3D Object Detection with Enriched Point Cloud

Dazhi Cheng

CMU-CS-20-137

December 2020

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Kris M. Kitani, Chair
Matthew P. O'Toole

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Keywords: Machine Learning, Deep Learning, Computer Vision, 3D Object Detection

To cars that will eventually drive themselves.

Abstract

State-of-the-art 3D object detectors are designed based on datasets with sparse and single-echo point cloud information. However, with recent advancements in LiDAR sensing, it is of great significance that we understand how richer point cloud information can be leveraged to boost performance of 3D object detectors. In this thesis, we push the limit of 3D object detection by enriching point cloud in three ways: capturing multiple reflection points in each beam instead of one; capturing an additional ambient value (IR sunlight reflected) corresponding to each point; and increasing point cloud density. Specifically, based on point cloud with ambient information collected by a prototype LiDAR, we propose a multi-view segmentation-detection fusion framework that enhances metric-preserving yet sparse voxel feature learning by dense observations from perspective view. The proposed framework is shown to noticeably improve pedestrian detection accuracy. Also, based on multi-echo point cloud data collected from a prototype single-photon LiDAR with increased fill factor, we significantly boost performance of state-of-the-art detectors by introducing multiple points per beam. Lastly, by leveraging a synthetic dataset, we observe notable improvements in detection accuracy when point cloud density is increased. Our results show that with proper model design, 3D object detection will benefit greatly from enriched point cloud information, which calls for new benchmarks based on more advanced LiDAR sensors.

Acknowledgments

My deepest gratitude goes first to my advisor, Professor Kris M. Kitani, for offering me an opportunity to conduct research in the area of 3D perception, and for his invaluable guidance and support throughout the course. It has truly been an honor to work with Professor Kitani.

I am also extremely grateful to Professor Matthew P. O'Toole, for providing insightful understandings regarding remote sensing hardware, and for being on my thesis committee. It has been a privilege to work with Professor O'Toole.

My appreciation extends to my laboratory colleagues, Xinshuo Weng, Yunze Man, Jinhyung (David) Park, Haoshuo Huang, and Han Deng. Their generous assistance, as well as inspiring thoughts, have significantly enriched this thesis. I consider myself fortunate to work alongside my colleagues.

Lastly, I am indebted to my parents, my brother, and my girlfriend, for their constant and unconditional support. My obligation to them grows with time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to 3D Object Detection

In the context of autonomous driving, it is crucial to understand the surrounding environment around the vehicle. In order to achieve this, one of the most fundamental tasks to tackle is 3D object detection, which aims to accurately localize nearby objects of interest, including other vehicles, pedestrians, cyclists and all kinds of obstacles to which the vehicle needs to pay attention. For each object of interest, the detector should output its 3D location, 3D size, yaw rotation, class and optionally a confidence measurement.

To perform 3D object detection, we need to first obtain a sensing of the surrounding in order to reason about it. Multiple types of sensors can be utilized to enable 3D object detection, including LiDAR (Light Detection and Ranging) sensors, Radar, cameras, etc. Among these sensors, LiDAR is the most indispensable sensor for 3D object detection, for its accurate localization of obstacles, its relatively high sensing range (up to 300 meters), and its relatively high sampling density (up to 4.8 million points per second)[1]. LiDAR shoots laser beams at various directions and timestamps the returned points of each beam, if the beam hits an obstacle and is reflected back to the sensor. LiDAR sensors also captures the reflectance (intensity) value along with each detected point, which is measured by the ratio of the number of photons reflected back to the number of photons emitted. The intensity signal can be affected by the incidence angle, the surface composition, range and various factors. All the detected points in each frame form a point cloud.

Based on the LiDAR point cloud, state-of-the-art detection methods first optionally convert it to a structured representation including voxel grid or range image, then apply deep neural networks on top of the input data to obtain detection results. Existing works can be categorized by the input representation they use, and the representation determines the basic structure of the neural network.

The task of 3D object detection is extremely important for reliable autonomous driving. It is one of the most up-stream tasks in the software stack and influences the outcome of the whole system. More over, mistakes made by the detector, for example missed pedestrians, will likely lead to collisions and traffic accidents, which are unacceptable for self-driving cars.

[1] `https://en.wikipedia.org/wiki/Velodyne_LiDAR`

1.2 Related Work

1.2.1 3D Point Cloud Object Detection

3D LiDAR point cloud object detection aims to detect surrounding objects by estimating each object's class, location, size and orientation from point cloud data. State-of-the-art 3D Point Cloud object detection methods can be divided into four categories: Voxel based detection, perspective view detection, point based detection, and multi-modal fusion based detection.

Voxel based detection. This paradigm converts the input point cloud into a regular grid space named voxelized representation by extracting the feature of each voxel from the points inside the voxel. The pioneering work VoxelNet[39] uses a point-net like structure named voxel feature encoding layer to extract voxel features, then apply 3D convolutions to refine the 3D feature map. Afterwards 2D convolution on the horizontal plane is applied on the 3D feature to perform bounding box classification and regression, treating multiple feature maps across different vertical heights as feature channels. Building on VoxelNet, SECOND[38] proposed to use sparse 3D convolution and 3D submanifold convolution to exploit the natural sparsity in point cloud data to reduce memory footprint and increase inference speed. PointPillars[13] proposed to merge voxels along the same vertical axis into a large voxel denoted a pillar, exploiting the fact that objects do not lay on top of each other in street scenes. Thus, 3D convolution is replaced by 2D convolution to refine the resulting 2D birds eye view (BEV) feature map, and running time is significantly reduced. The advantage of voxelized representation is that is preserves metric space, which means object sizes remain constant with respect to the distance from sensor, and thus making 3D convolution filters easier to learn.

Perspective view detection. Instead of detecting objects in birds eye view, which suffers from point sparsity resulting in computation waste and information loss when extracting feature from highly populated voxels, LaserNet[16] proposed to present point cloud data in perspective view and perform 3D object detection in range image. Because the 2D perspective view feature map is much more compact, the inference time is significantly reduced compared to voxel based methods, but at the cost of lower accuracy due to object overlap and non-preserving metric space.

Point based detection. This paradigm of work regress bounding boxes directly from points instead of converting them to a structured representation beforehand. PointRCNN[29] proposed to refine point features using PointNet++[23], and perform bottom up proposal generation and canonical bounding box refinement. Similar to PointRCNN[29], VoteNet[24] proposed to regress only the bounding box center from each seed point instead of regressing the full bounding box. Then points are grouped by the estimated center, and bounding box regression is performed on grouped points. An advantage of detecting objects from points is that accurate coordinates of points are preserved until the final box refinement stage, while this precise localization information is partially lost due to feature downsampling when convolutions are applied.

Multi-modal fusion based detection. Three types of methods discussed above all have their own merits and shortcomings, hence it is often desirable to combine these methods together. PV-RCNN[30] claimed that voxel based methods can produce high quality proposals, and point based methods excel at accurate bounding box regression, therefore proposed a framework that

uses voxelized feature map for proposal generation and point based learning for proposal refinement. MVF[40] claimed that perspective view detection methods perform well at longer ranges where the point cloud becomes very sparse, and especially on small objects, and hence proposed to extract point level feature in both perspective view and birds eye view, then fuse features across two views at point level, before feeding into a voxel based detector.

This thesis mainly focus on analyzing voxel based detection methods as they achieve the state-of-the-art performance while maintaining a clean and simple architecture. This allows us to make modifications and analyze the results more easily. We believe our findings generalize to other detectors as well.

1.2.2 Point Cloud Semantic Segmentation

3D Point Cloud Semantic Segmentation aims to estimate point-wise class labels in order to recognize objects from points. PointNet[22] proposed to exploit the order-invariant nature of point cloud data by using max-pooling operator over points during feature extraction. Building on PointNet[22], PointNet++[23] furthur proposed to capture spatial relationships of points by applying PointNet on local regions and combine their outputs hierarchically. These two methods learn directly from raw point cloud input, and perform well in indoor scenes. However, due to the high memory footprint, they do not work well with street scenes in autonomous driving context. SqueezeSeg[36] and SqueezeSegV2[37] proposed to perform point cloud segmentation in perspective view on range images, exploiting sensor geometry using 2D convolutions. DarknetSeg[2] further investigated the representation power of the convolutional network, and obtained better results by using deeper networks. In this thesis we used DarknetSeg with 21 layers for point cloud semantic segmentation.

1.2.3 Synthetic Data Generation

Though there are many simulators (*e.g.*, Sim4CV [15], Nvidia Drive [3]) that can be used for synthetic data generation, most these simulators are not open-source (not easy to make modifications) and free-to-use license is not available (*i.e.*, derivative products are not allowed). As for the open-sourced simulators, AirSim [28] and Carla [8] are popular due to detailed documentation and many available sensors. However, AirSim does not allow low-level control over every agent in the way that Carla allows, though AirSim has advantages in aerial data capture. In addition to simulators, commercial video games such as GTA-V [26] can be also used for synthetic data generation but they do not allow for low-level control of the scene elements. Accordingly, we have selected to use Carla for data generation as it affords the most flexibility and customization.

1.3 Challenges of 3D Object Detection

Although there has been a substantial amount of effort made in the field of 3D object detection, the capabilities of state-of-the-art detection methods still do not meet the requirements for establishing a fully autonomous vehicle. Some of the most difficult challenges of 3D object detection

include: accurately classifying objects with complicated shapes like pedestrians, detecting far away objects, and detecting occluded objects.

The root cause of the under-performance mentioned above unlikely reside in the model design, but rather in the input point cloud data. For each captured point, the data only describes its location and intensity. It is hard to reason about fine grained shape and texture merely from geometric cues, and intensity value is highly sensitive to factors not related to object shape itself, including range and incidence angle. Therefore, state-of-the-art models struggle to classify pedestrians accurately. More over, the resolution of the point cloud data in popular 3D object detection datasets including [33] and [9] is much lower than normal rgb camera, making far away objects that are visually distinguishable poorly captured by point cloud. Without enough coverage, it is infeasible to detect the far away objects. Lastly, occluded objects, like pedestrians in crowds, are not depicted well by point cloud data, and are hence hard to be detected.

The challenges mentioned above are caused by the intrinsic limit of point cloud data, and are unlikely to overcame by proposing model designs. They may be address by introducing rgb images into the detector, by existing works have been unable to fuse point cloud and rgb images effectively, due to the lack of accurate depth estimation in rgb images. With recent advancements in LiDAR sensing, it is possible to obtain point cloud data with richer measurements. For example, some LiDARs have higher resolutions, while others can also measure brightness. They provide enriched point cloud information, which is potentially beneficial for 3D object detection. However, most popular 3D object detection datasets, including KITTI[9], nuScenes[5] and Waymo Open Dataset[33], only provide the limited point cloud data. As most state-of-the-art detectors are designed upon datasets that lack the rich point cloud information, they are unable to explore the benefit of enriched point cloud.

1.4 Approach Proposal

In this thesis, we aim to address the above challenges and push the limit of 3D object detection by enriching point cloud information in three ways: capturing multi-echo points in each beam; capturing ambient value for each beam; and increasing point cloud density.

Conventional LiDAR sensors with avalanche photodiode (APD) require hundreds or thousands of photons to be reflected in a laser beam in order to activate the photodetector and consequently detect a reflection point. If multiple points are detected by one beam with strong intensity, then they can all be detected. Yet popular datasets only provide up to one point per beam, because a laser beam is seldom partially reflected at multiple locations with high reflectance. However, with the recent development of single-photon LiDAR with single photon avalanche diode (SPAD), it is now possible to activate the photodetector with only one reflected photon. Therefore, it is easier for the LiDAR to detect multiple echo points from each beam. More over, if we increase the fill factor of the LiDAR, making the laser beams thicker, then the thick laser beams are more likely to hit multiple objects. With these advancements, we can obtain more points in the point cloud of each frame, and consequently present finer details of the surrounding environment. In addition, among multi-echo points detected by the same laser beam, the nearer ones often lie on object boundaries or semi-transparent surfaces in order to cause a partial reflection. Hence, multi-echo point cloud not only captures more points, but potentially convey

key properties of points as well. With more points, faraway objects will be easier to distinguish. And if we have knowledge of boundary points, we can localize object bounding boxes more accurately.

In this thesis, we collected multi-echo point cloud data using a prototype single-photon LiDAR, which can capture up to three points from each beam. Points from each beam are ordered by reflectance, with the first echo points having the largest intensity value and the third having the lowest. An visualization of captured multi-echo points is shown in Fig.2.2. We propose to utilize multi-echo point cloud by treating multi-echo points from each beam as individual points, and by feeding this multi-echo point cloud into state-of-the-art 3D object detectors. The investigation in this direction is presented in Chapter 2.

In addition to capturing reflected photons emitted by the LiDAR itself, the photodetectors of a LiDAR sensors can also capture photons from the infrared sunlight. This allows LiDAR sensors to measure an ambient signal associated with each laser beam. This ambient signal can be interpreted as the same signal which is contained in ordinary RGB images, but just on a different part of the light spectrum. Complementary to the point location and reflectance, the ambient value conveys texture-related properties of the detected obstacle, and potentially provides useful information when grouping points together or classifying the obstacle object. Visualizations of sample ambient images and intensity images are shown in Fig.3.3, and it demonstrates how ambient images capture finer details of the pedestrians compared to intensity images. To investigate how ambient value can help 3D object detection, we used the same prototype single-photon LiDAR mentioned earlier, which is also capable of capturing ambient signal.

We proposed two approaches to utilize the ambient information. In the first approach, we assign each point with the the ambient value captured along with the same laser beam which detected that very point. If multiple points are detected by one laser beam, then they share the same ambient value. Then we extend state-of-the-art 3D object detectors to process the ambient value the same way as intensity value and other point properties are treated. This approach, albeit simple, might suffer from ineffective ambient feature extraction if they are presented in sparse 3D structure. Our second approach proposal is to utilize ambient value in a dense form by performing point cloud semantic segmentation in perspective view, combining range images and ambient images as input data. Then we extend state-of-the-art detectors to process the output segmentation estimation as extra properties of each point, constituting a multi-view segmentation-detection fusion framework. Details regarding this direction of exploration can be found in Chapter 3.

The last direction that we investigated to enrich point cloud information is to increase point cloud density and range. State-of-the-art LiDAR sensors are able to capture points at increasingly higher resolution, both horizontally by shooting laser beams more frequently during its rotation time cycle, and vertically by adding more laser beams. Therefore, it is important to learn in advance how state-of-the-art detectors cope with high density point cloud data. Because we do not have access to high-end LiDAR sensors with high density sensing capability, we utilized an autonomous driving environment simulator, Carla[8], to generate a synthetic dataset with varying point cloud density and range. This not only allows us to control the point cloud density to make it beyond current state-of-the-art LiDAR metrics, but also let us make fair comparisons by maintaining the same labels while changing the point cloud density. Visualizations of synthetic point cloud is shown in Fig.4.2 and Fig.4.3. Note how the synthetic point cloud is captured in a larger range and at a much higher density.

An immediate challenge faced by state-of-the-art 3D object detectors when working with high density point cloud is that the high volume input data exceeds GPU memory limits. To address this problem, we adopted an input slicing technique which cuts input point cloud into multiple pieces, then perform object detection in each piece one-by-one, and combine the detections from all the pieces before using non-maximum-suppression to deduplicate detections near slicing boundaries. Chapter 4 contains the details of the high density point cloud data collection, model design and experiment results.

1.5 Contributions

The contributions of this thesis are three-fold:

(1) We observed the effectiveness of using multi-echo point cloud data collected from single-photon LiDAR with increased fill factor for 3D object detection.

(2) We demonstrated the improvement in pedestrian detection accuracy brought by capturing extra ambient signal along with the point cloud. We also proposed a multi-view segmentation-detection fusion framework to take full advantage of the ambient information.

(3) We exploited the usefulness of high density point cloud data in the context of 3D object detection by generating a synthetic dataset using simulation systems. We also proposed a point cloud slicing technique to run state-of-the-art detectors while keeping GPU memory in bound.

In summary, we obtained significant improvement by introducing enriched point cloud data into the task of 3D object detection, and proposed specific model designs to leverage the information properly. Our findings call for more advanced 3D object detection datasets and benchmarks with richer point cloud data, and for more research in exploiting the enriched information.

Chapter 2

Object Detection with Multi-Echo LiDAR Point Cloud

In Section 1.3 we described the challenges of detecting occluded objects in crowded scenes and detecting far away objects. These two challenges originate from the limited depiction of such objects from point cloud data. In this chapter we propose to enrich point cloud data using multi-echo point cloud obtained by single-photon LiDAR with increased fill factor. The resulting point cloud contains more points, and may potentially alleviate the problem.

2.1 Background

LiDARs sense remotely by shooting pulsed laser beams in a certain direction, and measure the range between the obstacle in that direction and the sensor using the time of flight of the light. Conventional LiDAR sensors with avalanche photodiode (APD) requires hundreds or thousands of reflected photons to be received in a laser beam in order to activate the photodetector and consequently detect a reflection point. It is possible to detect multiple points by one laser beam if the beam is partially reflected at multiple points with high intensity, yet this is not common and popular datasets including [33] all provide at most one point per beam. However, the state-of-the-art single-photon LiDAR with single photon avalanche diode (SPAD) can be activated by as little as one reflected photon, making weak ranging measurements easier to be captured. Therefore, if a beam is partially reflected at multiple locations, then all the corresponding reflection (echo) points can be captured from this beam. More over, if we increase the fill factor of the LiDAR, making laser beams thicker, the beams are likely to hit more objects. Thus, single-photon LiDARs with increased fill factor can detect more points in the point cloud of each frame, and consequently present finer details of the surrounding environment. This point cloud is denoted multi-echo point cloud. In addition, among the multi-echo points detected by the same laser beam, the nearer ones mostly lie on object boundaries, reflective surfaces, or semi-transparent surfaces in order to result in partial reflection. Hence, multi-echo point cloud not only captures more points, but potentially convey key geometric properties of points as well. With more points in the multi-echo point cloud, faraway objects will be easier to distinguish. And if we have knowledge of boundary points, we can localize object bounding boxes more accurately.

In this thesis, we collected multi-echo point cloud data using a prototype single-photon LiDAR with high fill factor, which can capture up to three points from each beam. Points from each beam are ordered by intensity, with the first echo points having the largest intensity and the third having the lowest. An visualization of captured multi-echo points is shown in Fig.2.2. Based on the captured multi-echo points, we constructed point cloud by treating multi-echo points from each beam as individual points. Then we extend a state-of-the-art 3D object detector, namely SECOND[38], to process this multi-echo point cloud. We observed significant improvement of detectors trained with multi-echo point cloud input data over single-echo point cloud, especially on the recall of detections.

2.2 Model Setup

Though multi-echo point cloud contains richer geometry information and more fine-grained details of the scene, it is unclear how to properly utilize the multi-echo point cloud and there is no prior work has been done in this direction. As the first work in this direction of investigation, we want to first verify the usefulness of additional points. Therefore we build a baseline approach by just transforming multi-echo point cloud data into normal point cloud data form by treating multi-echo points from each beam as individual points, ignoring the intensity ordering of points from each beam. In other words, we take all points the LiDAR detected and process them the same way as regular point cloud, regardless of which points belong to the same group.

With the processed point cloud as input, we perform 3D object detection using the state-of-the-art voxel-based detector SECOND[38]. SECOND structure is shown in Fig.2.1. The detector first samples a few points from each voxel, and apply voxel feature encoding (VFE) layers to extract voxel freature from the points. Then given 3D feature map of the voxels, 3D sparse convolutions and 3D submanifold convolutions are applied to refine the 3D feature map while keeping memory consumption and run time low. Then the resulting 3D feature map is treated as a 2D feature map by regarding multiple feature maps across different vertical heights as feature channels, and is processed by a region proposal network (RPN) with 2D convolutions to generate the final detections.

2.3 Results

2.3.1 Data Collection

To collect multi-echo point cloud, we used a prototype single-photon LiDAR with high fill factor. The LiDAR has 96 channels and each channel shoots laser beams at 600 horizontal directions in each frame, leading to $600 \times 96 = 57600$ laser beams per frame in total. Each beam detects at most three obstacle points, and the points from each beam are ordered by intensity, which is measured by the ratio of the number of photons reflected back to the number of photons emitted. Along with the return points, this LiDAR can also measure the ambient value associated with each beam, which is further investigated in next chapter.

Our dataset is collected by roof-mounting this prototype single-photon LiDAR on top of a vehicle while driving around a North America City. There are 10 video clips in this dataset,

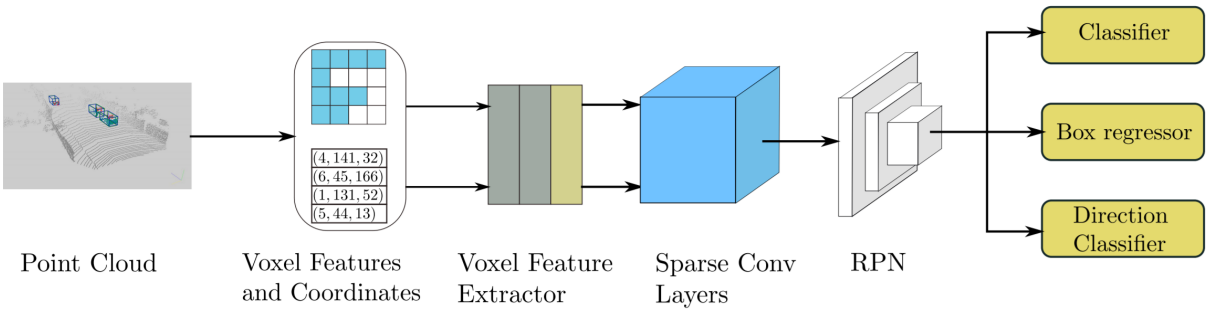


Figure 2.1: **The structure of SECOND.** The input point cloud is first voxelized, with the feature of each voxel extracted using VFE layers. Then a sparse 3D CNN is applied on voxel features to refine the 3D feature. Lastly a RPN with 2D convolutions is used to generate detections. Figure in courtesy of [38].

summing up to 35850 frames. To make sure our dataset encompass daily driving scenarios thoroughly, the video clips collected at diverse scenes including downtown, suburban areas and highway. Fig. 2.3 shows a few scenes in which our dataset is collected. We divided the dataset into training set and evaluation set at ratio 7:3, using 9 video clips for training, including 25002 frames, and 4 video clips for evaluation, summing up to 10848 frames. Training set and evaluation set are hand-picked to make sure the scene diversity remains. We have all the ground truth labeled for cars and pedestrians, along with less common vehicles including trucks, buses and towed trailers. But because we don't have enough ground truth available for these classes, we only train the model to detect cars and pedestrians.

2.3.2 Implementation Details

Baseline settings. For the baseline experiment, we only take the first echo points from each beam, the points with highest intensity value among all the points detected at the same direction. The model structure remains the same across single-echo and multi-echo experiments.

Hyper parameter settings. For the detection of cars, we crop the point cloud based on the ground truth distribution at $[-2.5, 7.5] \times [-100, 100] \times [0, 200]m$ along the $z \times y \times x$ axes, with x axis poinint front, y axis pointing left and z axis pointing upward. For detection of pedestrians, we crop the point cloud in a smaller range at $[-2.5, 7.5] \times [-60, 60] \times [0, 160]m$ along the $z \times y \times x$ axes, as pedestrians become indistinguishable more easily than cars as they get more distant from the sensor. Fig. 2.4 shows the distribution of object locations in birds eye view. For both classes, We use voxel size $v_z = 0.25m, v_y = 0.1m, v_x = 0.1m$, and sample at most 1 point from each voxel, and use the feature of the point as the voxel's feature. To exploit the sparsity of object distribution and increase convergence speed, we utilized ground truth augmentation proposed in [38], and samples at most 100 objects from the pre-built ground truth database. To train the region proposal network (RPN) effectively, we use fixed-size anchors determined based on the means of the sizes and center locations of all ground truths in our dataset with rotations of 0 and 90 degrees. We use anchors of dimension $w = 1.8m, l = 4.5m, h = 1.7m$ for car detection, and anchors of dimension $w = 0.9m, l = 0.9m, h = 1.8m$ for pedestrian detection, both centered

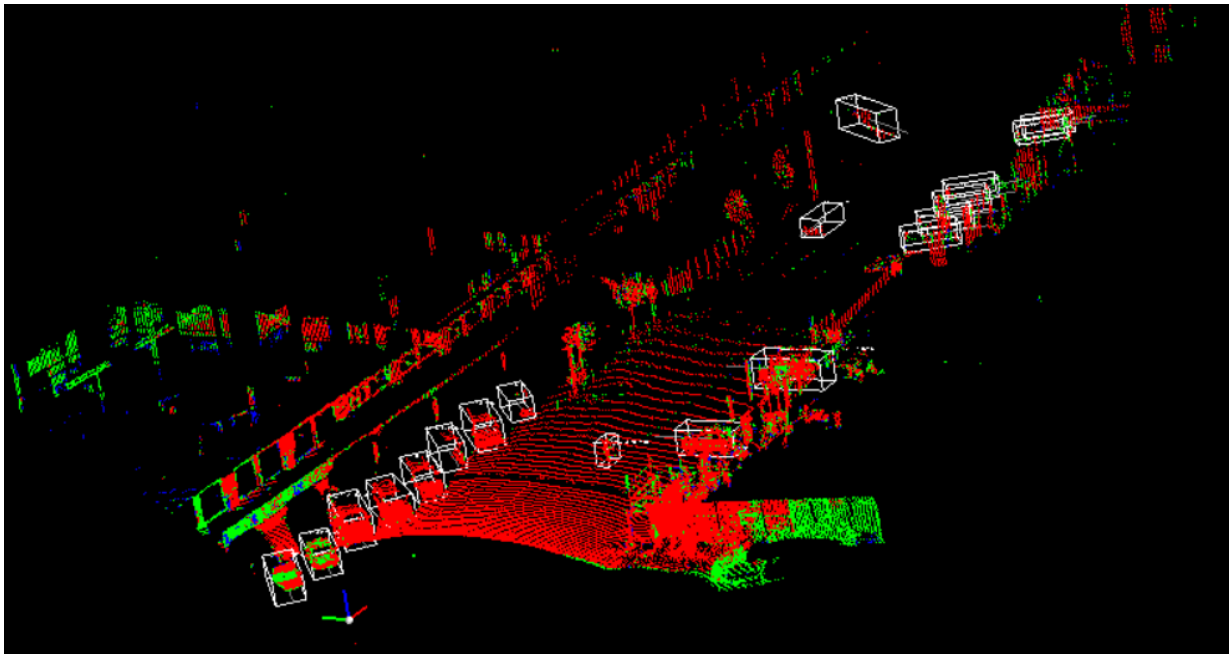


Figure 2.2: **Visualization of sample multi-echo point cloud captured by our prototype single-photon LiDAR.** Red points are the first echo points, green points are the second echo points, and blue points are the third echo points.

at $z = 0.0m$. The model is trained RTX 2080 Ti for 92850 steps, which amounts to around 7.4 epochs. We use cosine learning rate decay, with maximum learning rate at 0.001. Training takes roughly 15 hours to finish.

Evaluation metrics. Our evaluation metrics follow KITTI[10], using average precision (AP) with intersection over union (IoU) threshold at 0.7 and 0.5 for car detection evaluation, denoted AP@0.7 and AP@0.5 respectively, and AP with IoU threshold at 0.5 and 0.25 for pedestrian detection evaluation, denoted AP@0.5 and AP@0.25 respectively.

Average precision is defined as the area under the precision-recall curve, which plots the model output's precision and recall while varying the confidence threshold that is used to filter detections. A few samples of precision-recall curves are shown in Table 2.5. When using a high confidence threshold, there will be only a few detections covering a small portion of ground truths, yet they are likely accurate as the model has the most confidence in them, so the detections have high precision and low recall, corresponding to the top-left part of the plot. In contrary, having a low confidence threshold will greatly increase the number of detection, making them cover most, if not all of the ground truths, yet resulting in a low precision due to all the low confidence false positives included, corresponding to the bottom-right part of the plot. In applications like face recognition on our smart phones, we want detections to have high precision, because the bottom-line is not to allow other people unlock our phone, and we can trade some recall for this, as we are willing to try a few more times once in a while as long as we have good security. On the other hand, in the context of autonomous driving, we need an extremely high recall especially

Figure 2.3: **Scenes in which our dataset is collected.** We collect data in various scenes to make sure the dataset is diverse and cover our daily driving scenarios.

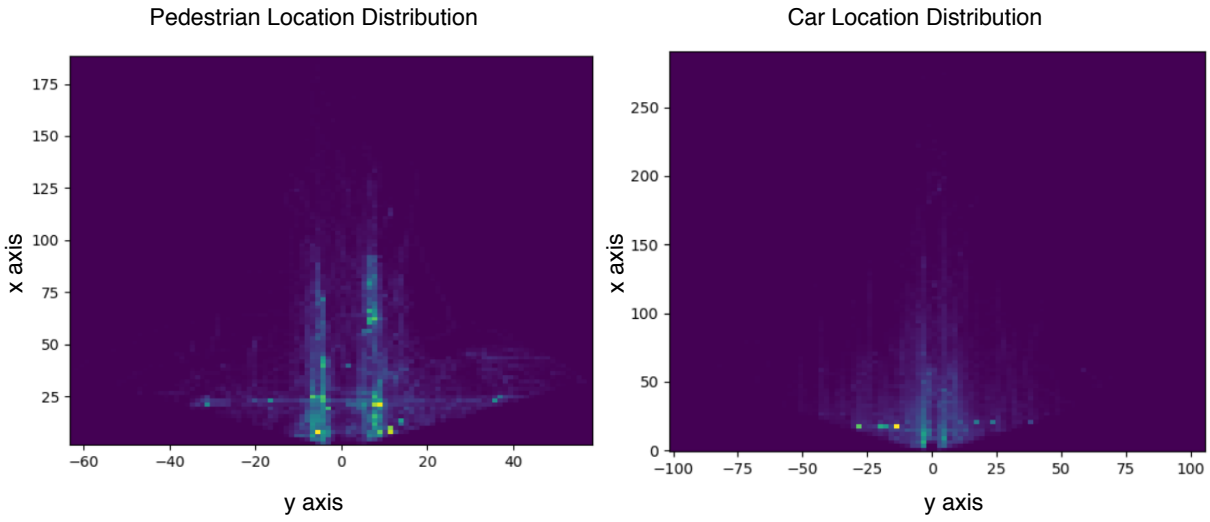


Figure 2.4: **Distribution of object locations in birds eye view.** There are less pedestrians with high distance from the sensor compared to cars.

for pedestrians because it is critical that we do not hit pedestrians, and for this we are willing to sacrifice by tolerating a few emergence brakes. Therefore, AP evaluates the model's accuracy under various scenarios, and has been the dominant metric for detection evaluation.

Class	Car	Pedestrian
near range, $[0, 40)m$	23601	21167
mid range, $[40, 80)m$	21904	14596
far range, $[80, +\infty)m$	16081	3419
all	61586	39182

Table 2.1: Number of ground truths in each distance range in validation set.

Based on KITTI's evaluation metrics, several modifications are made to adapt to our dataset:

1. Because the ground truths in our dataset do not have labels of level of occlusion, truncation or difficulty, we evaluate on all objects.
2. During evaluation, we ignore ground truths with less than 5 points inside. Meaning if an object with less than 5 points inside is missed, it is not counted as a false negative. If it is detected, the detection is neither counted as a true positive, nor a false positive.
3. Unlike evaluation in KITTI, We do not ignore ground truths or detections with 2D bounding box height below a certain threshold when projected to perspective view. We believe the number of points inside object is a more suitable metric for deciding if a model should be able to detect a certain object, instead of seeking help from 2D box height.
4. When evaluating car detection, we ignore the confusing classes including trucks, buses, and towed trailers in the same way we ignore ground truths with less than 5 points inside.
5. Along with evaluating with all ground truths, we also divide ground truths into three distance ranges for distance-wise evaluation. We divide the objects in ranges: $[0, 40)$, $[40, 80)$, $[80, +\infty]m$, denoted near range, mid range, and far range respectively. The number of ground truths in each range in validation set is shown in Table 2.1.

2.3.3 Experiment Results and Analysis

Car detection AP of the single-echo model and multi-echo model are presented in Table 2.2, and AP figures of pedestrian detection are presented in Table 2.3. For car detection, We observe a 3.9 point improvement in AP@0.7, and 5.6 point improvement in AP@0.5. For pedestrian detection, there is a 2.7 point improvement in AP@0.5 and 3.1 point improvement in AP@0.25.

We further analyzed the improvement in AP by investigating the precision-recall curve. From the curves we can tell that, with multi-echo point cloud input, the detections have a higher recall, meaning that the additional points brought by the multi-echo point cloud help reduce false negatives by depicting objects in a more detailed manner.

2.4 Conclusion

In this chapter, we enriched the point cloud input by obtaining multi-echo point cloud data with a prototype single-photon LiDAR with increased fill factor. We demonstrated the significant

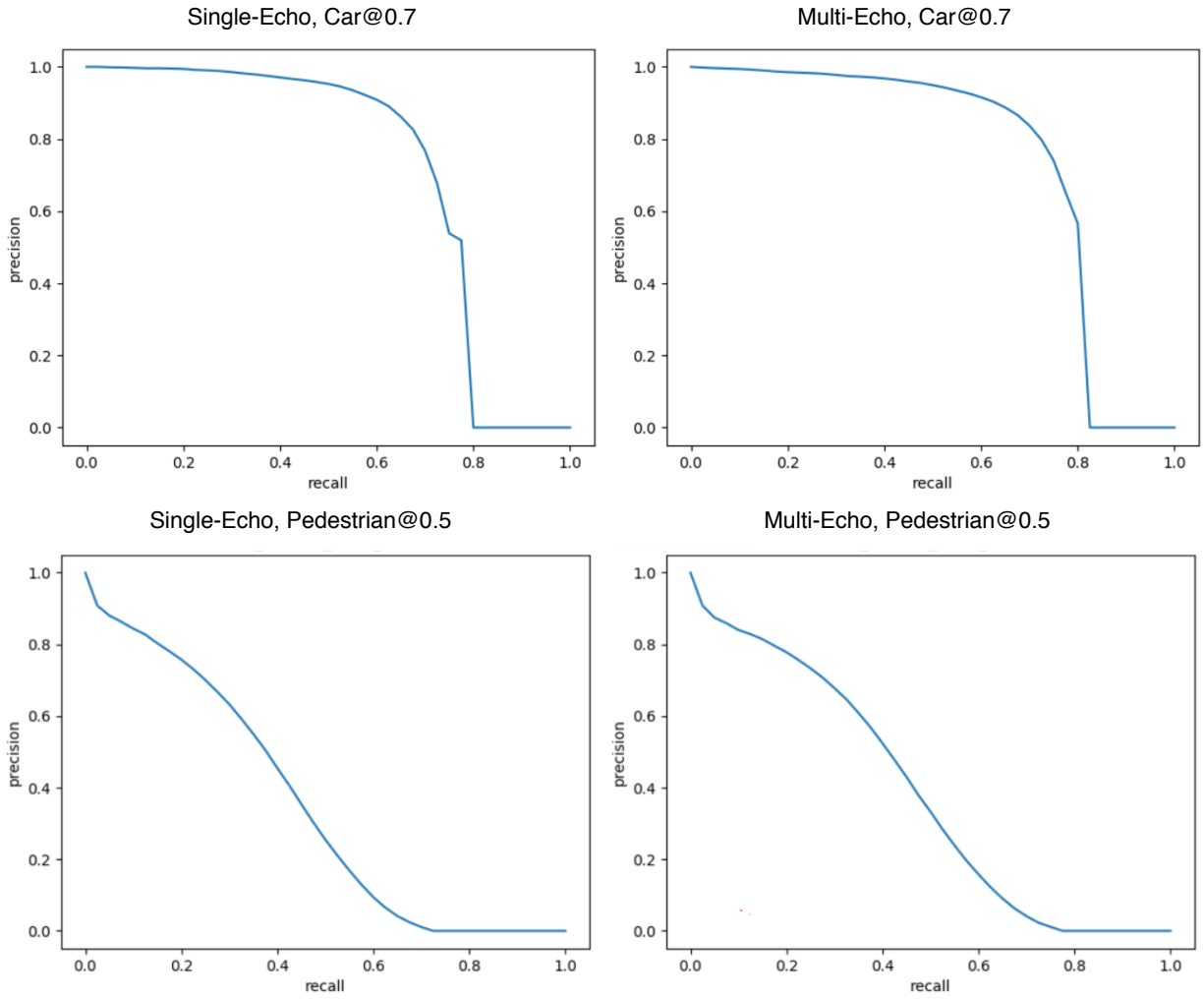


Figure 2.5: **Precision-recall curves of AP@0.7 for car and AP@0.5 for pedestrian.** We can tell from the curve that the boost in AP comes mainly from the improvement in recall, meaning less ground truths are missed by detector.

Method	AP@0.7				AP@0.5			
	all	near	mid	far	all	near	mid	far
single-echo	48.80	69.02	42.56	29.33	68.88	79.59	65.65	63.57
multi-echo	52.73	74.96	42.88	30.56	74.47	86.75	65.54	65.11

Table 2.2: Performance comparison of 3D car detection with single-echo point cloud and multi-echo point cloud.

Method	AP@0.5				AP@0.25			
	all	near	mid	far	all	near	mid	far
single-echo	36.80	45.88	25.27	15.42	46.03	56.39	32.87	20.03
multi-echo	39.55	47.40	29.50	19.68	49.15	58.47	37.23	25.09

Table 2.3: Performance comparison of 3D pedestrian detection with single-echo point cloud and multi-echo point cloud.

improvement in 3D object detection accuracy using the multi-echo point cloud, especially improving the recall.

Chapter 3

Object Detection with Ambient Signal

In Section 1.3 we described the challenge of detecting objects with complicate shape like pedestrians. In this chapter, we propose to address this challenge by additionally capture an ambient signal along with each laser beam, which presents finer texture details.

3.1 Background

Some advanced LiDAR sensors, including the prototype LiDAR we used in the previous chapter, are able to capture photons from the infrared sunlight that is reflected by obstacles in the target direction, which allows the LiDAR to measure an ambient signal associated with each laser beam. This ambient signal can be interpreted as the same signal which is contained in ordinary RGB images, but on a different part of the light spectrum. Complementary to the point location and intensity, the ambient value conveys texture related properties of the detected obstacle, and potentially provide useful information when grouping points together or classifying the obstacle object. Visualizations of sample ambient images and intensity images are shown in Fig.3.3, which shows how ambient images capture finer details of the pedestrians compared to intensity images. Hence, we expect the ambient information to be helpful for object detection, especially for pedestrian detection.

To utilize the ambient information, we first attempted to assign each point with the the ambient value captured along with the same laser beam which detected that very point, and extend the detection framework SECOND[38] to process the the point cloud with the addition ambient property. If multiple points are detected by one laser beam, then they share the same ambient value. But we found that it is non-trivial to take full advantage of the ambient information, as experiment results show that no improvement is brought by the extra ambient information under this approach.

We suspect that it is hard to extract useful features from ambient information if presented in sparse 3D structure, so our second attempt is to utilize ambient value in a dense form by performing feature extraction in perspective view, combining range images and ambient images as input data. Specifically, we directly performed point cloud semantic segmentation on the 2D images, and the feature we intended to extract is the estimated class label. Then we extend state-of-the-art detectors to process the output segmentation estimation as extra properties of each

point, constituting a multi-view segmentation-detection fusion framework, as shown in Fig.3.1. Under the assistance of segmentation estimation learned from ambient images, the proposed detector exhibited noticeable improvement on pedestrian detection accuracy.

Our results suggest that, although ambient value is helpful for 3D object detection, some state-of-the-art LiDAR based detectors are not able to utilize this information efficiently, and proper model design is required to take full advantage of this extra information.

3.2 Model Design

3.2.1 Detector Extension Approach

In this approach we try to utilize ambient signal by treating it as an additional property of points from LiDAR point cloud, in addition to original properties including x/y/z coordinates and intensity signal. Then we extend state-of-the-art detectors to process the additional ambient property the same way as other point properties.

We construct input point cloud data from the sensor output, including $600 \times 96 = 57600$ sets of points detected by each laser beam, each set including $[0, 3]$ points, and 57600 ambient signals associated with each beam. Because multi-echo points can be detected by each beam, yet only one ambient signal is associated with each beam, we assign each ambient signal to all the points detected by the associated laser beam. Then like Section 2.2, we treat multi-echo points from each beam as individual points, resulting in an input point cloud tensor with shape $N \times 5$, with N being the total number of points detected by all laser beams, and 5 corresponding to the 5 features of each point, namely x/y/z coordinates, intensity value and ambient value.

To perform 3D object detection on the point cloud data with additional ambient feature, we extend a state-of-the-art detection method SECOND[38] to adapt to the expanded input feature dimension. We first extend the voxel feature encoding (VFE) layers in SECOND to support 5-dimension feature, then increment the input channel number of the first subsequent 3D convolution after VFE layers. Other than this, the model structure stays the same.

3.2.2 Multi-View Fusion Approach

From Fig.3.3 we can tell how ambient images depict pedestrians in a more detailed way than intensity image, so there is a possibility that the ambient information is not effectively utilized by the extended SECOND model proposed above. Based on this assumption, we further make three assumptions about the feature learning process. Firstly, the ambient image can help the model draw the contour of objects, which can be very useful for classifying objects. However, distinguishing the contour of an objects requires reasoning of the foreground points and background points together, and this is hard to achieve using voxelized representation, in which foreground and background points are naturally far apart. Secondly, detecting pedestrians requires distinguishing them from objects with similar sizes and shapes like traffic signs, and this calls for learning on the fine grained details, which is hard to grasp using the coarse resolution of voxelized representation. If we increase the resolution, then the sparsity in input feature map will make it hard to learn convolutional filters effectively, not to mention not having enough GPU

memory to hold the larger tensors. Thirdly, unlike other point cloud properties, ambient appearance is highly sensitive to the sunlight exposure, which means the varying direction of sunlight and complicated sunlight reflection makes convolutional filters on ambient feature harder to learn than other features. This is even more so when learning 3D convolutional filters with 3D voxelized representation, due to the increased complexity.

Based on these assumptions, we proposed to extract ambient feature in 2D perspective view using 2D convolutional networks. This will enable the model to reason about object contours more easily, as background points and foreground points adjacent to object contours are natually close in perpective view. Also, unlike voxelized approach, no point will be discarded and therefore enable the model to learn the fine grained details. Lastly, it is easier to grasp the complicated variance of ambient appearances by learning 2D convolutional filters.

In order to extract ambient features from 2D perspective view while performing object detection in 3D view, we need to fuse features learned from perspective view into the voxel based detector. Generally there are two ways to fuse feature, one is point-level fusion and the other is voxel-level fusion. The point-level fusion approach takes each pixel from 2D perspective view, then treats the pixel's feature as an additional feature of the point that corresponds to the pixel. This approach is easy to implement, but suffers from the fact that only early phase fusion is carried out. The voxel-level fusion approach aims to perform late phase fusion by fusing downsampled 2D feature onto the downsampled 3D voxels. MVX-Net[31] proposed to project each downsampled voxel onto the 2D feature map, and use RoIPooling to extract the 2D feature. This approach suffers from feature alignment problem, as the pooled regions includes information from the whole frustum that the voxel of interest resides in, which is not aligned with this very voxel. Deep Continuous Fusion[14] proposed to address the feature alignment problem by first finding k nearest LiDAR points to a voxel, and project these LiDAR points onto perspective view feature map, then aggregate the features of the projected points and fuse the aggregated feature to the voxel. This approach utilize heuristic methods including KNN for feature fusion and is hence not robust to corner cases like empty voxels, in which cases this approach can end up fusing features from completely irrelevant parts of the scene. Since there is no good way of addressing feature alignment issue with voxel-level fusion, we chose point level fusion as our feature fusion approach.

Another design choice is what features should we aim to learn from perspective view input. One option is to train the 2D convolution network end-to-end with the 3D detector, yet it will be hard for the gradient to propagate from 3D detection network output all the way through the 3D convolutions and reach 2D deep network. We may also train a 2D detector on the perspective input separately, and extract the 2D features middle way. However, it is unclear how the 2D detection objectives related to 3D detection. Hence, we proposed to train a point cloud segmentation network and directly classify each point in the point cloud. Then we fuse the estimated class of each pixel in perspective view to the corresponding points.

Our approach share merit with PointPainting[34], in that they also trained a segmentation network and used the segmentation estimation for detection. However, we highlight our key difference is that PointPainting is based a different setting, where both RGB image and point cloud data are available, along with segmentation label on RGB images. Yet our experiments are targeting the setting with only point cloud data available. Also, they train the segmentation network on the RGB images, and project each LiDAR point to the RGB image to extract the

segmentation estimation, but we directly train the segmentation network on the point cloud, hence we have point to pixel correspondence and projection is no longer required.

Our model consists of two parts, the segmentation network and the detection network. The structure is shown in Fig.3.1. Our segmentation network is designed based on Darknet21Seg[2], which is inspired by Darknet[25]. Darknet21Seg is essentially an encoder-decoder hour-glass-shaped network, as shown in Fig.3.2. The encoder learns multi-scale feature abstraction of the input image by utilizing different levels of down-sampling factors up to 32. The following decoder up-samples the feature code to the original resolution. Skip connections between layers with same resolution from encoder and decoder are utilized to make low level feature propagation easier, which may get lost during feature down-sampling. Due to the low vertical resolution nature of LiDAR point cloud range images, down-sampling and up-sampling are only applied in the horizontal direction due to retain information in vertical direction. One last 1×1 convolution and softmax is applied to decoder's output feature map to obtain the estimated class probability distribution of each pixel in the range image. Weighted cross-entropy loss is used to train the segmentation network, where weight is the inverse of the frequency of each class, and therefore handles imbalanced data.

Based on Darknet21Seg, we made several modifications to adapt to our own dataset and training goals. Firstly, we changed model input shape from $64 \times 2048 \times 5$ to $96 \times 600 \times 6$, where 96 stands for verticle resolution, 600 stands for horizontal resolution and 6 stands for feature dimensions. The resolution is changed to adapt to our prototype sensor's specifications. Feature dimensions is increased by one due to our new ambient signal in addition to the five original pixel-wise features, including range, x/y/z coordinates and intensity value. Normally, each pixel of the input image corresponds to the point detected by the laser beam shooting in the pixel's direction, but because our prototype single-photon lidar can detect multiple points by each laser beam, we take the nearest point among all points detected by the same beam. We use the nearest point instead of the first echo point because the nearest point is more likely to lie on foreground objects. Because we are using the segmentation results as input for car detection model and pedestrian detection model, we care about the IoU of target class rather than the mean IoU of all classes, so instead of training all classes at the same time, we train two segmentation networks. One network learns to classify car and non-car pixels, and the other learns to classify pedestrian and non-pedestrian pixels.

After obtaining segmentation estimation of the point cloud, we treat the estimated class label as point property and extend SECOND[38] detector to process the point cloud with segmentation result the same way we extend SECOND for ambient values in Section 3.2.1. The class label estimation is encoded as one-hot vectors. Since among all points detected by the same laser beam, only the nearest point is fed into the segmentation network, some points in the multi-echo point cloud do not have the estimated class label. To address this, we extend the one-hot class vector by one dimension to represent unsegmented points.

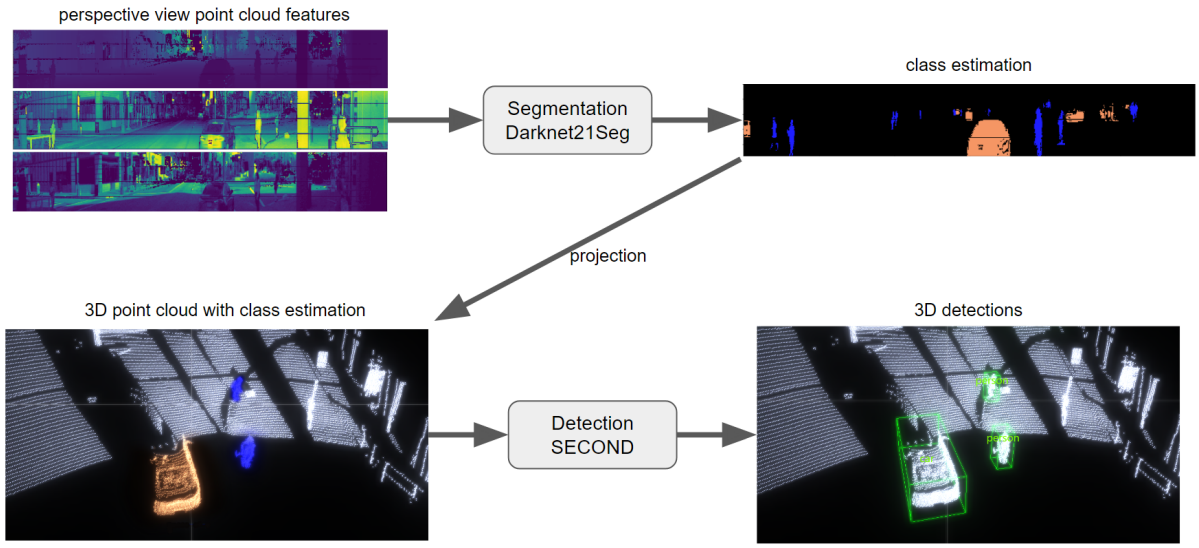


Figure 3.1: **The structure of our proposed multi-view segmentation-detection fusion framework.** For segmentation we use Darknet21Seg, which is shown in Fig.3.2. For detection we use SECOND, which is shown in 2.1. The segmentation input feature includes range, x/y/z coordinates, intensity and ambient value.

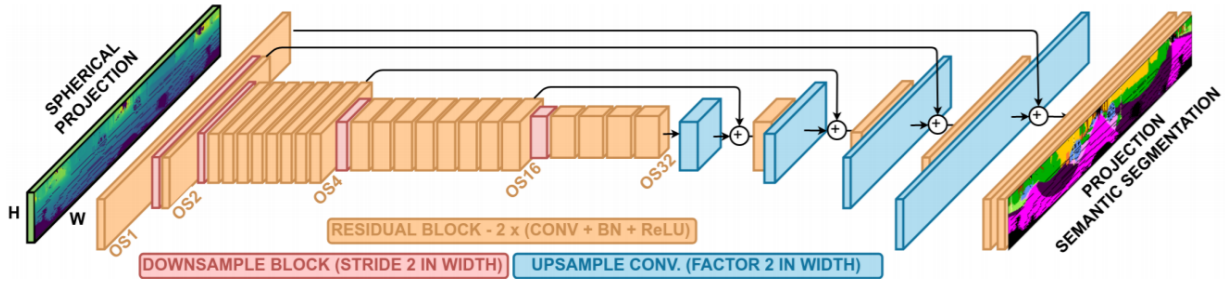


Figure 3.2: **The model structure of Darknet21Seg.** The model is essentially an encoder-decoder hour-glass-shaped network with skip connections. Figure in courtesy to [17].

3.3 Results

3.3.1 Data Collection

We use the same dataset collected in Section 2.3.1 for this chapter. Aside from capturing multi-echo point cloud, our prototype single-photon LiDAR can also capture the infrared (IR) sunlight reflected by objects from the same direction as the laser beam between consecutive laser signal detection. Intuitively, each beam from our LiDAR can detected up to three obstacle points, along with one ambient signal. It is worth noting that some laser beams will not be able to detect any point, for example if all photons are absorbed by a black surface, or if the beam is pointed at the

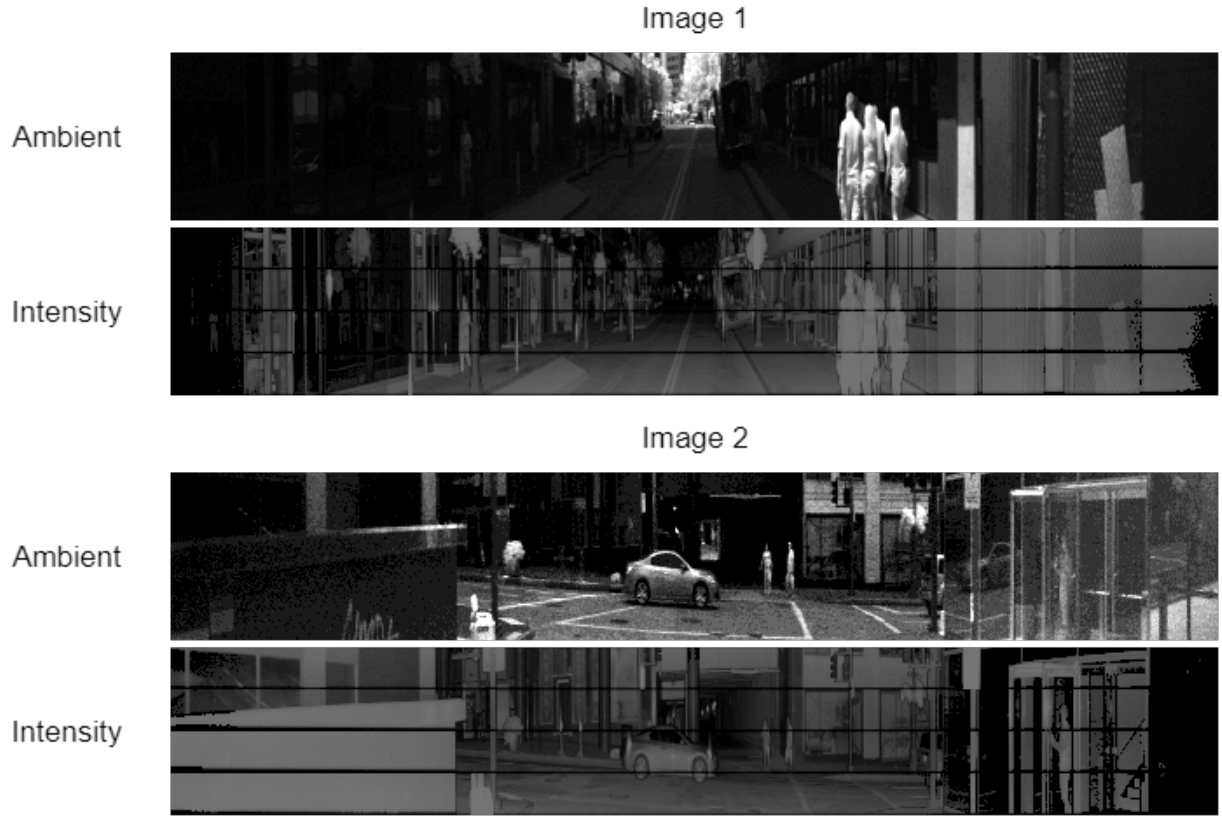


Figure 3.3: **Visualization of sample ambient image and intensity image.** Note how ambient image captures finer details of some objects, especially small objects like pedestrians.

sky. However, the ambient signal can still be measured in such cases. Two sample visualizations of the ambient image and the intensity image of first-echo points are shown in Fig. 3.3. From the visualizations we can see how ambient images are able to capture finer details of some objects, especially small objects like pedestrians.

For the experiments on the multi-view segmentation-detection fusion approach, we need to spare some data for segmentation network training. Here we split the data into three sets, named part A, B and C. We train the segmentation network on part A, then estimate the segmentation results of frames in part B and C. Then we train the detection network on part B with estimated segmentation results as additional input, and evaluate on part C. Of the 14 video clips and 35850 frames contained in the whole dataset, part A contains 5 video clips summing up to 13899 frames; part B includes 5 video clips amounting to 15601 frames, and part C includes 4 video clips with 6350 frames in total. The video clips in each part are hand-picked to make sure the scene diversity remains.

We do not have human-labeled segmentation masks for our dataset, so we generate the segmentation label using the detection ground truth bounding boxes by assigning all points in each ground truth bounding box the same class label as the object. This introduce the ambiguity for points on the ground, as some ground truth bounding boxes are marginally above ground while

others intersect with the ground plane. We expect our results to be further boosted if we has consistent and more accurate segmentation label.

3.3.2 Implementation Details

Baseline settings. For the detector extension approach, we use the detection model trained with multi-echo point cloud without ambient signal, the performance of our baseline model is from the multi-echo entry in Table 2.2 and Table 2.3. For the multi-view fusion approach, we retrain the baseline model on the three split dataset.

Hyper-parameter settings. For the detector extension approach, the hyper-parameter settings are the same as in Section 2.3.

For training the segmentation network, we use the learning rate of 0.01 under batch size 2, with a decay rate of 0.96 per epoch, and train the model for 20 epochs. We use learning rate linear warm-up during the first epoch for more stable converge. Other training parameters remain the same as in [2]. For training the car segmentation network, the model estimates the class distribution over three classes: car, other-vehicle, and non-car objects & background, and the class other-vehicle is ignored during training. This makes the segmentation training objective aligned with car detection evaluation metrics, where non-car vehicles are ignored. For training the pedestrian segmentation network, we simply train binary classification to distinguish pedestrian and non-pedestrian pixels.

Evaluation metrics. For the detection evaluation in both approaches, the evaluation metrics are the same as in Section 2.3.

For segmentation result evaluation in segmentation-detection fusion approach, we compute class-wise intersection over union (IoU), which is the number of correctly classified pixels of a particular class divided by the number of pixels either belonging to this class or mis-classified as this class. The class-wise IoU is calculated over the whole evaluation set, instead of calculating per-frame IoU and averaging over all the frames. In this way, although frames with no objects from a particular class may have an IoU value of 0 even if only very few pixels are mis-classified, these frames will not interfere with the evaluation of that class since they will not have much influence on the confusion matrix over the whole evaluation set. During evaluation of car segmentation, pixels that belong to non-car vehicles are ignored, meaning these pixels will not count towards the confusion matrix. Also, if a pixel of car is mis-classified as a non-car vehicle, it will not be counted. This evaluation metric is designed to align with the evaluation metrics of car detection. The detection evaluation metrics remain the same as in Section 2.3.

3.3.3 Experiment Results and Analysis

Detector Extension Approach

Table 3.1 and Table 3.2 shows the accuracy of detection model trained with ambient information. Interestingly, we find that the extend detector with ambient input is on par with the detector without. This means either the ambient information is not helpful for performing detection, or the model structure is not able to exploit the full advantage of the ambient signal.

Method	AP@0.7				AP@0.5			
	all	near	mid	far	all	near	mid	far
without ambient	52.73	74.96	42.88	30.56	74.47	86.75	65.54	65.11
with ambient	52.42	74.97	42.36	30.27	74.58	86.96	65.76	65.23

Table 3.1: Performance comparison of 3D car detection with ambient signal.

Method	AP@0.5				AP@0.25			
	all	near	mid	far	all	near	mid	far
without ambient	39.55	47.40	29.50	19.68	49.15	58.47	37.23	25.09
with ambient	39.27	46.71	29.85	20.31	49.33	57.89	39.21	25.61

Table 3.2: Performance of 3D pedestrian detection with ambient signal.

Multi-View Fusion Approach Oracle Experiment

To understand the upper bound of this approach, we perform oracle experiments by assuming the segmentation estimation is perfect and use the segmentation ground truth instead of segmentation model predictions. In other words, we investigated how point-wise class information can boost detection accuracy. Because there is no need to train a segmentation model for this section, the oracle experiments are carried out on the two split dataset used in previous chapters. For baseline experiments, we used the SECOND[38] detection model with multi-echo point cloud input, the same as the multi-echo entry in Table 2.2 and Table 2.3. The only difference between our baseline setting and segmentation-detection fusion oracle experiment setting is that in the latter, point cloud segmentation ground truth is fed into the detector as point-wise feature.

Table 3.3 and Table 3.4 show the results of oracle segmentation-detection fusion experiments. With point-wise class information, the average precision at IoU threshold 0.7 (AP@0.7) of car detection results improved by over 7 points, and AP@0.5 improved over 6 points. For pedestrian, the improvement is much more significant, with AP@0.5 improved by more than 28 points and AP@0.25 improved by over 43 points. The improvement is most prominent for pedestrians in mid and far distance ranges. This means that for pedestrian detection, the point segmentation results, if accurate, have the potential to greatly boost the accuracy of detection results. But for car detection, the room for improvement is relatively limited.

Method	AP@0.7				AP@0.5			
	all	near	mid	far	all	near	mid	far
baseline	52.73	74.96	42.88	30.56	74.47	86.75	65.54	65.11
seg-det fusion	60.02	80.42	57.24	34.42	81.04	90.76	80.70	78.41

Table 3.3: **Performance of oracle multi-view segmentation-detection fusion car detection.** Point cloud segmentation ground truth is fed into the seg-det fusion entry, but is not utilized in the baseline entry.

Method	AP@0.5				AP@0.25			
	all	near	mid	far	all	near	mid	far
baseline	39.55	47.40	29.50	19.68	49.15	58.47	37.23	25.09
seg-det fusion	68.05	72.99	64.20	56.61	92.23	93.20	91.97	90.86

Table 3.4: **Performance of oracle multi-view segmentation-detection fusion pedestrian detection.** The detection accuracy improvement brought by point-wise class information is much larger than that in car detection.

Segmentation Results

We train the Darknet21Seg[2] model on our three-split dataset for car segmentation and pedestrian segmentation, and the results are presented in Table 3.5 and 3.6 respectively. The pedestrian segmentation IoU is much lower than car segmentation, meaning that pedestrian point cloud segmentation is intrinsically a harder task. This is expected for two reasons. Firstly, there are many objects on the street that has a similar shape and size with pedestrians, such as traffic signs, garbage bins and decorative plants. To perform pedestrian segmentation accurately the model needs to learn to grasp the fine grained geometric cues. On the other hand, there are not much objects that resemble the size and shape of cars. Secondly, cars typically only appear in open space like on roads, but pedestrians most often appear on sidewalk and are more easily occluded.

We also carried out ablation experiments on the ambient information, which results are also shown in Table 3.5 and 3.6. From the results we can see that using ambient information does not help car segmentation. There is a slight drop after adding ambient feature input, but we believe this is due to the result variance caused by training instabilities. On the contrary, ambient signals bring a noticeable improvement (0.11 in IoU) to pedestrian segmentation. This confirms our previous assumption that ambient image provides finer texture details and could be most helpful for distinguishing small objects like pedestrians.

Method	Car IoU	Background IoU
with ambient	0.809	0.984
without ambient	0.816	0.984

Table 3.5: **Car segmentation evaluation.** Results of models trained with and without ambient information input are on par, meaning ambient signals do not provide much help in car segmentation.

Multi-View Fusion Approach

Using the segmentation results as addition detector input, we trained two detection models for car detection and pedestrian detection separately, and the results are shown in Table 3.7 and Table 3.8 respectively. From the results we can see that using segmentation results for detection improves pedestrian detection accuracy by over 1 point in AP@0.5 and AP@0.25, but car detection accuracy does not benefit from this approach. This conforms with our observations in the oracle

Method	Pedestrian IoU	Background IoU
with ambient	0.264	0.988
without ambient	0.253	0.989

Table 3.6: **Pedestrian segmentation evaluation.** The IoU is much lower than car segmentation, indicating pedestrian segmentation is a harder task. Also, ambient input boosts pedestrian segmentation IoU by 0.11.

experiments that the room for improvement from point-wise class information in car detection is limited.

Although pedestrian detection accuracy improved on the whole evaluation set, the accuracy dropped for pedestrians in the far range significantly. We believe this is because feature alignment in perspective view is not very accurate due to the down-sampling, up-sampling, and the large receptive field of the deep convolutional network. This problem is most influential for the faraway objects, which are relatively small in the perspective view. The segmentation training objectives do not penalize errors on instance level, and only optimizes for pixel level, hence the faraway objects incur less gradient signals during training as they occupy less pixels. However, detection evaluation is on a instance basis, and the inaccurate segmentation input will mislead the detector and hence affect detection AP greatly. This means there are inconsistencies between segmentation and detection training objectives, and more research is needed in designing more detection-friendly segmentation objective.

Compared to baseline, our multi-view segmentation-detection fusion pipeline has two major changes, first is the architecture and the training objectives of the fusion pipeline, and second is the additional ambient signal introduced. To investigate how much improvement is brought by the ambient signal, we carried out ablation studies and trained another detector with segmentation results estimated by a segmentation model without ambient feature input. As shown in the last row of Table 3.8, without ambient information, the improvement brought by the new architecture design is negligible. Hence, we can concluded that this architecture successfully exploited the advantage of ambient signal in pedestrian detection by learning ambient features from perspective view.

As a summary of this chapter, we have shown that ambient signals can indeed aim pedestrian detection, but existing state-of-the-art detector like SECOND[38] is not able to utilize this information effectively. To take full advantage of this additional information, proper model design is required. We proposed a multi-view segmentation-detection fusion approach that successfully utilized ambient input, but the inconsistency between segmentation and detection approaches, along with the feature alignment problem in multi-view fusion has made detection accuracy degrade for faraway pedestrians. There is still room for improvement if more effective models with ambient input can be designed.

Method	AP@0.7				AP@0.5			
	all	near	mid	far	all	near	mid	far
baseline	47.75	66.86	41.23	22.63	68.28	80.29	63.54	53.59
seg-det fusion	47.14	66.41	37.00	21.52	68.36	80.11	64.24	54.90
seg-det fusion w.o. ambient	47.00	65.75	41.21	22.22	68.00	79.75	64.25	54.14

Table 3.7: **Performance of multi-view segmentation-detection fusion car detection.** The results of segmentation-detection fusion and baseline are on par.

Method	AP@0.5				AP@0.25			
	all	near	mid	far	all	near	mid	far
baseline	35.22	44.14	29.11	16.10	46.83	59.23	37.77	21.89
seg-det fusion	36.28	45.37	29.67	9.23	48.14	60.51	38.56	15.10
seg-det fusion w.o. ambient	35.20	43.90	28.79	8.77	47.69	60.21	37.58	19.86

Table 3.8: **Performance of multi-view segmentation-detection fusion pedestrian detection.** Using segmentation results trained with ambient signals, the detection accuracy is boosted by over 1 point in AP@0.5 and over 1.3 points in AP@0.25.

3.4 Conclusion

In this chapter, we have demonstrated the benefit of ambient information in pedestrian detection. Although the ambient information is indeed beneficial, it is not trivial to take advantage of it, as we have shown that a simple extended voxel-based detector do not observe any improvement in detection accuracy brought by ambient information. We were able to use the ambient signals effectively by extracting ambient feature in perspective view using a multi-view segmentation-fusion framework.

Chapter 4

Object Detection with High Density Point Cloud

In Section 1.3 we described the challenge of detecting objects not well covered by point cloud, like far away objects. This problem can only be solved by introducing denser point cloud. In this chapter we propose to utilize a simulation engine and synthesize high density point cloud to address this challenge.

4.1 Background

Most popular datasets for 3D object detection, including KITTI[9], nuScenes[5] and Waymo Open Dataset[33], provides detects less than 200K points per frame on average. The latest state-of-the-art LiDAR sensor can detect up to 2.4M points per frame. Moreover, there is an ongoing trends that LiDAR sensors with increasingly high resolution are being designed and built, both by shooting laser beams more frequently during its rotation cycle and increase horizontal resolution, and by adding more laser beams and increase vertical resolution. With higher point density, far away objects and small objects will be depicted more precisely, and we will also be able to extract finer geometric features from more detailed perception of objects with complicated shapes, including pedestrians.

Because we do not have access to high-end LiDAR sensors with high density sensing capability, we utilized an autonomous driving environment simulator, Carla[8], to generate a synthetic dataset with varying point cloud density and range. This not only allows us to control the point cloud density to make it beyond current state-of-the-art LiDAR metrics, but also let us make fair comparisons by maintaining the same labels while changing the point cloud density. Visualizations of synthetic point cloud is shown in Fig.4.2 and Fig.4.3.

Utilizing the high point density, however, is non-trivial for state-of-the-art detectors. Point based detectors, including PointRCNN[29] and VoteNet[24], have a GPU memory footprint highly sensitive to the number of input points, and will not be able to handle high density point cloud without introducing architecture modifications or point sampling. With high density point cloud, we can detect faraway objects, and will increase the detection range of the state-of-the-art detectors. But for voxel-based detectors, there is not enough GPU memory to hold enlarged fea-

ture map without decreasing resolution. Thus, it is of great importance that we understand the benefit of introducing high density point cloud into state-of-the-art 3D object detection methods, and find out what modifications are required to process the dense points efficiently and effectively.

To keep GPU memory usage in bound, we introduced an input slicing technique which cuts input point cloud into multiple pieces, then perform object detection using SECOND[38] in each piece one-by-one, and combine the detections from all the pieces before applying non-maximum-suppression to deduplicate detections near slicing boundaries. Through this approach, we observed significant improvement in detection accuracy when using denser point cloud data. However, this input slicing approach has two short-comings: first, it slows down object detection process by the times equal to number of sliced pieces; second, point cloud of objects on slicing boundaries are truncated, likely leading to inaccurate detections. Therefore, we conclude that higher point cloud density is extremely helpful for accurate 3D object detection, but more research is needed in designing methods for leveraging high density point cloud efficiently and effectively.

4.2 Model Design

We use the state-of-the-art detector SECOND[38] for experiments with high-density point cloud as the GPU memory consumption of voxelized representation is rather insensitive to the number of points. However, SECOND is originally designed for KITTI[9] dataset, which only provides annotations in the front of the vehicle and in a limited range. As a result, SECOND crops the input point cloud at $[-40, 40] \times [0, 70.4]m$ along the $y \times x$ axes for car detection, and at $[-20, 20] \times [0, 48]m$ for pedestrian and cyclist detection. Our dataset provides accurate and comprehensive annotation in 120 meter range, and the whole area is covered during evaluation. Therefore, we need to change the input point cloud range to $[-120, 120]m$ along both x and y axis for detection of any class. If we enlarge the input point cloud range while keeping the original resolution, the memory consumption will increase by over 10 times for car detection, and by 30 times for pedestrian and cyclist detection, and exceed the GPU memory limit.

To address the memory issue, we introduced a point cloud slicing technique. The input point cloud data is sliced into 12 pieces, and the pieces are fed into a SECOND model one-by-one to perform 3D object detection individually, and detections from each piece are combined together. Following the original input point cloud range of SECOND model, we use a slice size of $80 \times 70.4m$ along the $y \times x$ axis. Fig.4.1 illustrates the point cloud slicing technique in birds eye view. Due to the point cloud range not divisible by the slice size, which is the original input range of SECOND model, there is overlapping area between slices. Having overlapping area will introduce more duplicate detections from the detection results on each of the overlapping slices. To remove duplicate detections on overlapping areas and near slice boundaries, we perform an additional non maximum suppression (NMS). Note that the point cloud slicing is carried out after ground truth augmentation[38], which introduces additional objects from other frames into the current frame, while maintaining their original location. This augmentation technique exploits the sparse nature of point cloud in autonomous driving context and speeds up convergence. If we perform ground truth augmentation on a slice basis, then the augmented point cloud no longer

conforms with the original object distribution. For example, the cars from less populated slices will be sampled more frequently.

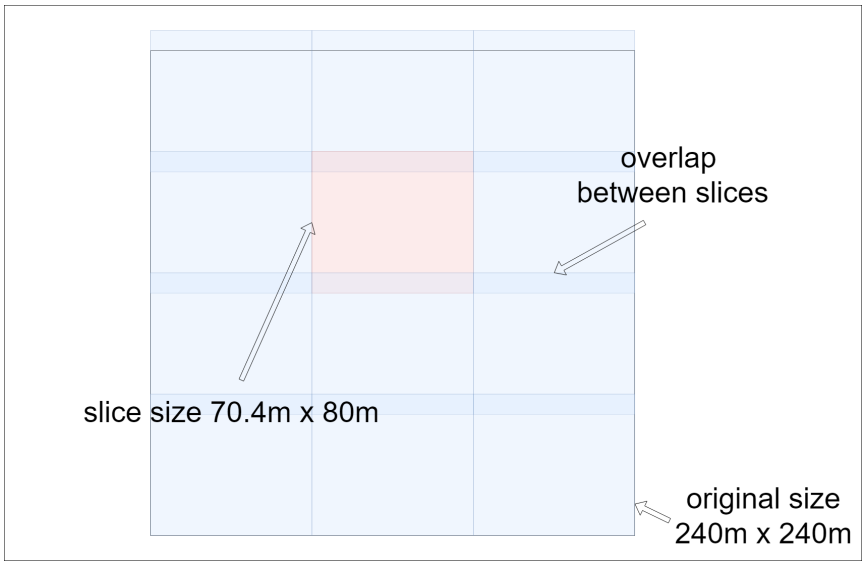


Figure 4.1: **Illustration of point cloud slicing technique.** Note that there is overlapping area between slices.

4.3 Results

4.3.1 Data Collection

Synthetic Data Generation

To acquire high density point cloud data, we generated a dataset using simulation engine. We use synthetic dataset for three reasons: firstly we do not have access to the latest LiDAR sensor with high resolution; secondly we want to go beyond the limits of state-of-the-art sensors to directly tackle the more advanced sensors in future; lastly labelling faraway objects accurately is a demanding task for human, but the simulation engines can provide us with precise ground truth labels. We used Carla[8] for dataset generation as it allows us to control every agent in the scene. Our synthetic dataset contains 250k frames with 26M labeled object instances. Object classes include cars, pedestrians, cyclists and motorcycles. Because the dataset is collected at 10 FPS, and contains a great amount of redundancy, we only selected 26043 frames for training and 8681 frames for evaluation.

For each scene in our dataset, we choose one of eight cities from Carla assets and sample locations covering the entire city to generate agents. For each agent, we set a random target destination in order to generate diverse trajectories. We randomly customize the behavior for each agent in the environment to increase the diversity of the data. Once the environment is set up, we randomly select a car as our ego-vehicle equipped with our sensor suite for data recording.

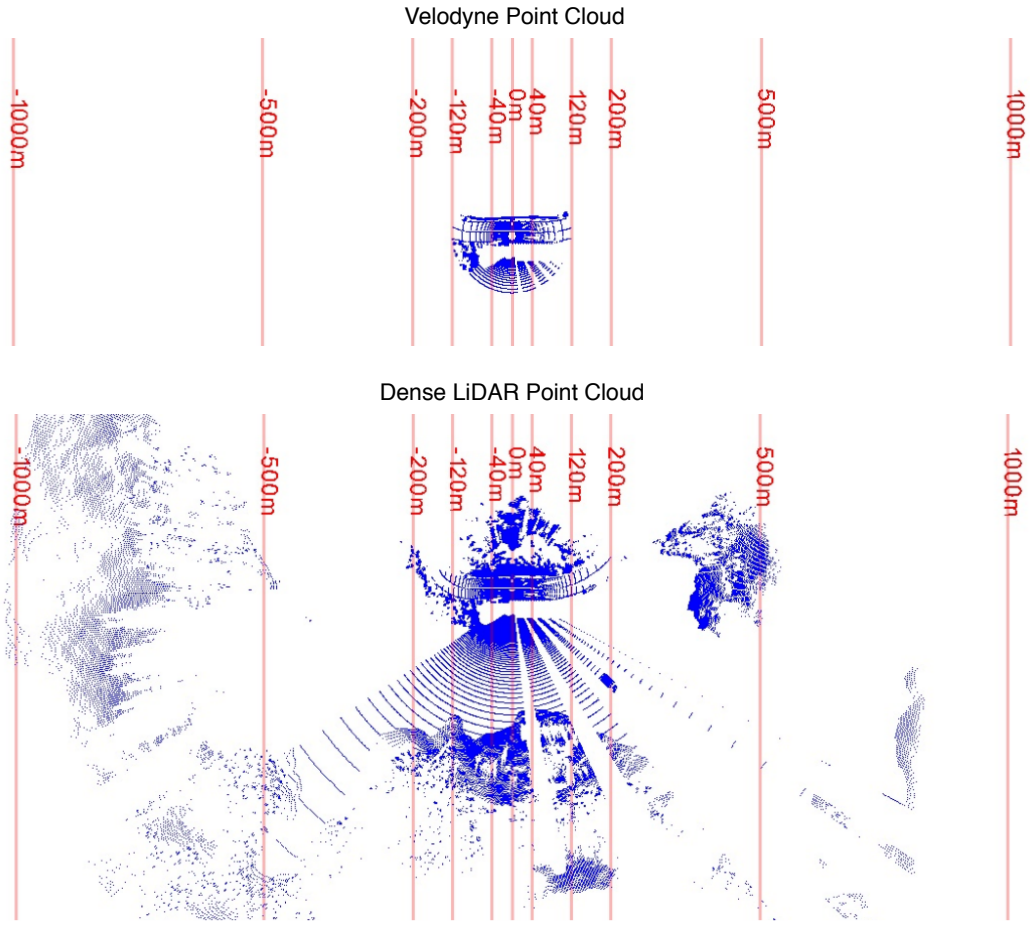


Figure 4.2: **Birds eye view visualization of synthetic dense point cloud compared with velodyne point cloud.** The dense point cloud can capture points in a much larger range.

Every 3 seconds, we also generate new agents in the empty areas of the city to further increase agent density. We have collected 250 such scenes in our dataset, each containing 1000 frames with annotation. As shown in Table 4.3.1, our dataset has the most number of annotated frames compared to other datasets.

Sensor Configuration

In our synthetic dataset, we provide two types of LiDAR sensors. One simulates the standard Velodyne LiDAR which is used in datasets like KITTI[9], which detects 100k points per frame and has a sensing range of 120m. The other detects 4M points per frame and has a sensing range of 1km. Fig.4.3 shows how the point cloud from two LiDARs look, and how the high-density point cloud can depict faraway vehicles in a more detailed way. In the visualizations, a car 80 meters away from the sensor only incurred less than 10 points using the simulated Velodyne LiDAR, while in our high density point cloud it was depicted by over 200 points. Also, a car at

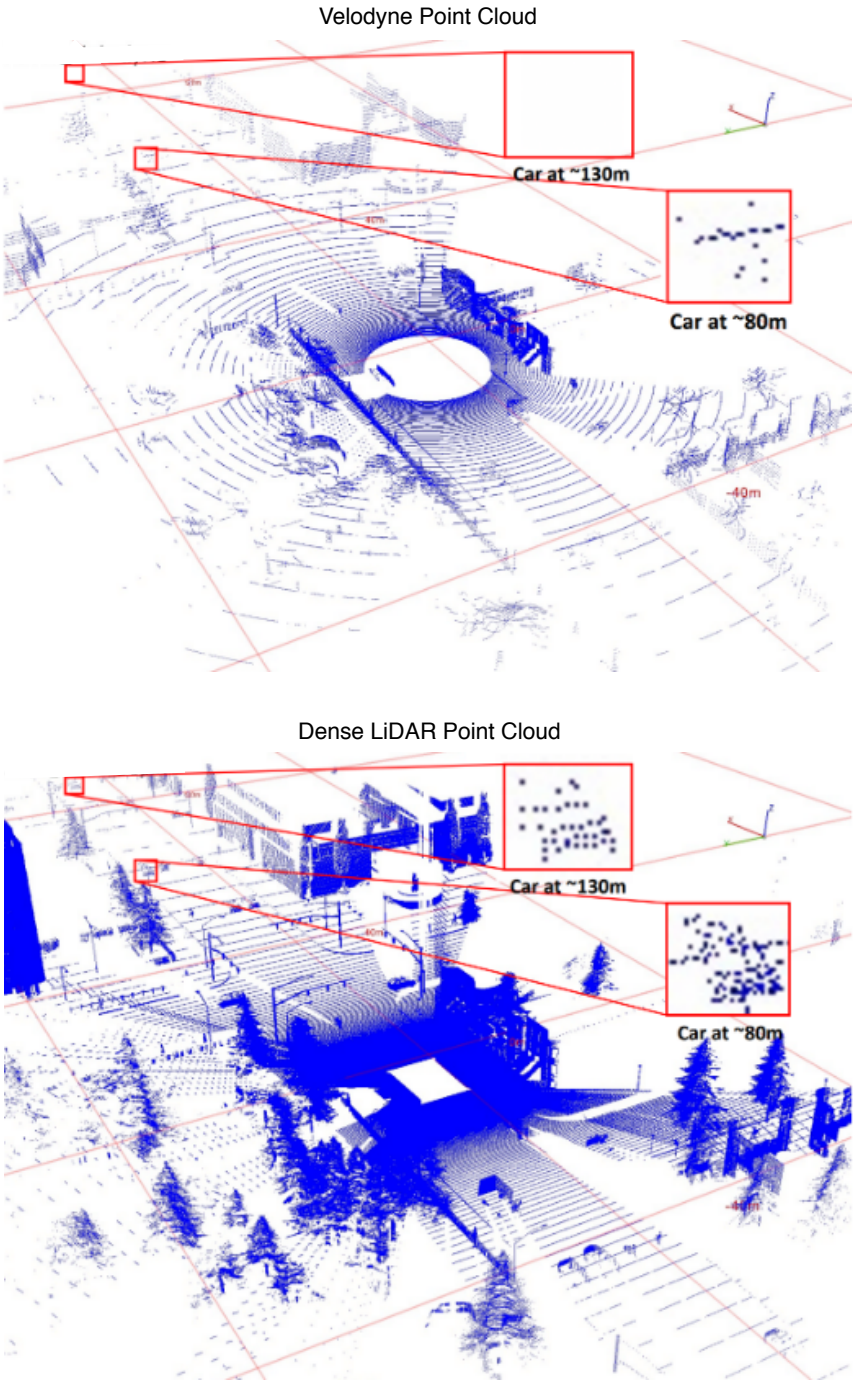


Figure 4.3: **3D visualization of synthetic dense point cloud compared with velodyne point cloud.** The dense point cloud can capture much more points than velodyne, especially for far-away objects.

130 meters range is completely missed by the regular density point cloud, but the high density point cloud is still able to capture the rough shape of the car. Fig.4.2 visualizes the different

Dataset	Cities	Hrs	Seq.	Images	Stereo	Depth	LiDAR	All 360°
KITTI [9]	1	1.5	22	15k	✓	✓	✓	
Cityscape [7]	27	2.5	0	5k	✓			
Mapillary Vistas [18]	30	-	-	25k				
ApolloScape [12, 35]	4	-	-	140k	✓		✓	
SYNTHIA [27]	1	2.2	4	200k		✓		✓
H3D [19]	4	0.8	160	27k			✓	
SemanticKITTI [1]	1	1.2	22	43k			✓	
DrivingStereo [32]	-	5	42	180k	✓	✓	✓	
Argoverse [6]	2	0.6	113	22k	✓		✓	✓
EuroCity [4]	**31**	0.4	-	47k				
CADC [21]	1	0.6	75	7k			✓	
Audi [11]	3	0.3	3	12k	✓	✓	✓	✓
nuScenes [5]	2	5.5	1k	40k			✓	✓
A*3D [20]	1	**55**	-	39k	✓		✓	
Waymo Open [33]	3	6.4	**1150**	230k			✓	
Ours	8	6.9	250	**250k**	✓	✓	✓	✓

Table 4.1: Comparison of size and sensor modalities between modern perception datasets. Note that we only count annotated images for all datasets. Our AIODrive dataset has the most comprehensive sensing modalities while being the largest in terms of the number of annotated frames.

sensing range of the two sensors. We obtain the high density point cloud by setting up five virtual cameras with depth sensing capability and a resolution of 1920×720 pointing at different directions, and projecting all five depth images to a common 3D space which forms a point cloud with each point representing a pixel in one of the depth images. The sensor layout and the coordinate systems of our synthetic dataset is shown in Fig.4.4.

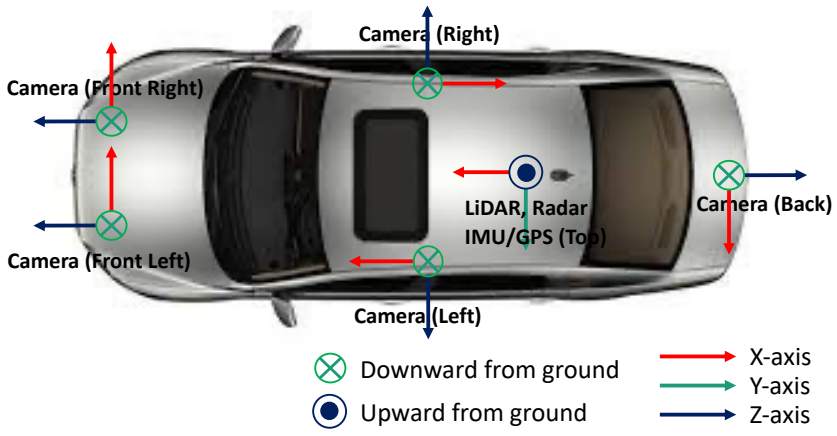


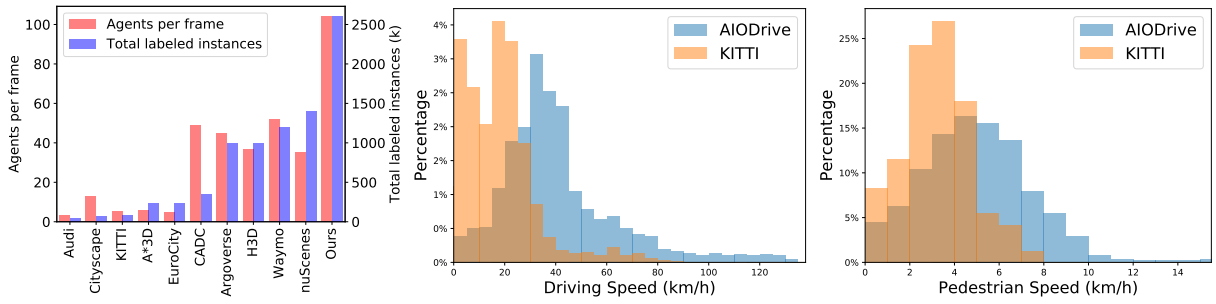
Figure 4.4: Sensor layout and coordinate systems of synthetic dataset.

Dataset	Adverse weather	Different lighting	Crowded scenes	High-speed driving	Traffic rule violation	Car accidents
KITTI [9]						
Cityscape [7]						
Mapillary Vistas [18]	✓	✓				
ApolloScape [12, 35]		✓				
SYNTHIA [27]	✓	✓				
H3D [19]			✓			
SemanticKITTI [1]						
DrivingStereo [32]	✓	✓				
Argoverse [6]			✓			
EuroCity [4]	✓	✓				
CADC [21]	✓		✓			
Audi [11]		✓				
nuScenes [5]	✓	✓	✓			
A*3D [20]	✓	✓		✓		
Waymo Open [33]	✓	✓	✓			
Ours	✓	✓	✓	✓	✓	✓

Table 4.2: Comparison of the environmental variations between modern driving datasets. Our AIODirve dataset provides the most variations with rare cases that are not present in prior datasets.

Scene Diversity

It is important for a dataset to have high environmental variations even including those rare yet challenging cases so that comprehensive evaluation can be performed and data-driven methods can be trained robust to these rare cases. However, collecting such rare examples is difficult in real world. To resolve this issue, we leverage the simulator for rare data generation and increase our variations. We compare the environmental variations between datasets in Table 4.2. Though most recent datasets have weather and lighting variations, some are limited by having too few agents, driving at a low speed and barely having examples of violation of traffic rules and car accidents. Instead, our dataset contains high variations and we can generate rare driving cases in the simulator without causing danger.



(a) High Crowdness (b) Driving Speed Distribution (c) People Speed Distribution

Figure 4.5: **Data Statistics**: (a) We compare the agent density between datasets in terms of agents per frame and total labeled agents, showing that our dataset includes many highly-crowded scenes; (b)(c) We compare the distribution of the ego-vehicle driving speed and pedestrians' speed, showing that our ego-vehicle drives at a similar speed to our daily driving, and our dataset includes more jogging and running ($\geq$5km/h) pedestrians.

Highly-Crowded Scenes. Driving in the crowd is challenging as agent interactions are complex. To solve this challenge, the first step is to collect such challenging data. To that end, our dataset is

designed to include many scenes with a high agent density. On average, we have 104 agents per frame. We show the comparison of agents per frame and total labeled instances between datasets in Figure 4.5 (a). Note that some datasets such as KITTI and Cityscape have a fewer number of agents per frame because that they only label objects in front. Although existing datasets such as H3D, nuScenes, Waymo and Argoverse also have crowded scenes and have an average of 30 to 50 agents per frame, our dataset is even twice more crowded as shown in Figure 4.5 (a). We believe that our dataset can be used to evaluate the robustness of autonomous driving models driving in the crowd.

High-Speed Driving is challenging because it causes difficulty in perception systems, and a short processing delay in the model might use up the time for planning and cause collision. To deal with the challenge, we need to collect data under a high speed and design models robust to such challenging data. However, modern datasets are limited by capturing data at a low speed (e.g., nuScenes at 16km/h on average) which hampers the model design. We argue that this low driving speed differs significantly from our daily driving which is 30 to 60km/h on local road and 80 to 120km/h on highway. To bridge this gap, we drive at a similar speed to our daily driving. In Figure 4.5 (b), we show the distribution of our ego-vehicle driving speed, which covers the speed ranging from 0 to 130 km/h.

Adverse Weather and Lighting can affect the accuracy of the perception systems. Therefore, it is important to include data under different weather and lighting conditions. In our dataset, we support different weathers such as sunny, rainy, cloudy, windy and foggy. Also, we simulate the sun rising and falling process so that we have data collected in morning, noon, afternoon, dusk and at night.

Other Environmental Variations. Besides high speed driving and crowded scenes, we also provides other data such as car accidents, vehicles that run over the red light, speed over the limit and change the lane frequently, children and adults jogging and running on the sidewalk. Though these cases happen in our daily life, they barely exist in prior datasets. To build robust perception systems, we believe it is important to include these variations in the training and evaluation set. As one example, we show the pedestrian speed distribution in Figure 4.5 (c), which contains jogging and running cases.

4.3.2 Implementation Details

Implementation details. We train two detectors, one for car detection and one for detecting pedestrians, cyclists and motorcycles. For training the car detector, We use voxel size of $v_z = 0.1m, v_y = 0.05m, v_x = 0.05m$, and extract feature of each voxel from up to 5 points inside. According to the ground truth size distribution, we use anchors of dimension $w = 1.6m, l = 3.9m, h = 1.65m$ for car detection, $w = 0.6m, l = 0.8m, h = 1.73m$ for pedestrian detection, $w = 0.79m, l = 2.2m, h = 1.5m$ for motorcycle detection, and $w = 0.6m, l = 1.76m, h = 1.73m$ for cyclist detection. Before point cloud slicing, we sample 100 cars from a pre-built ground truth database for augmentation in car detection training. For training the people detection model, we sample 100 objects of classes pedestrian, cyclist and motorcycle respectively. We train both detection model for 1,111,420 steps with batch size 2. Considering each traning frame consists of 12 slices and takes 6 steps, this amounts to training for 185,700 samples, which is approximately 7.1 epochs. Other implementation details are the

same as in Section 2.3.

Evaluation Metrics. Because we do not have trucks and buses in our synthetic dataset, during car detection evaluation, we do not ignore these classes. Other evaluation metrics on the synthetic dataset follow those in Section 2.3.

4.3.3 Experiment Results

Table 4.3 shows the detection accuracy of the detectors with regular point cloud and high density point cloud as input respectively. With high-density point cloud, the detection accuracy is greatly improved, especially under higher IoU thresholds. This means the bounding box localization is more precise with higher point cloud density.

In this chapter We have shown the improvement in 3D object detection accuracy brought by high density point cloud. Although we were able to utilize this enriched information by introducing a point cloud slicing technique, this approach suffers from two flaws. Firstly, as the point cloud of each frame is now sliced into 12 pieces and are processed individually, the training and inference speed is 12 times slower than SECOND's original speed. This is not acceptable in real world autonomous driving scenario as it is no longer real time. Secondly, the slicing introduces unnecessary point cloud truncation, and might lead to inaccurate detections. More research is needed for designing model structures that can leverage high density point cloud efficiently and effectively.

class		regular point cloud	high density point cloud
car	AP@0.7	39.55	70.39
	AP@0.5	52.37	78.14
pedestrian	AP@0.5	25.98	53.69
	AP@0.25	42.63	61.02
cyclist	AP@0.5	26.27	52.54
	AP@0.25	41.95	60.86
motorcycle	AP@0.5	29.75	59.84
	AP@0.25	44.89	61.68

Table 4.3: Performance of 3D pedestrian detection with high density point cloud.

4.4 Conclusion

In this chapter, we investigated the benefit of high density point cloud in 3D object detection. We acquired the data using a simulation engine, and proposed a point cloud slicing technique to run a voxel-based detection method on the high density point cloud while keeping GPU memory in bound. We observed significant improvement in detection accuracy brought by the high density point cloud.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this work, we investigated the influence of enriched point cloud on 3D object detection. Our findings are three-fold: Firstly, by introducing multi-echo point cloud produced by single-photon LiDARs with increased fill factor, we significantly boosted the detection accuracy of state-of-the-art detectors, particularly improving the recall of the detector. Secondly, with ambient signal captured along with the points, we noticeably improved pedestrian detection accuracy by proposing a multi-view segmentation-detection fusion framework to take full advantage of the ambient information. Thirdly, we utilized a simulation engine to generate a simulation dataset with high density point cloud data, and greatly improved detection accuracy while keeping GPU memory footprint down by introducing a point cloud slicing technique.

Although the we have successfully leveraged enriched point cloud using our proposed model designs, these designs have their merits as wells as weaknesses. In our proposed multi-view segmentation-fusion framework, even though we successfully exploited the benefits of ambient information by extracting ambient features from perspective view and fusing the feature into 3D detectors, this approach suffers from an internal objective inconsistency problem in that the segmentation objective puts less focus on the faraway objects, while detection evaluation metrics treat every instance the same. This problem has caused pedestrian detection accuracy on objects in far range to degrade. The point cloud slicing technique also suffers from low training and inference speed as well as unnecessary point cloud truncation leading to inaccurate detections along the slicing boundary.

In summary, we have successfully demonstrated the benefit of leveraging enriched point cloud data in 3D object detection, yet our proposed model designs have drawbacks and can be improved.

5.2 Future Works

Although we were able to effectively utilize the enriched point cloud information, our models have their merits as well as drawbacks, and we believe more research is required to improve their design or investigate other options.

In our multi-echo point cloud based detection framework, the points detected by the same beam are treated as individual points. We discard the same-beam point ordering information, which potentially convey information regarding the points. For example, ordering by distance present information about the points in the front that they are likely on object boundaries or semi-transparent surfaces, since they only cause a partial reflection. To utilize the ordering, we can simply use the ordering index as an extra point property and feed the point cloud with indexes to an extended detector. However, if we use a voxel-based detector, the points from the same-beam are naturally apart. To ease intra-beam point reasoning, we may also try to present multi-echo point cloud in perspective view, where each pixel has three slots to placed ordered points, and directly perform detection on this multi-slot range image.

In our proposed multi-view segmentation-fusion framework, there is an objective inconsistency issue in that, segmentation loss penalizes on pixel level, but detection evaluation is carried out on instance level. To address this problem, we will need to distribute the segmentation loss weight, to make the pixels that each instance occupies sum up to the same amount of loss weight. Alternatively we can back-propagate the gradients from detection network to the 2D convolutional segmentation network, and train two networks end-to-end.

Our proposed point cloud slicing technique introduces unnecessary point cloud truncation, and suffers from high computation time. To address these problems while keeping GPU memory in bound, we may try to build upon point-based detection methods, and only select a certain amount of most important points from the high density point cloud using farthest point sampling. Farthest point sampling can effectively reduce the number of points while keeping space coverage as high as possible.

Aside from proposing more effective approaches, we believe current research in 3D object detection is limited by the capabilities of LiDAR sensors equipped in our contemporary datasets. Our findings call for more advanced datasets and benchmarks that include high end LiDAR sensors with enriched point cloud sensing capabilities.

Bibliography

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. *ICCV*, 2019. 4.3.1, 4.3.1

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019. 1.2.2, 3.2.2, 3.3.2, 3.3.3

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *arXiv:1604.07316*, 2016. 1.2.3

[4] Markus Braun, Sebastian Krebs, Fabian Flohr, and Dariu M. Gavrila. The EuroCity Persons Dataset: A Novel Benchmark for Object Detection. *TPAMI*, 2019. 4.3.1, 4.3.1

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, and Qiang Xu. nuScenes: A Multimodal Dataset for Autonomous Driving. *CVPR*, 2020. 1.3, 4.1, 4.3.1, 4.3.1

[6] Ming-fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, B Sławomir, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3D Tracking and Forecasting with Rich Maps. *CVPR*, 2019. 4.3.1, 4.3.1

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *CVPR*, 2016. 4.3.1, 4.3.1

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. *CoRL*, 2017. 1.2.3, 1.4, 4.1, 4.3.1

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *CVPR*, 2012. 1.3, 4.1, 4.2, 4.3.1, 4.3.1, 4.3.1

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2.3.2

[11] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian

Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. *arXiv:2004.06320*, 2020. 4.3.1, 4.3.1

[12] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The ApolloScape Dataset for Autonomous Driving. *CVPRW*, 2018. 4.3.1, 4.3.1

[13] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1.2.1

[14] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 3.2.2

[15] M Matthias, Casser Jean, Lahoud Neil, and C V Mar. Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications. *IJCV*, 2018. 1.2.3

[16] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12677–12686, 2019. 1.2.1

[17] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. (document), 3.2

[18] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. *ICCV*, 2017. 4.3.1, 4.3.1

[19] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes. *ICRA*, 2019. 4.3.1, 4.3.1

[20] Quang-hieu Pham, Pierre Sevestre, Ramanpreet Singh Pahwa, Huijing Zhan, Chun Ho Pang, Yuda Chen, Armin Mustafa, Vijay Chandrasekhar, and Jie Lin. A*3D Dataset: Towards Autonomous Driving in Challenging Environments. *ICRA*, 2020. 4.3.1, 4.3.1

[21] Matthew Pitropov, Danson Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian Adverse Driving Conditions Dataset. *arXiv:2001.10117*, 2020. 4.3.1, 4.3.1

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017. 1.2.2

[23] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NeurIPS*, 2017. 1.2.1, 1.2.2

[24] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on*

Computer Vision, pages 9277–9286, 2019. 1.2.1, 4.1

[25] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 3.2.2

[26] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for Benchmarks. *ICCV*, 2017. 1.2.3

[27] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. *CVPR*, 2016. 4.3.1, 4.3.1

[28] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *Field and Service Robotics*, 2017. 1.2.3

[29] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1.2.1, 4.1

[30] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng 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 (CVPR)*, June 2020. 1.2.1

[31] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019. 3.2.2

[32] Xiao Song, Chaoqin Huang, Zhidong Deng, Jianping Shi, and Bolei Zhou. DrivingStereo: A Large-Scale Dataset for Stereo Matching in Autonomous Driving Scenarios. *CVPR*, 2019. 4.3.1, 4.3.1

[33] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. *arXiv:1912.04838*, 2020. 1.3, 2.1, 4.1, 4.3.1, 4.3.1

[34] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4604–4612, 2020. 3.2.2

[35] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape Open Dataset for Autonomous Driving and its Application. *TPAMI*, 2019. 4.3.1, 4.3.1

[36] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 1.2.2

[37] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382. IEEE, 2019. 1.2.2

[38] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. (document), 1.2.1, 2.1, 2.2, 2.1, 2.3.2, 3.1, 3.2.1, 3.2.2, 3.3.3, 3.3.3, 4.1, 4.2

[39] 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*, pages 4490–4499, 2018. 1.2.1

[40] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932, 2020. 1.2.1