3D Object Detection via Point Cloud and Image for Autonomous Vehicle

Liu Yang
Department of Computer Science
Stanford University
liuyeung@stanford.edu

Abstract

This report proposes an early fusion method to integrate point cloud and camera views at point granularity for 3D object detection in autonomous vehicle scenarios. There exists work [1, 4] considers point cloud from 2D perspective, builds feature connection between the frontal view or bird-eye view (BEV) of point cloud and the camera images. However, these methods can't been extended to leverage the new techniques like 3D sparse convolution [2] and set abstraction[6]. Meanwhile, there is a surge of work [11, 8, 3] which relies only on point cloud and achieves impressive performance beyond previous work. It motivates me to find a better way to fuse the two modalities. Instead of frontal view or BEV, I think that multi-modality fusion should be applied to each point of the point cloud, which best suits for further manipulation of 3D sparse convolution and set abstraction. In this report, I design two fusion methods for 3D sparse convolution and set abstraction respectively. Experiment shows my methods can improve the performance of PV-RCNN[7], which was originally a pure point cloud solution, by a considerable margin.

1. Introduction

3D object detection is a crucial perception module for robotics and autonomous vehicles. Autonomous systems generally gather information of the environment by cameras, radar or lidar sensors, etc. The task aims to solve the position and geometry metrics such as width, height and orientation of the 3D objects like pedestrians, traffic lights, vehicles etc.

In addition to images, 3D detection can leverage point cloud detected by radar/lidar, which injects more environmental inputs at the cost of increased complexity in feature extraction. To handle point cloud, VoxelNet [13] proposes to divide a point cloud into equally spaced 3D voxels, then summarizes features of points within the same voxel as the feature representation of that specific voxel cell. Voxel grid is a very sparse 3D data structure, of which convolution can be conducted by submanifold sparse convolution network [2]. By 3D convolution, a point cloud can be encoded to bird eye view (BEV) feature-map on which 2D detectors are applied to generate 3D bounding boxes. Convolution involved methods are generally computation expensive, alternative is point based method. PointNet++[6] proposes set abstraction (SA) operation, which samples a set of points by farthest point sampling (FPS). For each sampled point, PointNet++ groups features of its neighbouring points within a proper region to form order invariant point representation. By stacking several SA operations and adjustable regions, PointNet++ builds a perception stack which can extract multi-scale feature from point cloud. Voxelization, 3D sparse convolution and set abstraction are now the basic building blocks of 3D object detection pipeline based on point cloud.

Since point cloud providing accurate metrics and robust to light conditions, most work focus on pure point cloud approaches, the effectiveness of which is supported by winners of the 3D object detection challenges held by Waymo and Motional. On the leadboards, PV-RCNN[7] and CenterPoint [12] are both pure point based methods without image. However, the success of pure point methods does not imply that image inputs are useless. Intuitively, different objects generally have different colors, thus color aware feature may positively contribute to the object detection. Specific to autonomous vehicle scenario, with images captured by cameras from different viewports, we can construct the 3D environment to some degree by geometry methods. Thus images actually preserve abundant environmental information. Inspired by this, I propose to explore effective ways to fuse point cloud and image features for 3D detection of autonomous vehicle scenario.

The challenge is how to bridge point cloud and image. If we want to keep the advantages of point based method, features from the two modalities must be aligned at point level. A straightforward method is to calibrate the cameras with respect to the point cloud generator. Then we can build the correspondence between 3D points (x, y, z) and image coordinates (u,v) by camera projection. With image coordinate

(u,v), we can retrieve image color and image feature map. My fusion method is highly depend on this 3D-2D correspondence. In section 4.1, I illustrate my multi-modality fusion method for 3D sparse convolution operation. In section 4.2, I illustrate my multi-modality fusion method for set abstraction operation.

To the best of my knowledge, no previous work tried to fuse point cloud and camera views at the granularity of point. The positive impact on performance of PV-RCNN [7] demonstrates the effectiveness of my proposed method.

2. Related Work

Methods based on point cloud and images. MV3D[1] takes the bird's eye view and front view of LIDAR point cloud as well as an image as input. It first generates 3D object proposals from bird's eye view map and project them to three views. A deep fusion network is used to combine region-wise features obtained via ROI pooling for each view. The fused features are used to jointly predict object class and do oriented 3D box regression. ContFuse[4] has two streams, namely the camera image stream and the BEV LIDAR stream. It propose a Continuous fusion layers to fuse the image features onto the BEV feature maps. These methods are generally efficient for accurate 3D proposal generation but the receptive fields are constraint by the kernel size of 2D/3D convolutions.

Methods based on pure point cloud. PointRCNN [8] generates 3D proposals directly from the whole point clouds instead of 2D images for 3D detection with point clouds only. The following work STD[10] proposes the sparse to dense strategy for better proposal refinement. VoteNet[13] proposes the hough voting strategy for better object feature grouping. These point based methods are mostly based on the PointNet[5] series, especially the set abstraction operation[6], which enables flexible fields for point cloud feature learning.

3. Problem Statement

I am experimenting my method with Waymo Open Dataset, which has 4 kinds of objects: vehicle, pedestrian, traffic sign and cyclist. For each object, the annotation includes its width, length, height and yaw angle. The task is to predict these geometry metrics by the given 3D point cloud and visual inputs. Specific to 3D detection/tracking tasks, Waymo Open Dataset provides 1000 video segments each containing roughly 200 frames, which is further divided into 768/232 train/validation splits. For each frame, the dataset provides the raw range data and images acquired by lidar and 5 cameras respectively. The 5 cameras are placed on the front left, front right, side left, side right and rear of the vehicle, each camera is calibrated to the reference system of the 3D points.

3.1. Data Preprocessing

I use the PV-RCNN[7] implementation in OpenPCDet as the baseline. PV-RCNN has its own data preprocessing which preserves 3D data of each frame and the processed data takes roughly 1TB disk space. Since I want to leverage 2D information, I have to preserve the 5 images and the 2D correspondence of the 3D point cloud, the data volume is 10 times to the PV-RCNN case for each frame. Thus I have to sample the video segment at an interval of 5 frames. I modify the data preprocessing of PV-RCNN to serve my purpose. My final dataset is 1.7TB, I have developed a Pytorch dataset module to read in both 3D cloud and visual inputs.

3.2. Evaluation

3D object detection task uses average precision (AP) and mean average precision (mAP) as the evaluation metrics. Intersection of union (IoU) between a bounding box prediction and the ground truth is used to conduct ture/false positive judgement, by which we can compute precision and recall. There are 4 classes in Waymo Open Dataset: car, traffic sign cyclist and pedestrian. For each class, we generate its precision-recall curve and compute its average precision (AP) according to Eq.(1).

$$AP = \int_0^1 p(r)dr \qquad (1)$$

Then the mAP is computed by Eq.(2).

$$mAP = \frac{AP_{car} + AP_{cyclist} + AP_{pedestrian} + AP_{sign}}{4} \qquad (2)$$

4. Technical Approach

The toolkit of Waymo Open Dataset has a module to project arbitrary 3D points into 2D image, taking noising effects such as distortion and vehicle velocity into consideration. However, the toolkit supporting Tensorflow only and involving C++ implementation, I can't make it compatible to Pytorch in such a short time. Fortunately, the toolkit produces the 2D correspondences in at most two camera views for each 3D point accompany with the cloud generation. Although I can't project any 3D point to camera views, I do can access visual information via the given 3D-2D correspondence at the very beginning of the pipeline. Actually, this limits my method on early fusion because the 3D-2D correspondence will be lost in further manipulation. I plan to maintain 3D-2D correspondence to enable fusion in intermediate process in future work, however, at this report, I stick to early fusion to validate the effectiveness of my idea.

Figure 1 illustrates the pipeline of PV-RCNN [7]. The upper part is 3D sparse convolution[2], which considers the point cloud as an integrated structure to extract multi-scale

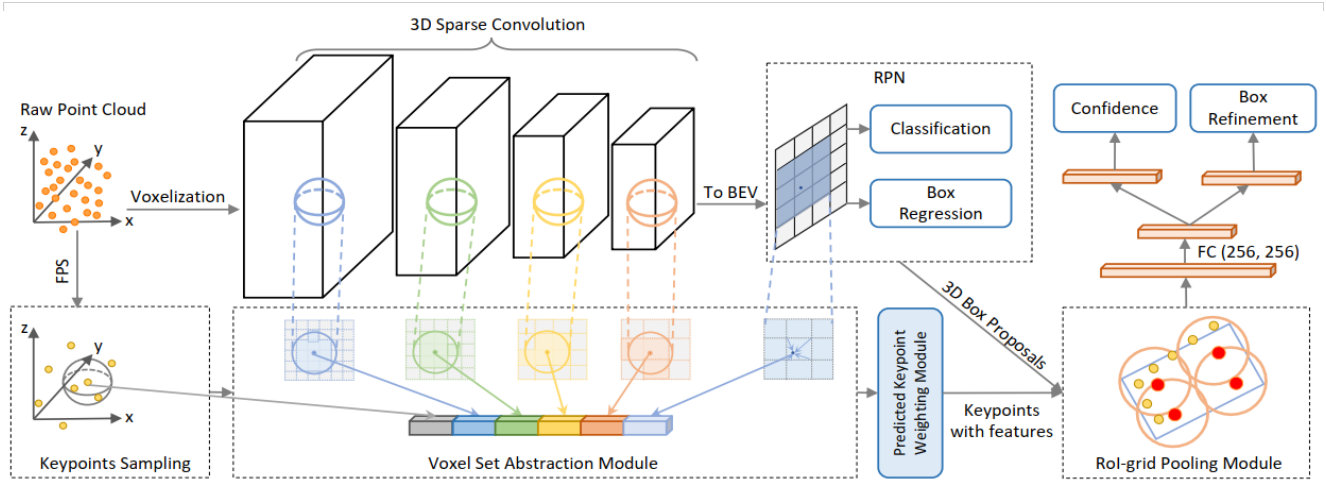

Figure 1. 3D detection pipeline of PV-RCNN

volumetric feature, like that 2D convolution dose to image. The lower part is set abstraction [6], which focus on the local structure of point cloud. PV-RCNN has an interaction between 3D convolution and set abstraction, which groups point representation in the 3D scaled convolutional features. Since my method is early fusion, I won't touch architecture of PV-RCNN deeply but focus on feature extraction stage. In section 3.1, I explain my method to inject RGB color in 3D sparse convolution. In section 3.2, I explain my method to inject visual information in set abstraction.

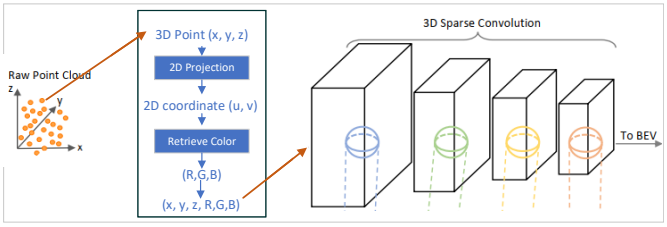

Figure 2. Injecting color in 3D sparse convolution

4.1. Visual Aware Point Cloud in 3D Sparse Convolution Stage

After voxelization, point cloud becomes a 3D grid. Each cell has a feature vector obtained by aggregating features of points locating in that cell in some way. Like 2D convolution, 3D convolution has a kernel applied to each cell to aggregate local information and stepped in a specific stride similar as 2D convolution.

My early fusion method doesn't modify 3D convolution components but modify the representation of point in point cloud to build visual aware cloud as input of 3D convolution. Specifically, as illustrated in Figure 2, a 3D point $p = (x, y, z)$ has a feature vector of 5 entries: $(x, y, z, intensity, elongation)$. Remember that Waymo toolkit provides 2D coordinates of p in at most 2 camera views V_p^1, V_p^2, by which I can retrieve the colors of the 3D point. Formally, I define the RGB feature c_p as below:

$$t_i = \begin{cases} V_p^i(p), & p\ is\ visible\ in\ V_p^i; \\ [0,0,0], & Otherwise; \end{cases}, i \in \{1,2\} \quad (3)$$

$$c_p = t_1 + t_2 \quad (4)$$

Eq.(4) uses addition to avoid the impact of the image order, which is inspired by PointNet [5].

The color augmented feature f_p is obtained by simply appending c_p to the raw point feature, that is:

$$f_p = (x, y, z, intensity, elongation, *c_p) \quad (5)$$

The 3D point cloud with the color augmented feature is then handed over to 3D convolution operation.

4.2. Visual Aware Point Cloud in Set Abstraction

Unlike 3D convolution, in which the point cloud is considered as a whole, set abstraction firstly samples a small seed set of points via FPS (furthest point sampling) as the representation of the point cloud, then gathers points in neighbor region of different size around each seed point to build multi-scale feature. Intuitively, the first sampling stage will drop many points which incurs spacial information loss. I want to complement the information more from visual side. Thus instead of RGB color, I try to augment the point representation by feature maps of Efficientnet[9].

As illustrated in Figure 3, I use EfficientNet-b5 to extract visual features for the 5 camera views. Given a image

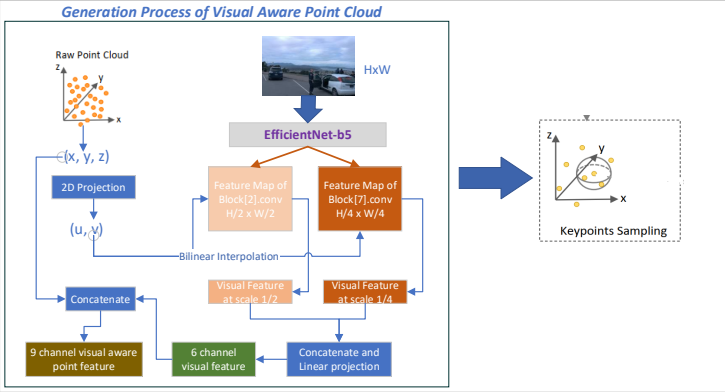

Figure 3. Injecting visual feature in set abstraction

with width W and height H, I use the convolutional feature-maps of the last block with size $\frac{H}{2} \times \frac{W}{2}$ and the last block with size $\frac{H}{4} \times \frac{W}{4}$. I use only 2 feature-maps here in order to save computation and graphic memory usage. With the 3D-2D correspondence, given a specific 3D point, I can retrieve its corresponding feature in Efficientnet-b5 feature-maps of the visible views.

Formally, for point p, following the terminology defined in section 4.1, the augmented visual feature is obtained as below:

$$e_p^{ij} = \begin{cases} EfficientNet - b5[j][V_p^i], & p\ is\ visible\ in\ V_p^i \\ \vec{0}, & Otherwise \end{cases} \quad (6)$$

, where $i \in \{1, 2\}$ is the index of the potential visible camera view and $j \in \{2, 7\}$ is the block index of the convolutional feature-maps in Efficientnet-b5.

By concatenating $[e_p^{i2}, e_p^{i7}]$, I obtain the visual feature of the i^{th} potential view. As Eq.5, define E_p by:

$$E_p = [e_p^{12}, e_p^{17}] + [e_p^{22}, e_p^{27}] \quad (7)$$

To avoid E_p dominating the point feature, I project E_p to be a 6 entry vector and concatenate it with the raw 5 entry point feature to produce the final 11 entry vector S_p as the efficientNet augmented feature.

The 3D point cloud with EfficientNet augmented feature is then handed over to set abstraction operation.

4.3. Data Augmentation

The implementation of PV-RCNN [7] in OpenPCDet conducts data augmentation in various ways. For example, flipping flips the point cloud randomly along x, y, z axis; rotation rotates the point cloud about a random rotation axis. I keep an augmentation method as long as it won't change the predefined 3D/2D coordinates correspondence. In addition, I don't apply any data augmentation to the 2D images because it would be hard to maintain the 3D/2D correspondence otherwise.

4.4. Training Loss

My early fusion methods don't change components of PV-RCNN [7], I just reuse its training loss, which I refer to its paper for reference reading.

5. Experiments

The purpose of the experiment is to find whether my early fusion methods have positive impact on the performance of PV-RCNN. So I don't pay much effort on reproducing the performance declared in PV-RCNN paper, but focus on the performance difference between the configurations which turns on and turns off the visual fusion module.

5.1. Experiment Setup

Dataset. I have create two datasets, *dataset-mini* is a small dataset serves the quick debugging and validation purpose. It contains only 200 vehicle instances and is split to 160/40 train/validation sets. The *dataset-full* covers the whole 1000 videos, has 4587 instances of cars, pedestrians and cyclists. The *dataset-full* is split to 3188/1339 train/validation sets. I presented the primary result in milestone report using *dataset-mini*, at that time, the result looked negative. I keep on reporting result on *data-mini* in final report because it reflects the debugging work and the progress.

Training Details. I train the network from scratch in an end-to-end manner with the ADAM optimizer. The batch size is set to 2, learning rate 0.01 for 30 epochs on 1 Titan RTX GPU with 24G graphic memory. The cosine annealing learning rate strategy is adopted for the learning rate decay. It takes around 1 hours for the training on *dataset-mini* to finish and 40 hours for that on *dataset-full*.

5.2. Performance on dataset-mini

On the *dataset-mini*, I report 2 metric, the average precision (AP) of vehicle and the Recall at IoU 0.7. The latter means the true-positive criteria is the IoU between predicted bounding box and the ground truth should larger than 0.7. I evaluate the last 10 epochs of the training checkpoints on the validation split.

Figure 4 illustrates the comparison of AP and Recall between PV-RCNN with and without visual information at milestone stage. The model was not fully trained since the AP and Recall of both configurations were very low. This is reasonable because the number of updates is too small compare to the parameters' volume. At that time, the model **without** visual information (yellow curve) performed much better than that **with** visual information (blue curve).

After carefully debugging and reviewing the code, I find the root cause is two folds. 1. I didn't scale the range of

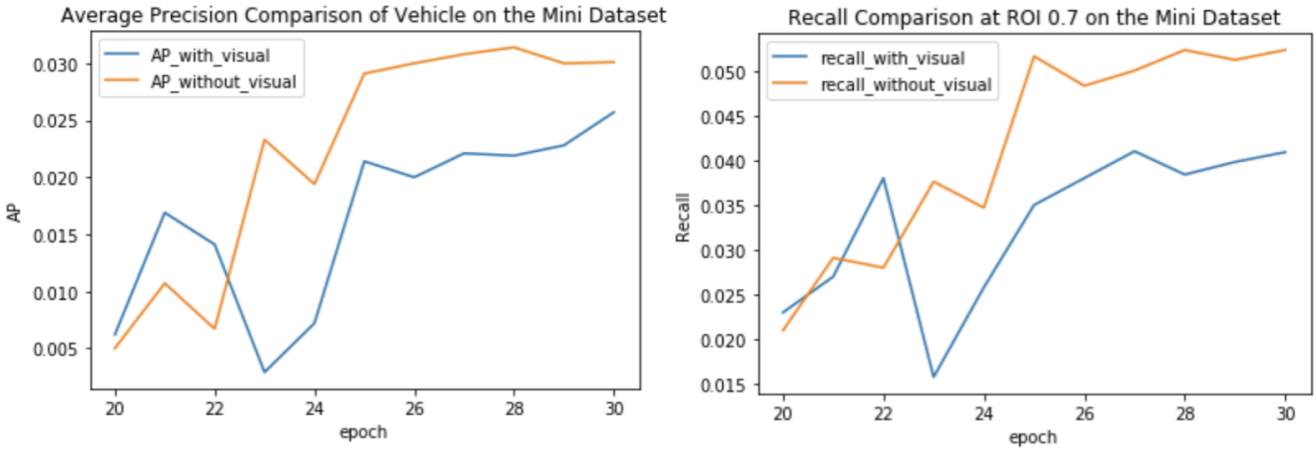

Figure 4. Comparison of PV-RCNN with/without visual information at **milestone stage**

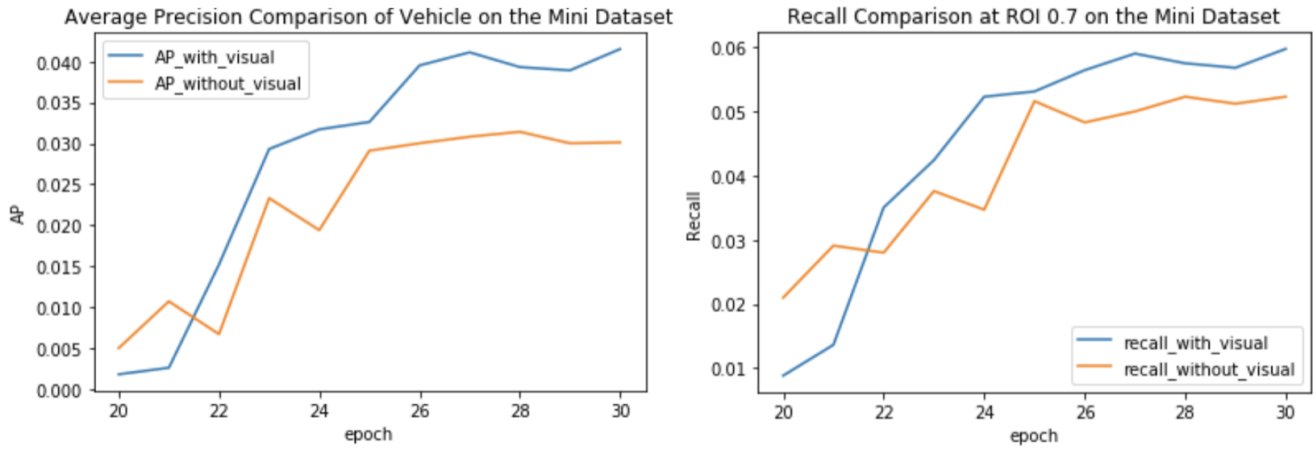

Figure 5. Comparison of PV-RCNN with/without visual information **NOW**

pixel value from [0, 255] to [0, 1]. 2. I didn't normalize the input image with mean and variance of ImageNet which required by processing of EfficentNet. As illustrated in Figure 5, after fixing this bug, the model **with** visual information (blue curve) performs better than that **without** visual information (yellow curve) on the *dataset-mini*, which is expected.

5.3. Performance on dataset-full

On the *dataset-full*, I report average precision (AP) for 3 classes: car, pedestrian and cyclist. The recalls are reported at IoU 0.3, IoU 0.5 and IoU 0.7. I evaluate the last 10 epochs of training checkpoints on validation split to obtain the statistics.

In Figure 6, we see the model is well trained on *dataset-full* since the average precision has reasonable high value. In all the subplots, the blue curve represents the performance of model **with** visual information, the yellow curve represents the performance of model **without** visual information. On car prediction, the two models draw similar performance curve, especially at the end of the training. Both models hit 70% AP on car prediction. On the pedestrian prediction, both models achieve performance above 50%, however, the model **with** visual information consistently outperforms the one **without** visual information by a large margin ($> 2\%$). On the cyclist prediction, both models don't perform well, but we identify a huge performance gap. The peak performance of model **without** visual information is around 8%, compare to the 12% of model **with** visual information, the relative performance gap is $\frac{12\%-8\%}{8\%} = 50\%$.

To explain the observation in Figure 6, note that the color variance of car is not as significant as that of human. Car is generally has a dominant color: white, black, golden etc. However, for a person, there are var-

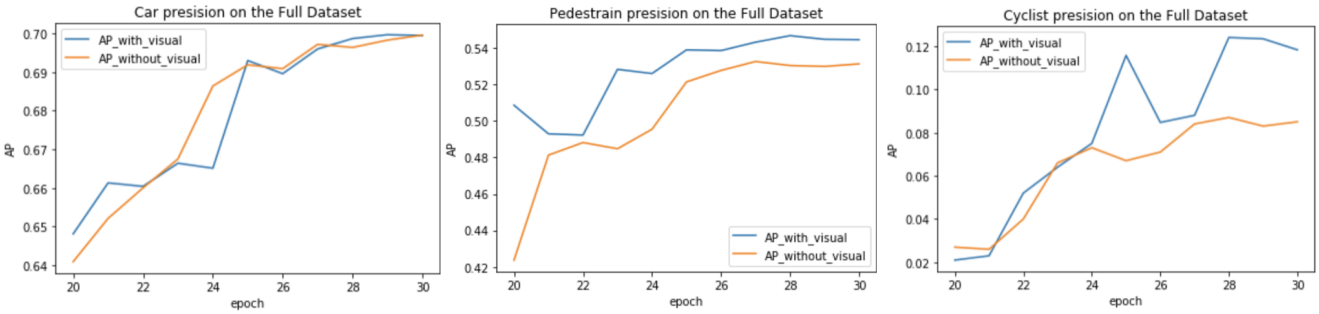

Figure 6. Precision Comparison of PV-RCNN with/without visual information on *dataset-full*

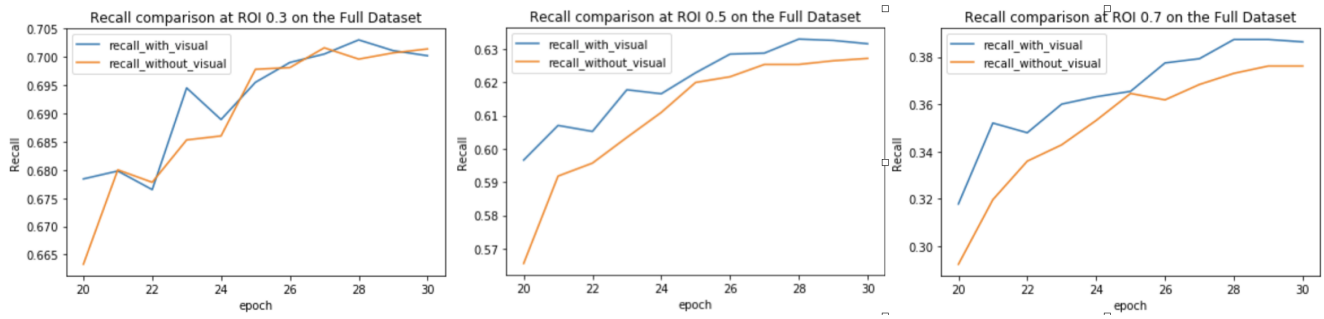

Figure 7. Recall Comparison of PV-RCNN with/without visual information on *dataset-full*

ious colors can be found on his/her appearance. He/she may have white/black/blond hair; black/white/yellow skin; red/black/varicoloured clothes etc. Figure 6 actually demonstrates that my method has no harm to the capability to detect car while can greatly enhance the capability to detect human related objects with high color variance, which is expected.

In Figure 7, we see that the two models have comparable recall at IoU 0.3. However, at IoU 0.5, the recall of model **with** visual information is consistently larger than that of the model **without** visual information through out the 10 test epochs. The gap tends to be wider at IoU 0.7. The true positive-criteria becomes more strict as the IoU threshold increasing, thus Figure 7 demonstrates that my method can enhance prediction accuracy and is more robust to strict criteria.

I select statistic of the 29^{th} epoch to do quantitative comparison. Table 1 lists the comparison on the average precision. We see that my method (Point cloud + Image) outperforms the pure point cloud configuration on detection of all the 3 classes.

Table 2 lists the quantitative comparison on the recall. It can be identified that my method outperforms the pure point cloud configuration on recall at different IoU criteria as well.

Based on analysis above, the experiments reveal the fact that my early fusion method is effective to the 3D object detection task.

Model	Car	Pedestrian	Cyclist
Point cloud only	69.83%	53.00%	8.30%
Point cloud + Image	**69.97%**	**54.49%**	**12.34%**

Table 1. Average precision comparison between models with and without visual information at epoch 29

Model	Recall 0.3	Recall 0.5	Recall 0.7
Point cloud only	70.07%	62.64%	37.62%
Point cloud + Image	**70.11%**	**63.25%**	**38.74%**

Table 2. Recall comparison between models with and without visual information at epoch 29

6. Conclusion

In this report, I explain the motivation and techniques of my early fusion method for point cloud and camera images in details. Experiment shows my method is effective and improves the performance of a state of the art network PV-RCNN. In future work, I shall enable free point projection rather than limited by 3D-2D correspondence initialized by Waymo toolkit. I shall keep on exploring the possibility

of fusion at the granularity of point in deeper stage of the 3D object detection pipeline. Code of my project is available at https://github.com/xiaoxiaoheimei/cs231A-winter2021-project

7. Acknowledgement

Many thanks to CS231A staff to offer such an amazing course, I really learned a lot from the lectures, the homework(though they made me suffer sometimes..), the midterm(though I didn't do very well..), and the project(though I don't have enough time to train..). Wish you happy every day!

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016. 1, 2

[2] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *CoRR*, abs/1706.01307, 2017. 1, 2

[3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CoRR*, abs/1812.05784, 2018. 1

[4] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection, 2020. 1, 2

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 2, 3

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. 1, 2, 3

[7] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 1, 2, 4

[8] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *CoRR*, abs/1812.04244, 2018. 1, 2

[9] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 3

[10] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), Oct. 2019. 2

[11] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. 1

[12] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020. 1

[13] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499. IEEE Computer Society, 2018. 1, 2