.03, 0.03]$. For GSDN, voxel size is set at $[0.05, 0.05, 0.05]$ for Sun-RGBD.

The training settings for PV-RCNN and its modified versions are the same as already implemented in [36]. We re-tuned only the parameters: $\tau_{\text{foreground}} = 0.5$ and $\tau_{\text{background}} = 0.3$. For GSDN, our implementation leverages parameters $\tau_{\text{foreground}}$ and $\tau_{\text{background}}$ as tuned in PV-RCNN. Further, the parameter for focal loss α is set at 0.8.

We train vanilla PV-RCNN and its variants on 4 GPUs Tesla V-100 of 16GB each. We train GSDN on two GPUs.

3.4.1 On Sun-RGBD

Method	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
Votenet [13]	0.74	0.83	0.29	0.75	0.22	0.30	0.62	0.64	0.47	0.90	0.58
Vanilla PV-RCNN [24]	0.46	0.77	0.18	0.7	0.05	0.14	0.34	0.53	0.4	0.54	0.41
PV-RCNN Voxelnet	0.57	0.78	0.18	0.72	0.09	0.16	0.36	0.6	0.41	0.76	0.46
PV-RCNN U-net Sparse	0.	0.76	0.19	0.63	0.03	0.	0.	0.54	0.18	0.74	0.31
PV-RCNN U-net Dense	0.45	0.69	0.13	0.67	0.06	0.1	0.24	0.6	0.34	0.83	0.41
Our GSDN	0.58	0.83	0.01	0.56	0.10	0.03	0.37	0.56	0.35	0.81	0.44

Table 3.1: Quantitative results of different experiments on Sun-RGBD. IoU threshold is set at 0.25.

We trained five models on Sun-RGBD. The point clouds are generated from RGB-D images as in [13]. In four experiments investigated, only PV-RCNN U-net Dense and GSDN use 3D anchors. The others remain BEV backbone in their architectures with 2D anchors.

The results in table 3.1 shows that, two BEV-projection methods return better or equivalent results compared with the others. The 0.1 gaining from PV-RCNN U-net when comparing to U-net Sparse shows that the addition of a dense residual block in the output of U-net backbone is necessary. However, 3D anchors seem not to add in robustness for model in this dataset. This may be due to the fact that most of these objects placed on the ground. The addition of anchors which are placed in the middle of space will only increase the number of negative examples, but does not contribute in the performance of the detector.

The experiment with GSDN does not end up as expected. However, when we train it and PV-RCNN U-net Dense on Sun-RGBD to detect only class *chair*, then compare between these twos. Both quantitative (Table 3.2) and qualitative results (Figure 3.7) show that the detector works. This may be due to the failure of the model to generalize the classification over multiple classes. The imbalance between classes is also an important problem.

3.4.2 On Chaufferie

We focus on the detection of class *valve*. The infra-difference of the class is shown in Figure 3.10. There is a great difference between shapes and sizes of the two types of valves. Moreover, statistics in Figure 3.11 show that, the number of ball valves dominate in the class *valves*. This may trouble the detector since ball valves are pretty small, with the mean size ranging only to $0.1\,\mathrm{m}$. Thus, as already mentioned in section 3.1 we have to re-sample the scan and crop

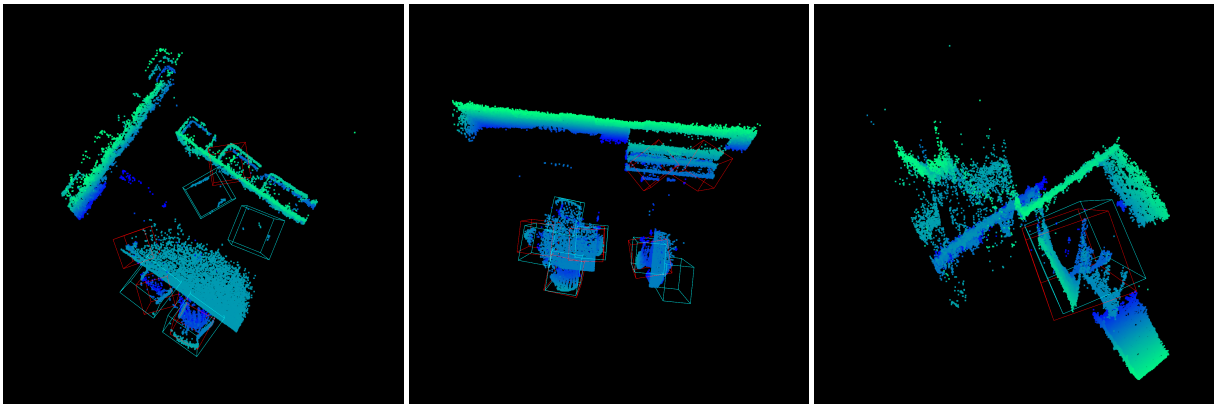


Figure 3.7: Three qualitative results of our implementation of GSDN on Sun-RGBD (only class *chair*). Cyan bounding boxes are ground-truths and red boxes are predictions.

Method	chair
PV-RCNN U-net Dense	0.75
Our GSDN	0.65

Table 3.2: Quantitative results of PV-RCNN U-net Dense and our implementation for GSDN when being trained on Sun-RGBD to detect only class *chair*.

it. Firstly, to prevent the great infra-difference in the class, we train the model respectively on: only round valves, only ball valves and mixed. As shown in Table 3.3, the detector work best on round valves, where APs at IoU threshold 0.10 and 0.25 are equivalent. In other words, the detectors succeed at localizing round valves. It is not the same as in ball valves. There is a big difference between APs at two IoU thresholds, which means, the detector can recognize ball valves but fail to regress the bounding boxes precisely. Finally, the results on all class *valves* are between two results above. Qualitative results are shown in Figure 3.12.

Moreover, in this experiment, we can notify that Vanilla PV-RCNNN (with 3D anchors) does not score as high as PV-RCNN U-net Dense.

Method	Round valves		Ball valves		All valves	
	AP@0.10	AP@0.25	AP@0.10	AP@0.25	AP@0.10	AP@0.25
Vanilla PV-RCNN [24]	0.57	0.52	NA	NA	NA	NA
SECOND [20]	0.22	0.22	NA	NA	NA	NA
PV-RCNN U-net Dense	0.84	0.84	0.30	0.05	0.68	0.36

Table 3.3: Quantitative results of different experiments on Chaufferie. Note that the implementation of Vanilla PV-RCNN and SECOND in these experiments removes BEV backbone and replaced by a 3D anchor. NA indicates that no experiment is conducted.

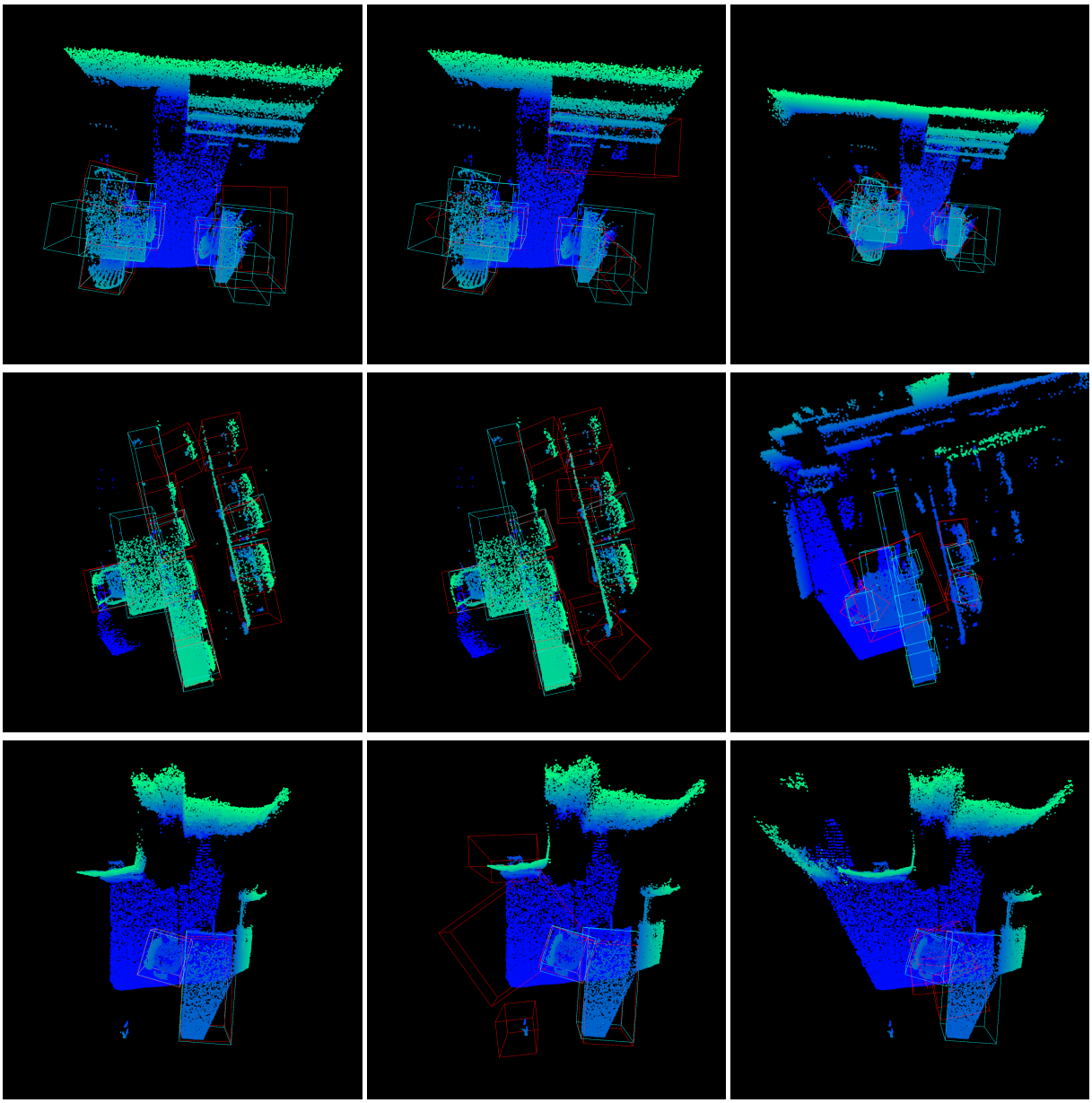


Figure 3.8: Comparison between predictions of three models. **Left**: PV-RCNN U-net Dense. **Middle**: PV-RCNN U-net Sparse. **Right**: Our GSDN implementation. Cyan bounding boxes are ground-truths and red boxes are predictions.

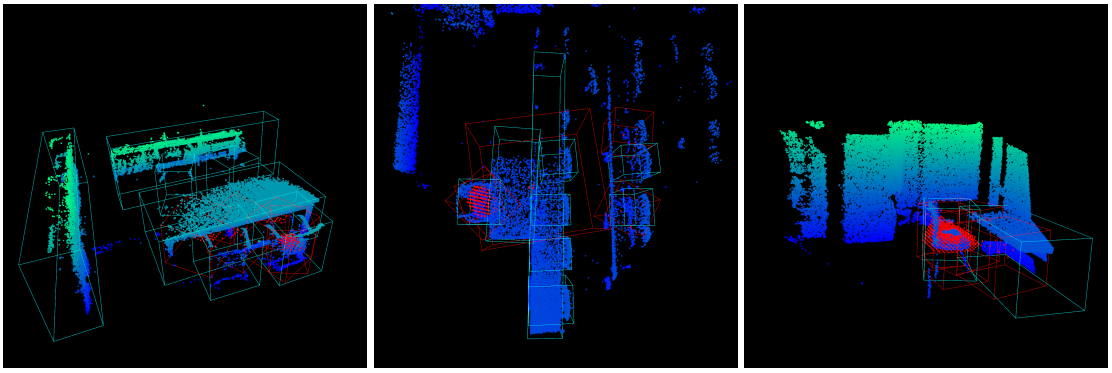


Figure 3.9: Illustrations of predictions generated from GSDN after training on Sun-RGBD. Red points are points which are kept after the pruning module. Figures on the left and on the middle shows red points locating in the center of chairs and tables. Despite being misclassified, the detector remains succeeding at generate center points, meaning points to which anchors attached contain objects. These results show that the GSDN indeed, follows the same principle as in Votenet [13]. This bridges the gap between two approaches, point-based and voxel-based.

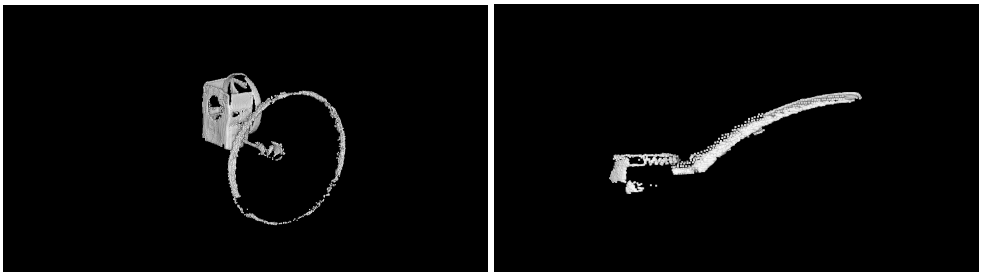


Figure 3.10: Two major types of valves. **Left**: A round valve with average dimension of $0.5\,\mathrm{m}$. Right: A handle of a ball valve with average dimension of $0.1\,\mathrm{m}$.

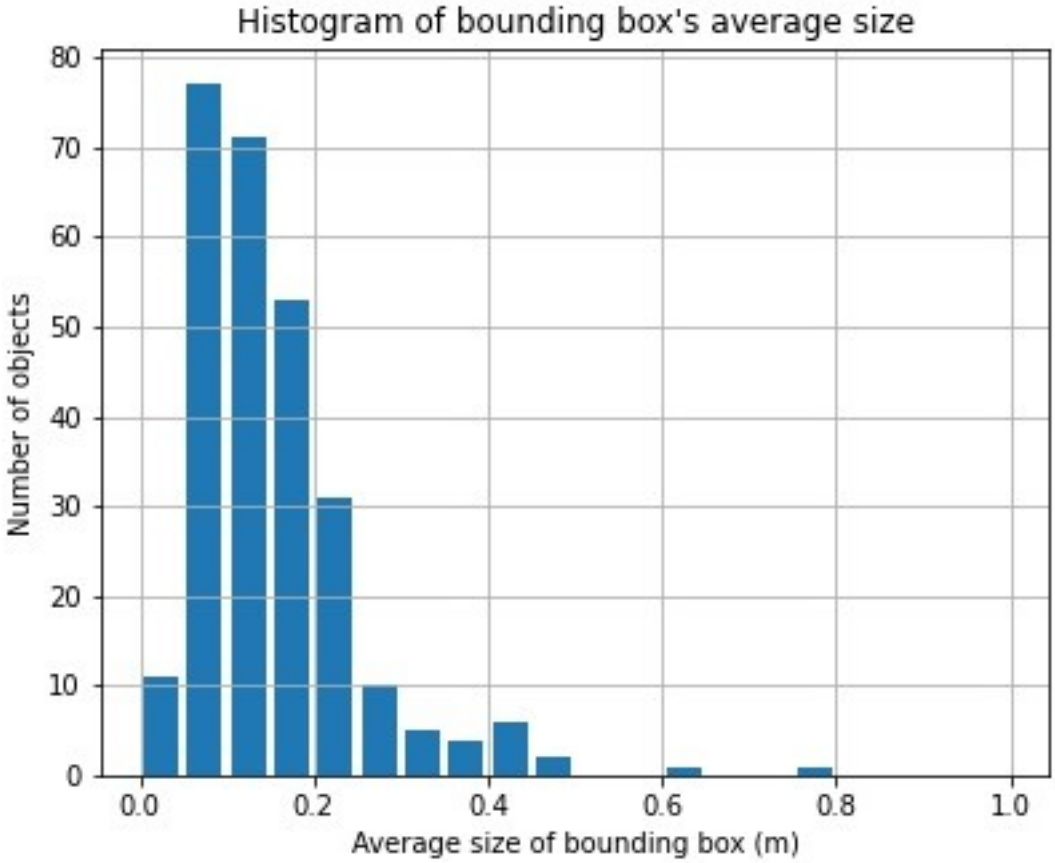


Figure 3.11: A histogram showing the distribution of object sizes in class *valves*.

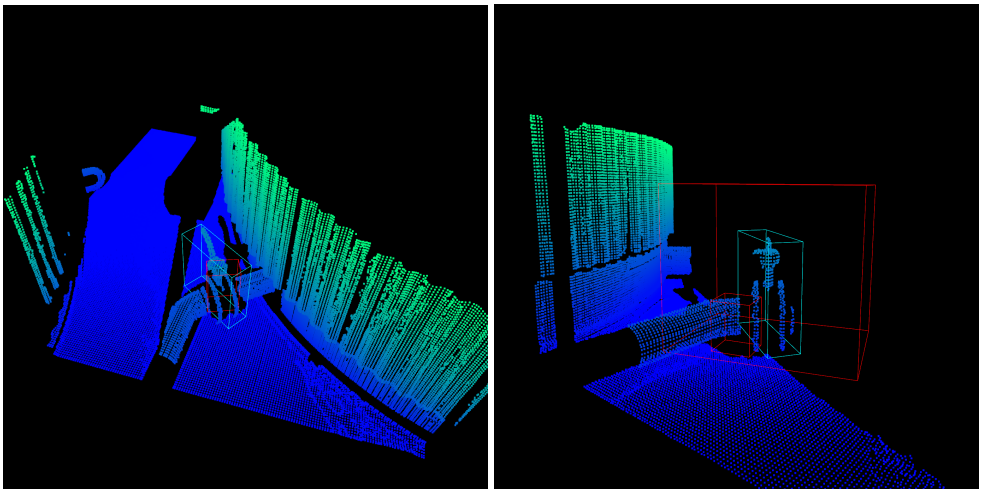


Figure 3.12: To qualitative results of PV-RCNN U-net Dense trained on the entire class *valve*.

Chapter 4

Conclusion

The project corresponds to the study of two main models: PV-RCNN and GSDN. Different variants of PV-RCNN are proposed and evaluated on both Sun-RGBD and a dataset of EDF - Chaufferie. Proposed models haved worked on Chaufferie but remained future work:

- Generalize the detector to detect other classes of objects in Chaufferie, technical boxes for example.
- Improve performance of constructed models or propose more robust designs. It should not only work well on Chaufferie but also score a state-of-the-art result on another similar dataset has been widely researched.

For works on GSDN, it is necessary to debug/improve the models so that it gives better results on Sun-RGBD. Meanwhile, an experiment on Chaufferie should be considered as well. Once succeeded with GSDN, it is also interesting to leverage generating/pruning modules in the 3D backbone of PV-RCNN.

Throughout the project, we have studied only classical detectors with the mechanism of anchors so far. Besides, there are also other interesting works that perform the detection without any preset anchors or proposals [42]–[46]. Future work may exploit these directions, which are capable of better generalizing the object detectors.

Bibliography

[1] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593.

[2] Hang Su, Subhransu Maji, Evangelos Kalogerakis, et al. "Multi-view Convolutional Neural Networks for 3D Shape Recognition". In: *CoRR* abs/1505.00880 (2015). arXiv: 1505.00880. URL: http://arxiv.org/abs/1505.00880.

[3] Daniel Maturana and Sebastian Scherer. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[4] Ross B. Girshick, Jeff Donahue, Trevor Darrell, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[5] Jason Ku, Melissa Mozifian, Jungwook Lee, et al. "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *CoRR* abs/1712.02294 (2017). arXiv: 1712.02294. URL: http://arxiv.org/abs/1712.02294.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, et al. "Multi-View 3D Object Detection Network for Autonomous Driving". In: *CoRR* abs/1611.07759 (2016). arXiv: 1611.07759. URL: http://arxiv.org/abs/1611.07759.

[7] Weijing Shi and Ragunathan (Raj) Rajkumar. "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *CoRR* abs/1706.02413 (2017). arXiv: 1706.02413. URL: http://arxiv.org/abs/1706.02413.

[9] Dario Rethage, Johanna Wald, Jürgen Sturm, et al. "Fully-Convolutional Point Networks for Large-Scale Point Clouds". In: *CoRR* abs/1808.06840 (2018). arXiv: 1808.06840. URL: http://arxiv.org/abs/1808.06840.

[10] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[11] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, et al. "Frustum PointNets for 3D Object Detection from RGB-D Data". In: *CoRR* abs/1711.08488 (2017). arXiv: 1711.08488. URL: http://arxiv.org/abs/1711.08488.

[12] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[13] Charles R Qi, Or Litany, Kaiming He, et al. "Deep Hough Voting for 3D Object Detection in Point Clouds". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[14] Shuran Song and Jianxiong Xiao. "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images". In: *CoRR* abs/1511.02300 (2015). arXiv: 1511.02300. URL: http://arxiv.org/abs/1511.02300.

[15] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[16] Hongwu Kuang, Bei Wang, Jianping An, et al. "Voxel-FPN: Multi-Scale Voxel Feature Aggregation for 3D Object Detection from LIDAR Point Clouds". In: *Sensors* 20.3 (2020). ISSN: 1424-8220. DOI: 10.3390/s20030704. URL: https://www.mdpi.com/1424-8220/20/3/704.

[17] Benjamin Graham. "Spatially-sparse convolutional neural networks". In: *CoRR* abs/1409.6070 (2014). arXiv: 1409.6070. URL: http://arxiv.org/abs/1409.6070.

[18] Ben Graham. "Sparse 3D convolutional neural networks". In: *CoRR* abs/1505.02890 (2015). arXiv: 1505.02890. URL: http://arxiv.org/abs/1505.02890.

[19] Benjamin Graham and Laurens van der Maaten. "Submanifold Sparse Convolutional Networks". In: *CoRR* abs/1706.01307 (2017). arXiv: 1706.01307. URL: http://arxiv.org/abs/1706.01307.

[20] Yan Yan, Yuxing Mao, and Bo Li. "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103337. URL: https://www.mdpi.com/1424-8220/18/10/3337.

[21] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks". In: *CoRR* abs/1904.08755 (2019). arXiv: 1904.08755. URL: http://arxiv.org/abs/1904.08755.

[22] JunYoung Gwak, Christopher B Choy, and Silvio Savarese. "Generative Sparse Detection Networks for 3D Single-shot Object Detection". In: *European conference on computer vision*. 2020.

[23] Yilun Chen, Shu Liu, Xiaoyong Shen, et al. "Fast Point R-CNN". In: *CoRR* abs/1908.02990 (2019). arXiv: 1908.02990. URL: http://arxiv.org/abs/1908.02990.

[24] Shaoshuai Shi, Chaoxu Guo, Li Jiang, et al. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *CoRR* abs/1912.13192 (2019). arXiv: 1912.13192. URL: http://arxiv.org/abs/1912.13192.

[25] Shaoshuai Shi, Li Jiang, Jiajun Deng, et al. *PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection*. 2021. arXiv: 2102.00463 [cs.CV].

[26] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[27] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2446–2454.

[28] Angela Dai, Angel X. Chang, Manolis Savva, et al. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[29] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. "SUN RGB-D: A RGB-D scene understanding benchmark suite". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.

[30] Alex H. Lang, Sourabh Vora, Holger Caesar, et al. "PointPillars: Fast Encoders for Object Detection From Point Clouds". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 12689–12697.

[31] Charles R Qi, Xinlei Chen, Or Litany, et al. "Imvotenet: Boosting 3d object detection in point clouds with image votes". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, et al. "SSD: Single Shot MultiBox Detector". In: *ECCV*. 2016.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[34] Shaoqing Ren, Kaiming He, Ross B. Girshick, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[35] M. Everingham, L. Van Gool, C. K. I. Williams, et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. 2012.

[36] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.

[37] Shaoshuai Shi, Zhe Wang, Jianping Shi, et al. "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[39] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, et al. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[40] Ilya Loshchilov and Frank Hutter. "Fixing Weight Decay Regularization in Adam". In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[41] Leslie N. Smith and Nicholay Topin. "Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates". In: *CoRR* abs/1708.07120 (2017). arXiv: 1708.07120. URL: http://arxiv.org/abs/1708.07120.

[42] Guojun Wang, Bin Tian, Yunfeng Ai, et al. "CenterNet3D: An Anchor free Object Detector for Autonomous Driving". In: *CoRR* abs/2007.07214 (2020). arXiv: 2007.07214. URL: https://arxiv.org/abs/2007.07214.

[43] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, et al. "AFDet: Anchor Free One Stage 3D Object Detection". In: *CoRR* abs/2006.12671 (2020). arXiv: 2006.12671. URL: https://arxiv.org/abs/2006.12671.

[44] Qi Chen, Lin Sun, Zhixin Wang, et al. "Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots". In: *CoRR* abs/1912.12791 (2019). arXiv: 1912.12791. URL: http://arxiv.org/abs/1912.12791.

[45] Ze Liu, Zheng Zhang, Yue Cao, et al. “Group-Free 3D Object Detection via Transformers”. In: *CoRR* abs/2104.00678 (2021). arXiv: 2104.00678. URL: https://arxiv.org/abs/2104.00678.

[46] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, et al. “End-to-End Object Detection with Transformers”. In: *CoRR* abs/2005.12872 (2020). arXiv: 2005.12872. URL: https://arxiv.org/abs/2005.12872.