SegMask for Frustum PointPillar: A Multi-Sensor Approach for 3D Object Detection

Vignesh Ravikumar[1]
ravikumar.vi@northeastern.edu

Keshav Bharadwaj Vaidyanathan[1]
bharadwajvaidyanat.k@northeastern.edu

Vanshika Jain[1]
jain.van@northeastern.edu

[1]**Northeastern University, Boston, MA**

December 20, 2023

Abstract

2D image recognition tasks like object detection and instance segmentation, have made significant strides lately but beyond obtaining 2D bounding boxes or pixel masks, 3D comprehension is avidly sought in many major applications including autonomous driving and augmented reality. A growing amount of 3D data is being gathered and processed as a result of the widespread use of 3D sensors on mobile devices and autonomous cars. In light of this, there is an increasing demand for 3D object detection models that classify accurately and estimate the oriented 3D bounding boxes of actual objects using multi-sensor data. This work leverages a multi-stage sensor fusion concept that develops on top of Frustum PointPillar [13] as a baseline by leveraging robust 2D object detectors as priors to build a 3D viewing frustum that predicts a 3D bounding box of the object from the points in the frustum. We propose to modify the Frustum-PointPillar architecture to make it an end-to-end network by adding the 2D detection model, YOLOv7 [10], as part of the learning pipeline. This work also replaces the object masking technique present in [13] with a semantic segmented mask which improves the Average Precision by 3%. All models are tested and evaluated with the baseline on the KITTI 3D detection benchmark involving the camera and Velodyne point clouds. Code is available at https://github.com/vigneshr2306/SegMask-Frustum-PointPillar

1. Introduction

Recent years have seen significant advancements in 2D object recognition tasks like instance segmentation [6] and object detection [7]. With the steady rise in 3D data analysis in many applications including autonomous driving and augmented reality, information beyond just 2D bounding boxes or pixel masks is more favorable. In this work, we explore 3D object detection, one of the most significant 3D perception tasks that classify the object into categories and generate the 3D orientation of bounding boxes of real-world objects using both 2D and 3D sensor data.

The majority of previous works typically represent 3D sensor data as point clouds that are projected into images [8] or quantized into volumetric grids [9] and then fed into convolutional networks. However, this data representation conversion could mask the data's inherent 3D patterns and invariances. A few studies recently suggested processing point clouds directly rather than converting them to other formats. PointNet [2], a new class of deep learning architectures, has demonstrated higher performance and efficiency in 3D understanding tasks like object classification and semantic segmentation. It remains unclear how this architecture may be used for instance-level 3D object recognition, despite the fact that PointNets are capable of classifying an entire point cloud or predicting a semantic class. In order to achieve this, we must successfully propose potential locations for 3D objects in a 3D space, which is a significant problem.

The problem of encoding a point cloud into a format appropriate for a downstream detection pipeline is still an open research topic. Recent studies suggest two types of encoders; fixed encoders tend to be fast but sacrifice accuracy, while encoders that are learned from data are more accurate, but slower. PointPillars [14] is a novel encoder that utilizes PointNets to learn a representation of point clouds organized in vertical columns named pillars to predict 3D-oriented boxes for objects. There are several advantages of this approach. First, by learning features instead of relying on fixed encoders, PointPillars can lever-

age the full feature map given by point clouds. Further, by operating on pillars instead of voxels, hand-tuning of the binning is removed.

Although lidar data is good at the detection of large objects like cars and vans it suffers at localizing smaller objects like pedestrians in 3D point clouds due to their point perturbations and the absence of distinctive geometric structures, making them difficult to distinguish from other objects in the environment. To deal with this difficulty, leveraging multi-sensor data is a promising option. In this work, we follow a multi-sensor approach with RGB-D data for 3D object detection based on Frustum-PointPillars [13], a novel architecture for 3D object detection in point clouds. F-PointPillars leverages 2D detections in RGB images, to reduce search space in 3D and remove the sole reliability on point cloud features. They also develop a novel probability masking approach to further improve the localization of objects. Although, [1] [13] have proposed a frustum-based approach, they assume the 2D locations to be given. Hence, in this paper, we provide a solution using YOLOv7 that previous works lack. Additionally, we propose a semantic segmented mask that performs better than their novel Gaussian mask.

2. Related work

In this section, we first discuss various methods applied to point cloud dataset, and then we mention the works about multi-stage methods.

a. Approaches on Point Cloud dataset:

3D Object Detection from RGB-D Data: Researchers have used a variety of RGB-D representation techniques to tackle the 3D detection problem.

Front-view image-based approaches: [3] infer 3D bounding boxes using monocular RGB images, shape priors, or occlusion patterns. [5] use CNNs to locate objects in a 2D picture and encode depth data as 2D maps. In contrast, we deploy sophisticated 3D deep networks that can better utilize 3D geometry and represent depth as a point cloud.

Methods based on bird's eye view: MV3D [8] trains a region proposal network(RPN) for 3D bounding box proposal and projects a LiDAR point cloud to a bird's eye perspective. However, the approach struggles to detect small items like walkers and cyclists, and it is difficult to adapt to scenarios with many objects arranged vertically.

Deep Learning on Point Clouds: Prior to feature learning, the majority of existing works convert point clouds to images or volumetric forms. [11] generalize image CNNs to 3D CNNs and voxelized point clouds into volumetric grids. [12] create 3D CNN or neural network designs that are more effective by making use of point cloud sparsity. These CNN-based techniques still demand the quantization of point clouds with a specific voxel resolution, though. Our work investigates how to expand the architecture for the aim of 3D object recognition, even though PointNets have already been used for single object classification and semantic segmentation.

3D Instance Segmentation: A 2D image and its matching 3D frustum can be used to determine an object's 3D location using a number of different techniques. One simple way [1] is to use 2D CNNs to directly regress the locations of 3D objects from a depth map. However, this issue is challenging since natural environments frequently contain occluding items and background clutter, which can seriously interfere with the 3D localization task. Segmentation in a 3D point cloud is considerably more natural and simple than it is in photographs, because pixels from far-off objects may be close to one another due to the way that things are naturally separated in real space. Similar to the scenario in 2D instance segmentation, object points in one frustum may get cluttered or occlude points in another, depending on the position of the frustum.

b. Multi-stage methods:

Frustum PointNet: In multi-stage methods, two independent networks, one for each modality, are stacked together. The output of the first network (Stage-I) constitutes the input to the second network (Stage-II). The key idea is to use different sensor modalities such as RGB cameras for 2D detection and LiDAR for 3D localization according to their strength to perform a particular task. Following [1], we choose to use the multi-stage method of sensor fusion to develop our 3D object detection system as they have shown high accuracy for non-uniform objects such as pedestrian and cyclist detection. Given 2D region proposals in RGB images, [1] first finds local points corresponding to pixels inside the 2D regions and then uses the PointNet architecture to segment these local points into foreground and background. F-PointNet is not an end-to-end learning method and the estimation of 3D bounding boxes relies on fewer foreground points. Their method, without sensor fusion or multi-view aggregation, outperforms those methods by large margins on all categories and data subsets for 3D bounding box IoU threshold is 70% for cars and 50% for pedestrians and cyclists with AP 80.62, 50.88, and 69.36 respectively for an easy difficulty level test set.

Frustum PointPillar: Their design approach uses both RBG and LiDAR data for 3D detection and it consists of 4 main stages as shown in Fig.1

(1) Frustum proposal: With a given 2D bounding box information, a frustum is extruded and filtered out all the points

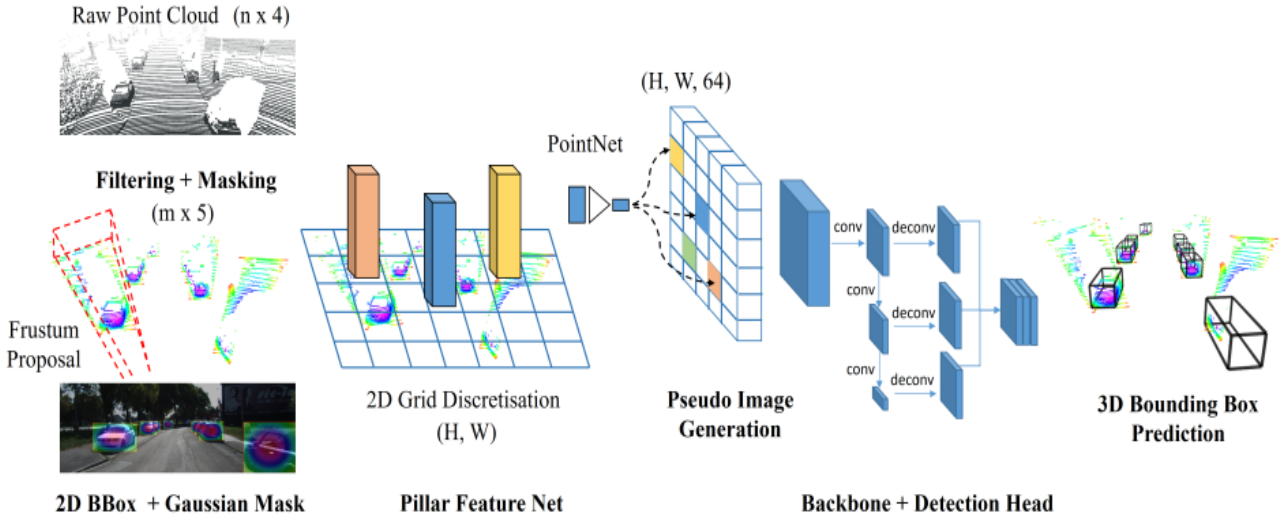

Figure 1. Frustum-PointPillars architecture referenced from [13]

outside of the 2D bounding box, thus, reducing 3D search space for the object.
(2) Point Cloud masking: They devise a simpler and faster approach to masking objects is using a probability mask that defines the likelihood of the points belonging to the object as a Gaussian function:

$$L(\overline{x}, \overline{y}) = exp\left(-\frac{(\overline{x} - \overline{x}_0)^2}{2w^2} - \frac{(\overline{y} - \overline{y}_0)^2}{h^2}\right) \quad (1)$$

where $\overline{x}$, $\overline{y}$ are point cloud projections on the image plane, $\overline{x}_0$, $\overline{y}_0$ are the center coordinates and w, h are the width and height of the 2D bounding box.
(3) Pillar feature encoding network converts a point cloud to a sparse pseudo image. We discuss this encoder in detail in the later sections of this paper.
(4) Finally, they use a backbone to process the pseudo-image and produce a high-level representation and a detection head to regress the position, orientation, and size of 3D bounding boxes.

The performance is compared with approaches using different modalities such as F-PointNet [1] and F-ConvNet [15] that use RGB and LiDAR with multi-stage design architecture. Their method outperforms the mentioned approaches for 3D detection at the hard difficulty level due to the masking of the point cloud as it provides the network with more information about the location of objects in 3D space. Thus, filtering point clouds with the help of 2D region proposals significantly reduce the number of non-empty cells in the pseudo-image. Our method develops on this approach and it is trainable end-to-end unlike previous works since they solely rely on given 2D detections.

3. Problem Definition

3D object detection and classification majorly depend on 2D detectors for 2D bounding boxes and the 3D representation of the objects to accurately classify the objects. The point cloud data structure is a significant type of geometric data and given its irregular format, most researchers transform such data into regular 3D voxel grids. This, however, renders data unnecessarily voluminous, and to tackle this issue, we use PointNet [2] which is a type of neural network that directly consumes point clouds with the ability to permute the invariance of points. Another key challenge of using point cloud data is how to efficiently localize objects of large-scale scenes based on 3D region proposals, and we use the Frustum Proposal concept from [1] to achieve higher efficiency and recall for even small objects and to precisely estimate 3D bounding boxes under strong occlusion. The above-mentioned work sets our baseline for this paper and thus, further developments are mentioned in the later sections of this paper.

3.1. Dataset

All experiments use the KITTI object detection benchmark dataset, which contains a variety of vision tasks developed on an autonomous driving platform and consists of samples from both lidar point clouds and images. This dataset includes the bounding boxes and monocular images

from the object detection dataset and we experimented our model for the car class. Previous works [13] and [14] have trained only on lidar point clouds but we train our model on both lidar and images. The samples are originally divided into 7481 training samples with 3D bounding boxes labeled and 7580 testing samples.

Type	Directory	Description
Training (7418)	Images	Car, Pedestrian, Cyclist
	Calibration	Cam, Cam-to-GPS/ Velodyne
	Velodyne	Point Cloud data
Testing (7580)	Velodyne reduced	Point Pillars stacked

Table 1. KITTI Dataset Preparation

3.2. Method

For complicated scenarios or real-time applications like autonomous driving, the computing complexity of 3D search often climbs cubically with respect to resolution. So for our methodology, we will take advantage of advanced 2D object detectors like YOLOv7 [10] to minimize the search space while adhering to the dimension reduction principle. By extruding 2D bounding boxes from image detectors, we first extract the 3D bounding frustum of an object. Next, we extract features from the PointPillar architecture. Fig. 2 explains the Frustum concept.

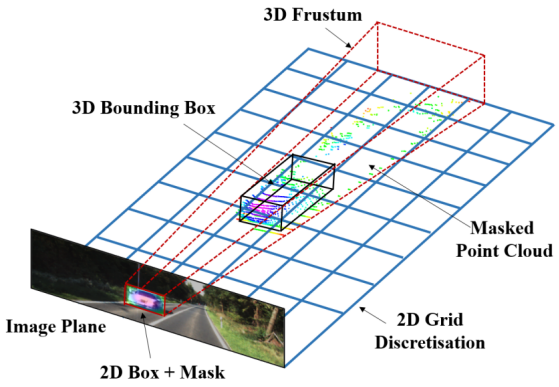

Figure 2. **Overview of Frustum Proposal**: 2D detections of objects are extruded to 3D bounding frustum. [1]

The frustum proposal leverages a mature 2D object detector to propose 2D object regions in RGB images as well as to classify objects. So, with a known camera projection matrix, a 2D box can be lifted to a frustum that defines a 3D search space. We collect all the points in frustum to form a frustum point cloud. The frustum may be oriented in any direction leading to large variations in the placement of point clouds. So we normalize the frustums by rotating them towards the center view such that the center axis of the frustum is orthogonal to the image plane and helps improve the rotation-invariance of the algorithm.

The amodal 3D bounding box is estimated by the regression network, while the segmentation network forecasts the 3D mask of the object of interest. Our approach, which lifts depth maps to 3D point clouds and processes them using 3D tools, is more 3D-centric than prior work, which considers RGB-D data as 2D maps for CNNs. This 3D-centric perspective opens up new possibilities for more efficient 3D data exploration. The geometric and topological structure of 3D space can be better utilized when learning in 3D space. We will further be referring to other research work in this domain and try to incorporate a few of those relevant techniques. We evaluate our model performance using mean Average Precision(mAP) to compare the ground-truth bounding box to the detected box for three levels of occlusion namely, easy, medium, and hard. The minimum size of the bounding box height is varied from 40px for a fully visible level to 25px for partly clouded images.

4. Proposed Methodology

Our method develops on [13] and this section explains the baseline model first and our further developments on the model.

4.1. Baseline Architecture

Our method is built on Frustum PointPillars [13] algorithm that utilizes PointPillars [14] network to detect the 2D features of the objects in the 3D space. First, we explain the architecture of [14] to grasp the idea of the Pillar feature encoder network. PointPillars network has a learnable encoder that uses PointNets to learn a representation of point clouds organized in pillars (vertical columns). It consists of three main stages:

(1) Pillar Feature Encoder Net: A feature encoder network that converts a point cloud to a sparse pseudo image using Pillar feature net that has the following processing:
(a) First, the point cloud is divided into grids in the x-y coordinates, creating a set of pillars as visualized in fig.3. Each point in the cloud, which is a 4-dimensional vector (x,y,z, reflectance), is converted to a 9-dimensional vector containing the additional information of (Xc, Yc, Zc) which is the distance from the arithmetic mean of the pillar c the point belongs to in each dimension. (Xp, Yp) is the distance of the point from the center of the pillar in the x-y coordinate

system. Hence, a point now contains the information,

$$D = [x, y, z, r, Xc, Yc, Zc, Xp, Yp] \quad (2)$$

(b) Second, the set of pillars will be mostly empty due to the sparsity of the point cloud, and the non-empty pillars will in general have few points in them. This sparsity is exploited by imposing a limit both on the number of non-empty pillars per sample (P) and on the number of points per pillar (N) to create a dense tensor of size (D, P, N). If a sample or pillar holds too much data to fit in this tensor the data is randomly sampled. Conversely, if a sample or pillar has too little data to populate the tensor, zero padding is applied. Note that D is explained in eq.2
(c) Finally, the PointNet model is applied to each point, a linear layer is followed by BatchNorm and ReLU to generate high-level features, which in this case is of dimension (C,P,N). This is followed by a max pool operation which converts this (C,P,N) dimensional tensor to a (C,P) dimensional tensor.

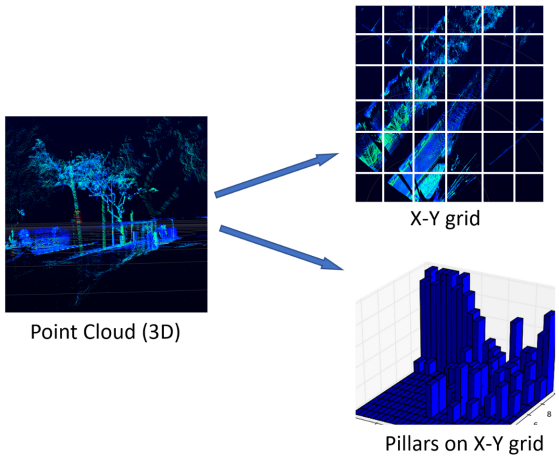

Figure 3. Pillar Feature Encoder Network of [13]

(2) 2D Backbone: A 2D convolutional backbone to process the pseudo-image into a high-level representation is applied to the encoder. The backbone constitutes sequential convolutional layers to learn features from the transformed input at different scales. The input to the RPN is the feature map provided by the Pillar feature network.

(3) 2D Detector: A detection head that detects and regresses 3D boxes with the help of the SSD network to generate 2D bounding boxes on the features generated from the backbone layer of the Point Pillars network.

[13] has 4 stages but we explain the general idea of their design in two brief concepts. First, the Frustum Proposal network is used to refine the detected objects by using the masking technique, and second, the encoder network similar to the network explained in [14] is used and given to the backbone architecture for 3D object detection. Fig.4 explains the architecture of Frustum-PointPillars.

(1) Frustum Proposal and Object Masking: With a known camera projection matrix, a 2D bounding box is extruded to a frustum that defines a 3D search space for the object. After filtering out all the points outside of the 2D bounding boxes, they perform gaussian masking which selects the region near the center of a 2D bounding box that is more likely to be occupied by the object. Similarly, the projected 3d points near the center region are also more likely to belong to the object instead of the background clutter. This likelihood value is added to point P as an additional feature vector making the input point feature C_{in} 5 dimensional (x, y, z, intensity, L). If a point is shared by multiple 2D bounding boxes, then the highest likelihood value is chosen. Let $P \in R^3$, a point in the point cloud, transformation matrix $T : R^3 \rightarrow R^2$ transforms P to its projection $\overline{P} \in R^2$ on the image plane.

(2) Backbone and Detection Head: To extract spatial features, they use two sub-network as a backbone. One performs upsampling using transposed 2D convolutions and concatenation of the top-down features. The top-down network uses a sequence of 2D convolution blocks to produce features at increasingly small spatial resolutions. For the detection head, the Single Shot Detector (SSD) is used following prior works to match the prior anchor boxes to the ground truth using 2D Intersection over Union (IoU). Thus, given a positive 2D match, the bounding box parameters are regressed, and the height and elevation become additional regression targets.

4.2. Proposed Architecture

The main contributions of our work over F-Pointpillars are as follows.

4.2.1 State-of-the-art 2D Object Detector

The major drawback of F-PointPillars is that the pipeline is not end-to-end, and the 2D bounding boxes are assumed to be given. This makes it impractical for real-time applications like autonomous driving. Therefore, we leveraged the robust state-of-the-art 2D object detector YOLOv7 for generating the 2D bounding boxes of objects which can be used as a prior for constructing the frustum proposal and detecting 3D objects from point clouds. It can be seen from Fig.5 that YOLOv7 performs more accurate detections of cars than the KITTI ground truth.

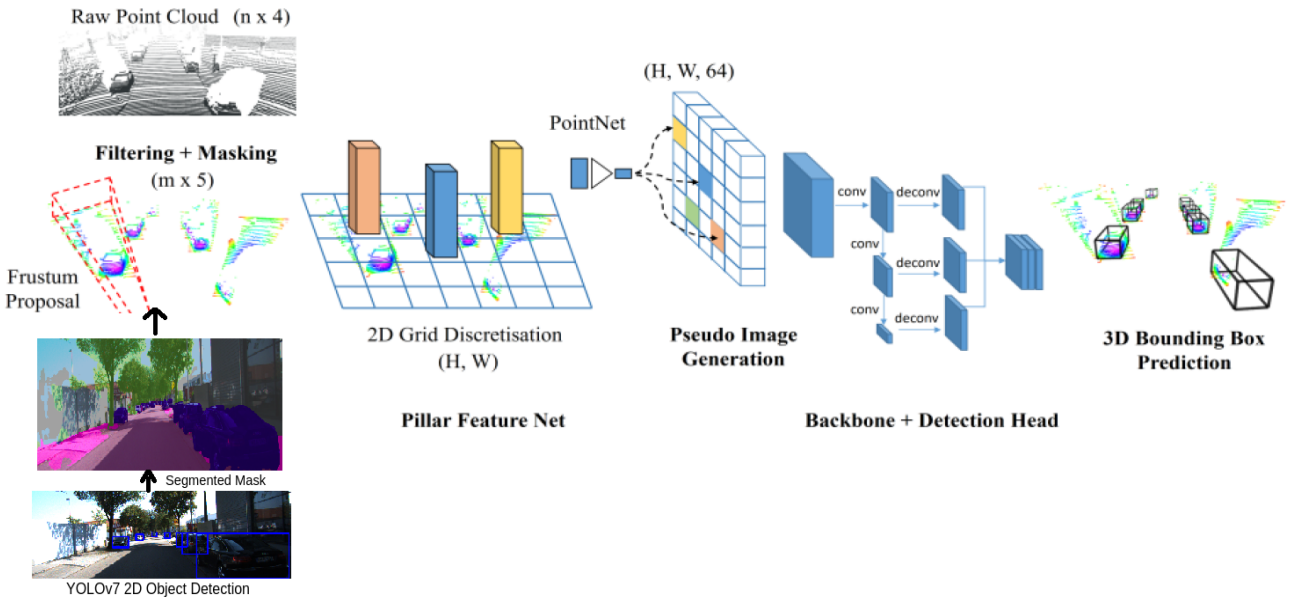

Figure 4. SegMask-Frustum PointPillars Architecture (Ours)

Figure 5. YOLOv7 detection (blue) vs Ground-Truth (Green) on KITTI

Car	**Frustum PointPillar**			**SegMask-FP (Ours)**		
	Easy	Medium	Hard	Easy	Medium	Hard
BEV (0.7)	85.26	83.41	80.12	86.43	85.12	**82.34**
3D (0.7)	84.31	75.47	75.58	86.32	75.42	**78.01**

Table 2. Evaluation Table

4.2.2 Segmentation Masking for Frustum Proposal

[13] proposes a Gaussian masking technique to mask the point clouds for better localization of objects as shown in Fig. 6(a). However, this mask naively takes the euclidian distance of the projected point clouds from the center of the object, which makes the farther points from center less

significant. Therefore, we propose a segmentation layer on top of the YOLOv7-detected bounding boxes using pre-trained PSPNet [16], which focuses on the entire object of interest, thus improving the localization of the object. Fig. 6(b) shows segmented mask of the same person.

Semantic segmentation associates a label or category with every pixel in an image. The pipeline involves PSPNet segmenting the entire image, giving us probability scores for every pixel. We iterate through every object present in the image given by YOLOv7, and extract the probability scores from the segmented image, and use these scores as mask for the 3D-to-2D projected lidar points.

5. Experiments

We trained our baseline Frustum PointPillar model and the SegMask Frustum PointPillar model on the Discovery cluster with Tesla P100 GPU. The maximum number of points per pillar (N) is kept at 100. We use Adam optimizer with an initial learning rate of $2 * 10^{-4}$ and decay the learning rate by a factor of 0.8 every 15 epochs. We use a batch size of 2 to train the model. We run the model for 1,67,000 steps. Table 3 shows the training time taken by the individual models. We use combination of losses for the training of Frustum-PointPillars and SegMask Frustum-PointPillars. loss equation 3 has two hyper parameters β_{cls} and β_{loc}. We assign value of 0.8 to both.

$$L_{total} = \beta_{cls} L_{cls} + \beta_{loc} L \beta_{loc} \tag{3}$$

Fig.7 shows the loss curves for both Frustum-PointPillar model as well as SegMask Frustum-PointPillar model. Fig.8 and Fig.9 shows the visualization of 3D object detection. Table 2 shows performance of our models.

Model	Time per 50 steps	Total train time
Frustum PointPillar	41 sec	∼38 hrs
SegMask Frustum PointPillar	58sec	∼54 hrs

Table 3. Train time

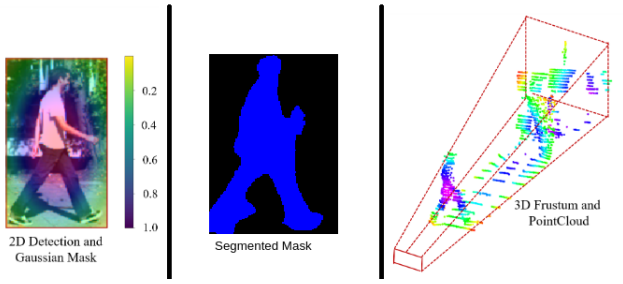

Figure 6. (a) Gaussian masking proposed by [13]; (b) Segmented Mask (Ours); (c) 3D Frustum and Point Cloud using the mask

6. Conclusions

In this work, we addressed the difficult issue of 3D object identification in point clouds in this study. We improved on Frustum-PointPillars which leverages both RGB and LiDAR data for 3D detection. We added a YOLOv7 object detector for generating a prior for the frustum proposal, which was assumed to be given in [13]. We altered the naive gaussian masking technique with a matured pre-trained PSPNet segmentation mask, which improved the car AP scores by 3%. However, the model gets bulkier with the addition of two extra models, and hence increases the computation time. The future scope of this work could be improving 2D detection in low-light conditions using data from RGB and LiDAR and avoiding outlier detections from reflections in surfaces such as glasses and windows.

Acknowledgement

This work was conducted at Northeastern University, Boston as part of the Deep Learning course. We'd like to thank Dr. Yanzhi Wang for her guidance and support throughout the course.

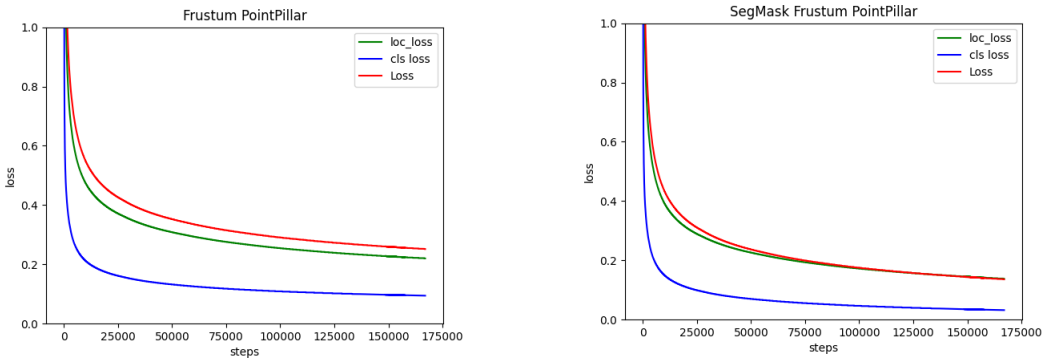

Figure 7. Loss curve for Frustum PointPillar and SegMask Frustum PointPillar

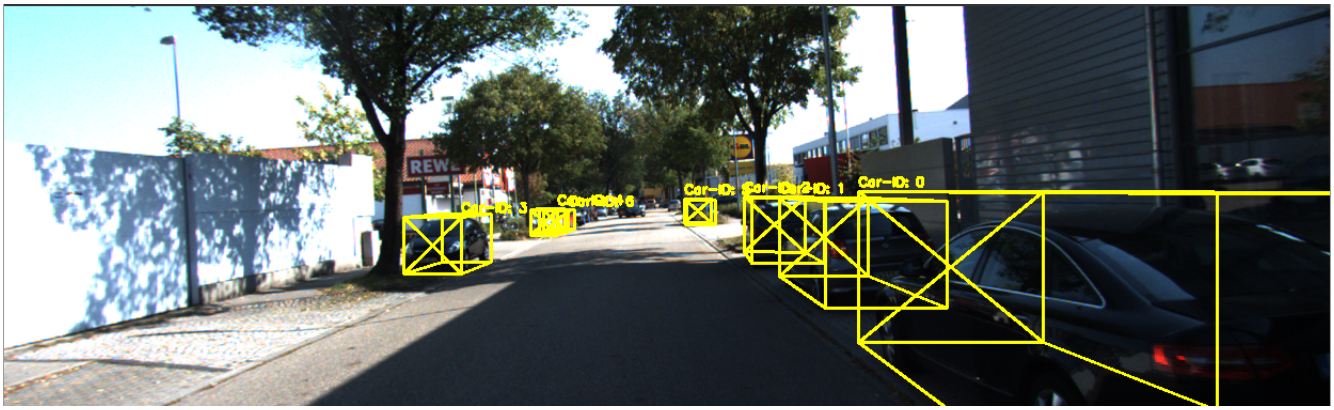

Figure 8. 3D Visualization on image predicted by our model

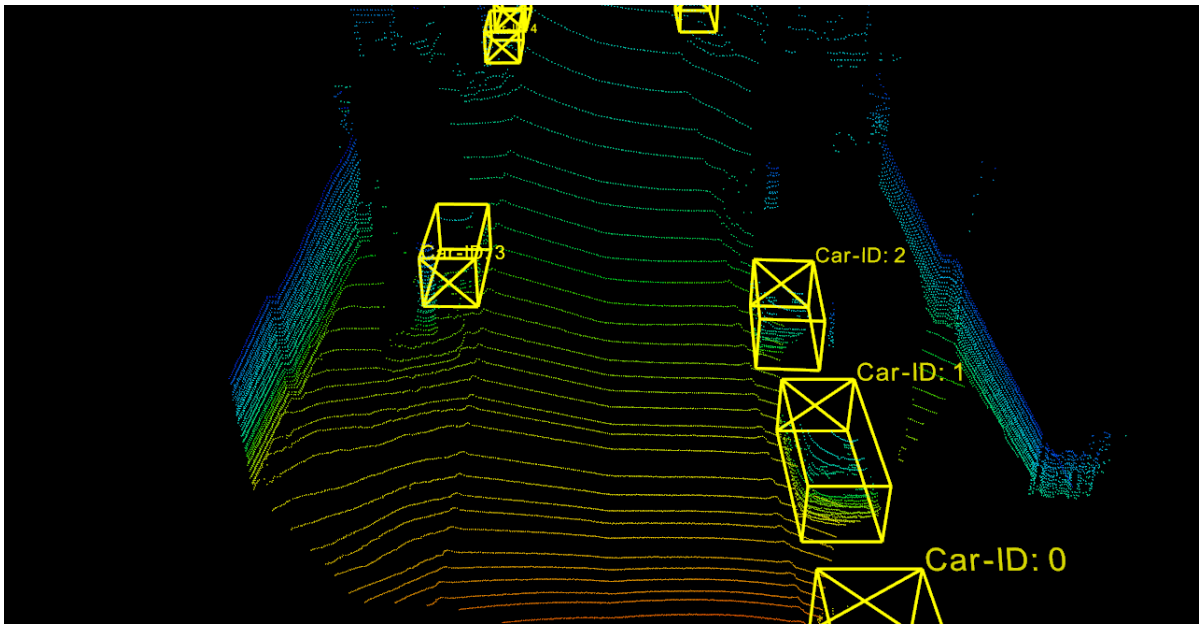

Figure 9. 3D Visualization using labels predicted by our model

References

[1] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918-927, doi: 10.1109/CVPR.2018.00102.

[2] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77-85, doi: 10.1109/CVPR.2017.16. 2, 3, 4

[3] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler and R. Urtasun, "Monocular 3D Object Detection for Autonomous Driving," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2147-2156, doi: 10.1109/CVPR.2016.236. 1, 3

[4] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3D Object Detection Network for Autonomous Driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526-6534, doi: 10.1109/CVPR.2017.691. 2

[5] B. Li, "3D fully convolutional network for vehicle detection in point cloud," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1513-1518, doi: 10.1109/IROS.2017.8205955.

[6] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322. 2

[7] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81. 1

[8] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 945-953, doi: 10.1109/ICCV.2015.114. 1

[9] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015 1, 2

[10] Wang, Chien-Yao and Bochkovskiy, Alexey and Liao, Hong-Yuan Mark. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, arXiv preprint arXiv:2207.02696, 2022. 1

[11] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 1, 4

[12] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. arXiv preprint arXiv:1605.06240, 2016. 2

[13] A. Paigwar, D. Sierra-Gonzalez, Ö. Erkent and C. Laugier, "Frustum-PointPillars: A Multi-Stage Approach for 3D Object Detection using RGB Camera and LiDAR," 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021, pp. 2926-2933, doi: 10.1109/ICCVW54120.2021.00327. 2

[14] Lang, Alex H. et al. “PointPillars: Fast Encoders for Object Detection From Point Clouds.” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018): 12689-12697. 1, 2, 3, 4, 5, 6, 7

[15] Z. Wang and K. Jia, “Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 1742–1749. 1, 4, 5

[16] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," 2017 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6230-6239, doi: 10.1109/CVPR.2017.660. 3