Domain Adaptation for 3D Detection in Autonomous Vehicle Perception : An Empirical Study

Divyanshu Tak
Department of Electrical and Computer Engineering
The Ohio State University
Columbus, OH 43221
tak.22@osu.edu

Mengdi Fan
Department of Chemical Engineering
The Ohio State University
Columbus, OH 43221
fan.825@osu.edu

Chaeun Hong
Department of Computer Science
The Ohio State University
Columbus, OH 43221
hong.930@osu.edu

Abstract

Domain adaption is a vigorous topic in 3D detection due to the mismatch problem of LiDAR sensors. Different sensors have different scan patterns because of their mechanics of beam acquisition, which results in the generation of different point clouds of the same scene. Our study aims to investigate this phenomenon and estimate the performance of model when trained and tested on different LiDAR datasets. The datasets used forthis study are KITTI and CMU AIO dataset.

1 Introduction

With the recent developments in deep neural networks, 3D detection with point clouds has become a much more feasible problem. This has led to the development of numerous popular 3D detection algorithms like PointPillars,[1] Point RCNN,[2] PointNet[3], etc. All the popular 3D detection frameworks on the data captured with LiDAR as the primary sensor modality. This rise in research and exploration of 3D detection algorithms has also concurrently led to emerging technologies for LiDAR sensors, especially in the domain of autonomous navigation like self-driving cars, and autonomous rovers and robots. Currently, there are 70 different LiDAR companies in the market, each company with its unique and private technology to capture the scene using the laser beams. This results in a huge variety of LiDAR point clouds with different point cloud densities and scan patterns. This variety naturally creates a sensor mismatch problem which falls under the general umbrella of domain adaptation problems in 3D detection.

Our study aims to analyze this problem by deeply looking into the scan patterns of different LiDAR point clouds and estimating the model performance of 3D detection models on different datasets. Another interesting question that comes up in this direction is, how different 3D detection model performs when training data and testing data come from different LiDAR sensors. Detection performance is good when train and test data come from the same modality, but the performance gets affected when testing data comes from a different sensor. For example, a model trained on LiDAR data with 32 beams performs badly on testing data with 16 beams, but a model trained on LiDAR sensor data with 32 beams performs well on testing data containing 64 beams. This is an interesting finding which forces us to rethink the 3D detection model's performance and how we can leverage data from different modalities to make sure that 3D detection performance is not affected when modalities change during testing and training.

	Lidar sensor
KITTI	1 Velodyne(64 channel, 10Hz, 100k points/frame)
Apollo Scape	2 Velodyne(64 channel, 10Hz)
nuScenes	1 Velodyne(32 channel, 10Hz)
Lyft Level 5	1 (40 channel) + 2 aux.(40 channel)
Waymo OD	1 + 4 aux.(64 channel)
A2D2	5 (16 channel, 10Hz)
H3D	Velodyne (64 channel)
DrivingStereo	Velodyne (64 channel)
Argoverse	2 (10Hz)
CADC	1 (32 channel, 10Hz)
A*3D	1 Velodyne (64 channel)
CMU AIODrive	1 Velodyne (64 channel, 10Hz, 2.2M points/s) + 2 dense (800/1280 channel, 10Hz, 10/16M points/s)

Figure 1: 3D point cloud datasets along with their employed LiDAR sensor charectersticks

Figure 2: Point cloud visualization for different datasets

For training and testing of 3D detection models, a lot of datasets have been released recently like KITTI,[4] nuScenes,[5] and Waymo[6] datasets (Figure 1). These datasets contain the LiDAR point clouds with different beam densities along with annotations and 3D bounding boxes (Figure 2). These are real-world datasets as they are generated by driving the car in different rural and urban scenarios. Another category of the dataset is the synthetic dataset in which the data is collected in a high fidelity simulated environment. One of the most popular datasets that falls into this category is the CMU AIO Drive dataset.[7] This dataset is generated by running an autonomous agent in the CARLA simulator, with the sensor setup and data format resembling that of the KITTI dataset. The CMU AIO drive dataset provides RGB, Depth images of 5 different FOVs and LiDAR point clouds of 3 different densities. In our study, we leverage the depth image and LiDAR data provided in this dataset for the purpose of point cloud data conversion.

Apart from the publicly available datasets, there are also other LiDAR data from different LiDAR vendors like Cepton, which can also be leveraged for estimating the performance of popular 3D detection models. One problem that arises with using the data from newly developed LiDAR sensors like Cepton P60, is the lack of annotation and bounding boxes. This is because this data is collected manually and the vendor does not provide an annotated dataset. Our work includes the development of an algorithm to leverage cepton data and annotations from CMU AIO Drive into a one annotated dataset. Overall the contribution of our study can be summarised as follows:

- In depth analysis of different LiDAR scan patterns.
- Development of an algorithm to convert from Cepton point cloud to Velodyne-like point cloud.
- Development of an algorithm to generate Cepton-like point cloud from depth map.
- Development of annotated Cepton-like point cloud dataset.
- Analysis and results of 3D detection models on different training and testing data modalities.

2 Related Work

2.1 3D Object Detection Algorithm

3D object detection aims to detect and classify objects using LiDAR point clouds. Different strategies have been explored in the current study of 3D object detection algorithms. One way is to project the 3D point cloud to bird's eye view (BEV) or front view to extract features from 2D maps using 2D image detection methods, like MV3D[8]–[10]. Another road is to divide point clouds into voxels and detect them using 3D convolutions, like VoxelNet and PointPillar[1], [11], [12]. Also, there are several popular algorithms directly detect objects from point cloud via 3D convolutional networks to avoid geometric information loss in representation transformation, including PointNet[3], PointNet++[13], Frustum PointNet[14] and PointRCNN[2].

2.2 Unsupervised Domain Adaptation

As the target domain may have limited labeled training data, domain adaptation aims to adapt models trained on the source domain to work well on the target domain. Most of the current 3D domain adaptation studies focus on the dataset from different collecting conditions, like different cities and weather,[15]–[17] though another interesting topic is the different LiDAR model. Wang et al. state the domain gap between KITTI (collected in Germany) and other datasets collected in the USA can be the car size and propose SN to normalize the object size to overcome the domain gap between source and target domain[15]. Xu et al. address the point cloud quality as the key factor between different Waymo point clouds and propose SPG to help solve the domain shift.[16] Recently, Yang et al. develop ST3D, a pipeline involving random object scaling, a quality-aware memory bank, and data augmentation to eliminate the domain gap in 3D object detection using LiDAR point cloud.

2.3 Point Cloud Transformation

One way to solve point cloud domain mismatch is to randomly down-sample or up-sample the source domain to match the density of the target domain.[18]–[20] Or adapting the style transfer methods to generate 3D point clouds.[21]–[23]

3 Approach

To understand the sensor mismatch problem, the first step is to understand how a point in the LiDAR data can be viewed in a 3D space. Since, the LiDAR works on the time of flight principle of a laser beam hitting the target and returning to the receiver, a point in the LiDAR point cloud data is stored in the form of X, Y, Z, and R. Here the X, Y, and Z coordinates reflect the position of the target in the Cartesian frame and R represents the reflectivity of the target surface. we understood that Cartesian representation of a point restricts the freedom with which it can be analyzed in a 3D space. For this reason, we transformed each point from Cartesian coordinate to spherical coordinate as seen in the 3

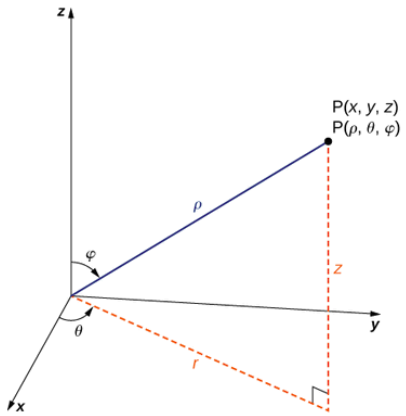

Figure 3: Point in a 3D space with different coordinate systems

This conversion allows us to see how a point can also be referenced using (θ, ϕ, r). From 4 it can be observed that for two points at a different depth from the origin, it is possible to have the same θ value with different Φ. Similarly, for two points at a different depth from the origin and different θ, it is possible to have the same Φ. In other words, it can be said that two points in a point cloud with different depths will have the same Φ and θ. This is a very crucial observation because this allows us to decouple the scene invariant mask of a point cloud from the scene-dependent information. This scene invariant mask is essentially what represents a LiDAR scan pattern. This mask when mixed with the depth of each point, represents a 3D point cloud.

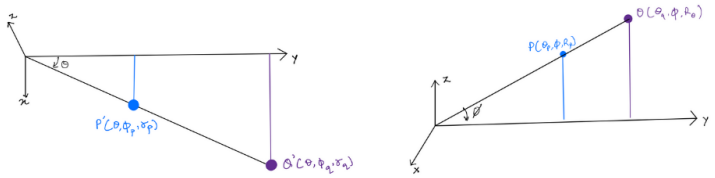

Figure 4: Theta and Phi invariance of points in 3D space. (Left) top view for theta, (right) side view for phi

Using the above-mentioned observations, we extracted the scan patterns of different LiDARs, specifically Velodyne-64 and Cepton. Each point in the scan pattern scatters plot represents a point's (Φ, θ). This scan pattern essentially represents the scan properties of LiDAR, capturing how the LiDAR captures the reflected beams. The Figure 5 displays the scan pattern of the two different LiDAR point clouds, Cepton and Velodyne-64. As it can be observed that the Cepton scan pattern consists of distinct overlapping leaf patterns, which can also be observed in its point cloud visualization. Similarly, the Velodyne point cloud scan patterns consist of discreet lines of points which are also present in its point cloud data visualization. This observation validates our concepts of generation and characterization of LiDAR point clouds.

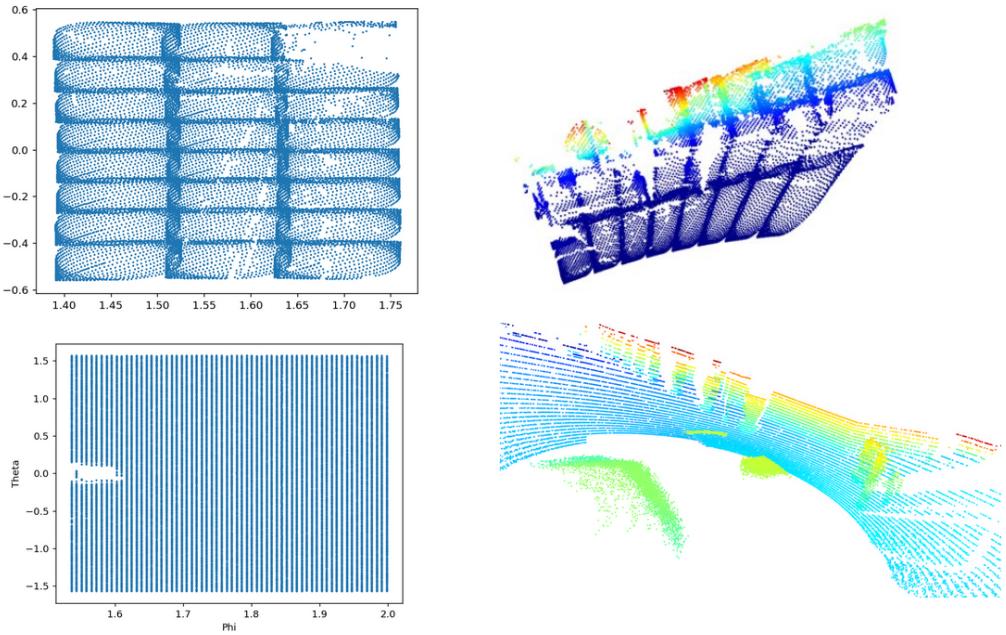

Figure 5: Scan Pattern (top left) of the Cepton LiDAR point cloud (top right). Scan Pattern (bottom left) of the Velodyne LiDAR point cloud (bottom right).

Now, that we can generate and characterize a LiDAR point cloud's scan pattern, we can compare different LiDAR point clouds based on their scan pattern. Since each LiDAR has its own horizontal and vertical Field Of View (FOV), this can limit the overlapping range of the two separate point

clouds. This can also be observed in the Figure 6. It can also be observed from the zoomed version that the subset of points of both the scan pattern is really close to each other, this observation gives motivation to our next approach which is the concept of nearest neighbor sampling from two scan patterns.

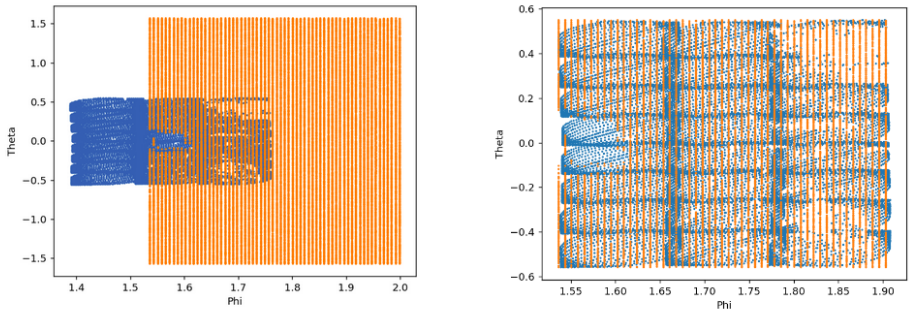

Figure 6: Overlapping scan pattern of Velodyne and Cepton LiDAR.

Since each point in the scan pattern represents a particular point in the point cloud but in a 2D scan space. As the subset of points in the scan pattern is extremely near in terms of L2 distance, our methods aim to leverage this spatial relation by sampling the points based on the nearest neighbour algorithm. The Figure 7 displays the result of performing the nearest neighbour sampling on the Cepton scan pattern by taking the Velodyne scan pattern as the mask. Taking a scan pattern as a mask means that each point in the original scan pattern is compared to each point in the mask based on the l2 distance. It can be observed that now the Cepton pattern resulting after nearest neighbour sampling does not contain the overlapping leaves which were present in the original scan pattern, rather it contains the distinct lines like the Velodyne scan pattern. From this, we can say that we have converted the Cepton scan pattern to Velodyne-like scan pattern. From the plot below there are a few observations that can be made. First, the density of lines is not uniform as was the case in the original Velodyne scan pattern. Secondly, the original scan pattern is uniform, which is clearly evident by the blank patches in the Cepton scan pattern, this non-uniformity leads to poor quality of conversion. we aim to address these problems using the method called "Masking".

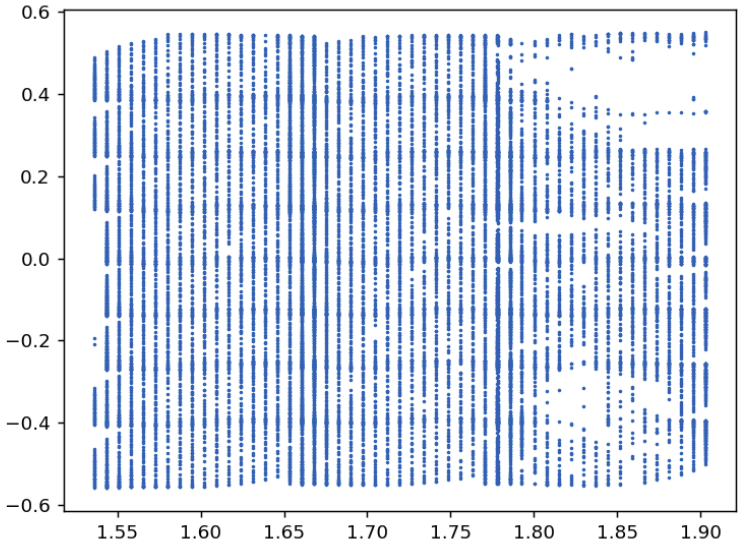

Figure 7: Output scan pattern after nearest neighbour sampling.

3.1 Masking

To address the problem of nonuniformity in the original scan pattern and sampled scan pattern, we develop an approach called masking. This involves extracting a singular leaf from the original scan pattern and replicating each leaf in horizontal and vertical axes including an overlap between leaves. This overlap parameter can be controlled to make the mask resemble the original Cepton scan pattern. As it can be observed from 8 the resulted Cepton mask is much more complete and uniform compared to the original Cepton mask. This resultant mask provides a solid reference to perform nearest neighbour sampling on different LiDAR point clouds.

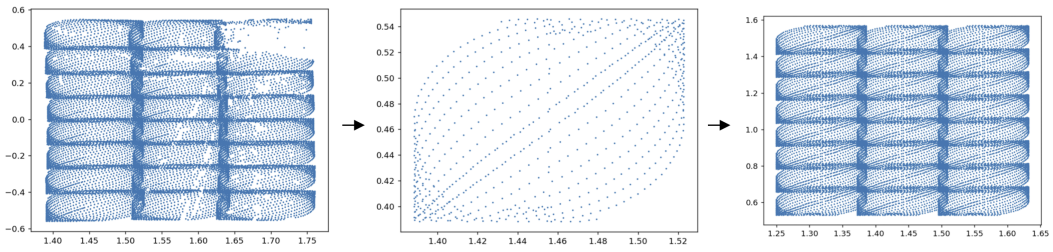

Figure 8: Generation of Cepton Mask.

Now that we have a uniform and complete mask, the next step is to generate the dense and complete point cloud to perform the nearest neighbour sampling. This is achieved by employing the depth map provided by the CMU AIO Drive dataset. The depth map provides pixel-wise depth in a particular scene. Our approach involves converting the depth map into a 3D point cloud using the camera intrinsics and FOV range. This resulted point cloud is dense compared to the original Cepton and Velodyne-64. This can be observed in Figure 9 below. This dense point cloud provides a good reference for the point clouds conversion using nearest neighbour sampling.

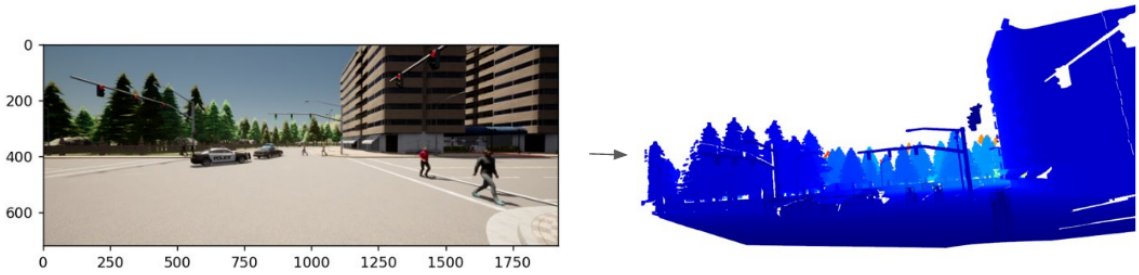

Figure 9: Depth map to Dense Completed point cloud.

3.2 Completed/Dense point cloud to Cepton Alike point cloud

As mentioned in the previous section, the depth map and Cepton mask provide a good reference for performing point cloud conversion. In this section, we mention our algorithm for point cloud conversion from dense/completed point clouds which is generated from a depth map to the synthetic/Cepton alike point cloud.

This algorithm can is also visualized in the Figure 11 below. The depth map is first converted to a dense point cloud, once converted we extract the scan pattern of the dense point cloud and use the Cepton mask to perform the nearest neighbour sampling. This results in a Cepton alike point cloud.

3.3 Dataset Generation

As directed by the abstract, the main motivation of our study is to test and evaluate the 3D object detection models on different LiDAR modalities. This requires training and testing of these models on different LiDAR point clouds with bounding box annotations. For this purpose, we create our own

Algorithm 1: Cepton Point Cloud Generation from Completed lidar Point Cloud

Input : Completed_point_cloud, generated from depth map
Cepton_mask

Output: Ceton-alike point cloud

Load Cepton_mask
Initialize K=1, Treshold = 0.006
Initialize NN_object = Nearest Neighbour(K, Treshold)
NN_object.fit(Cepton Mask) // fit mask to the Nearest Neighbour tree
For *each point ∈ Completed_point_cloud*
    Calculate (θ, ϕ)
    L2_dist = NN_object.kneighbour([θ, ϕ]) // do the Nearest Neighbour search
    If *L2_dist<=Treshold:* // save the point if L2 distance is less than treshold
        Inverse project (θ, ϕ, r) to (x,y,z)
        Append sampled_point_cloud with (x,y,z)
Save sampled_point_cloud

Figure 10: Algorithm for conversion of dense point cloud to cepton alike point cloud.

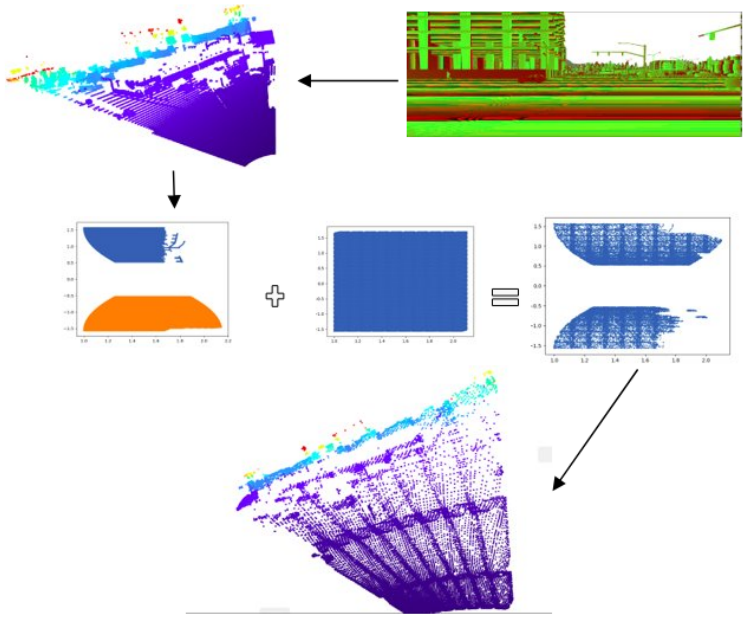

Figure 11: Algorithm for conversion of dense point cloud to cepton alike point cloud.

dataset which contains three point cloud data and their bounding box annotations. These three point cloud data consist of a Cepton point cloud, dense/completed point cloud, Velodyne-64 point cloud. The dataset contains 219 frames for each type of the point cloud.

This dataset is created by taking the CMU AIO drive dataset as the baseline reference. CMU AIO drive dataset is classified as a class of synthetic dataset since it is collected in a high fidelity simulator instead of setting up sensors on a car and driving around in different environments. The CMU AIO Drive dataset is generated with a similar sensor setup and data format as the KITTI dataset. The CMU AIO drive is a comprehensive dataset with multiple density point clouds, bounding box annotations,

tracking information, the depth map, and RGB images. Hence making it a perfect baseline for our study.

Our generated dataset contains point clouds that are aligned to the same frame of reference, and match the FOV of the associated camera frame. The bounding box annotations are also formatted into the KITTI format, to make the dataset transferable and easy to train on different 3D object detection models. The Figure 12 displays all the point cloud formats in the dataset.

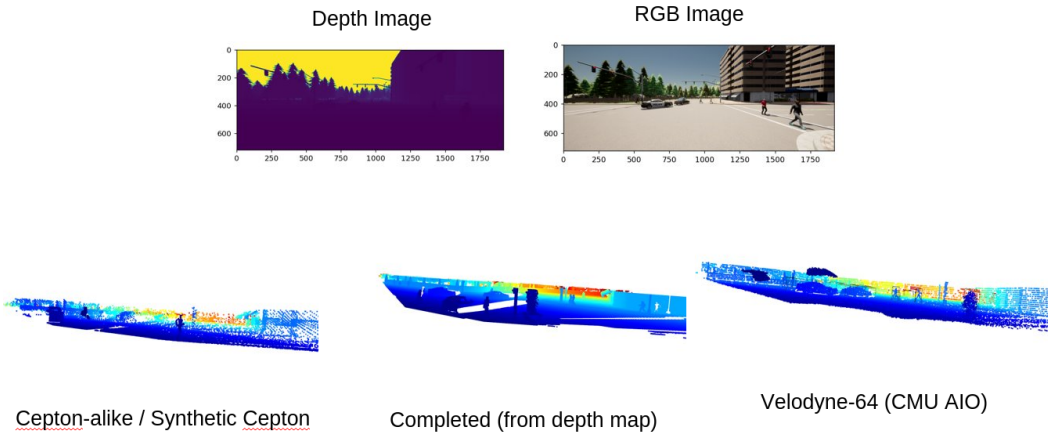

Figure 12: Visualization of the FOV dataset

4 Experiments

We use the open-source package OpenPCDet[24] to train 3D detection models and compare the performance of these models trained on different point clouds we generated (Velodyne-64, Cepton-alike, and Completed). OpenPCDet provides support for various LiDAR-based 3D object detection algorithms, including PointRCNN and PointPillar. We use PointPillar[1] as the baseline for our comparison in this study. Specifically, we split 219 frames of generated three point clouds into 175 frames of training and 44 frames of testing, and train three detection models based on Velodyne-64, Cepton-alike, and Completed point clouds, respectively. The evaluation metric we use is the Average Precision (AP) with IoU threshold of 0.7 for cars and 0.5 for pedestrians. We evaluate the models by the 2D bounding box, bird-eye view, and 3D bounding box. Since we only have a limited amount of training data, the overall AP values are not high for all models.

Table 1, 2, 3 list the results for model trained on Velodyne-64, Cepton-alike, and Completed, respectively. We find that generally, all models have the best performance when tested on validation data from the training dataset. The model trained on Velodyne-64 has better performance on the Cepton-alike point cloud than the Completed point cloud. One reason can be the similar point density between Velodyne-64 and Cepton-alike point clouds. The model trained on Cepton-alike performs similarly for three point clouds. The model trained on Completed has better AP on Cepton-alike point cloud.

To intuitively visualize the performance of the three models, we use a heat map to compare the AP difference of the 3D bounding box on Car at IoU of 0.7, shown in Figure 13. Clearly, the AP along diagonals are the highest among all nine results, indicating models are the best when testing data from the same training point cloud. As expected, the model train on the Cepton-alike point cloud has the overall best performance among the three models. On one hand, since the Cepton-alike point cloud is generated from the Completed point cloud, they share some data information similarities. As a result, the model train on Cepton-alike has quite a nice performance on Completed point cloud compared to other scenarios. On the other hand, the Cepton-alike point cloud has a similar point density as Velodyne-64 compared to Completed which led to relatively acceptable results when the model train on Cepton-alike was tested on the Velodyne-64 point cloud. Meanwhile, all three models have relatively high AP when tested on Cepton-alike point clouds due to the same reasons. The lowest AP is the model trained on Velodyne-64 tested on the Completed point cloud, which is a result of the high uncorrelation between these point clouds.

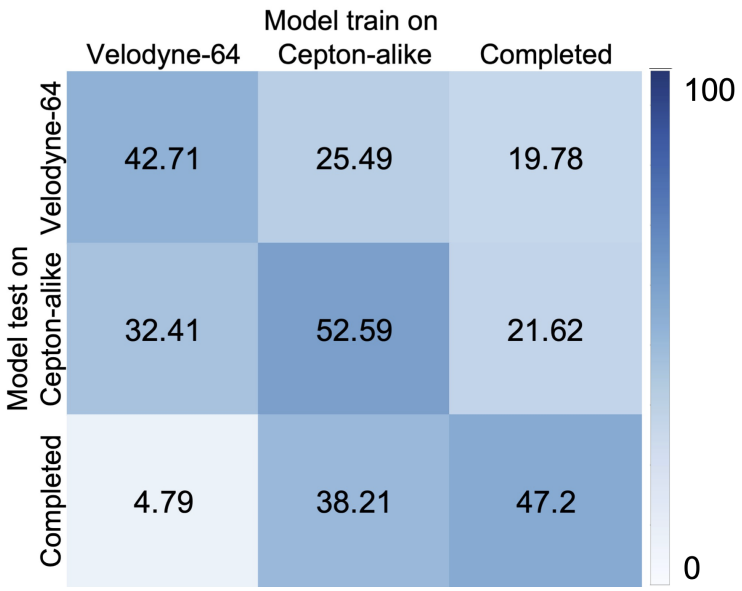

Figure 13: Compare three models: AP of 3D bounding box on Car at IoU of 0.7

Table 1: Performance of detection model trained on Velodyne-64 point cloud

Test on Velodyne-64						
AP	Car(IoU=0.7)			Pedestrian(IoU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	22.92	27.47	24.76	85.54	74.60	74.63
bev	51.40	57.94	58.22	79.32	66.77	65.22
3D	42.71	46.96	47.45	73.30	55.62	55.25
Test on Cepton-alike						
AP	Car(IoU=0.7)			Pedestrian(IoU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	18.17	22.44	28.45	84.65	75.14	75.24
bev	39.21	48.35	48.54	80.14	71.45	71.56
3D	32.41	37.07	37.09	70.02	57.79	57.92
Test on Completed						
AP	Car(IoU=0.7)			Pedestrian(IoU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	3.89	15.51	16.63	48.11	47.41	47.68
bev	5.29	17.50	18.01	41.90	43.40	43.55
3D	4.79	13.92	14.28	32.93	30.89	31.37

5 Conclusion

In this project, we use the CMU AIO drive dataset and PointPillar 3D detection algorithm as a baseline to build models trained based on the different types of point clouds and compare the performance of these models. Specifically, we generate the Completed point cloud from images and depth maps of the AIO dataset, and then develop an algorithm to create a Cepton scan pattern mask and generate Cepton-alike point clouds from the Completed point cloud. Alongside with original Velodyne-64 point cloud from the AIO date, we have three different types of point clouds. We then cropped the FOV point clouds and set the height of all point clouds to the same. Using OpenPCDet, we are able to train three models based on three point clouds. Overall, all models have the best performance when tested on data from the same training dataset. The Cepton-alike model has the best performance as the Cepton-alike point cloud has similarities with both Completed and Velodyne-64. However, the model train on Velodyne-64 has the worst AP result when tested on the Completed point cloud.

Table 2: Performance of detection model trained on Cepton-alike point cloud

Test on Velodyne-64						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	20.95	23.71	20.83	80.75	68.69	68.75
bev	37.72	45.87	44.55	75.19	57.74	57.82
3D	25.49	29.09	27.50	60.31	44.31	44.17
Test on Cepton-alike						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	22.47	30.70	32.54	91.38	78.25	78.30
bev	55.93	67.09	67.26	84.43	74.80	74.90
3D	52.59	55.87	56.61	83.10	63.95	64.29
Test on Completed						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	22.49	32.99	32.79	86.55	75.49	75.55
bev	45.71	56.48	55.20	85.89	71.49	71.59
3D	38.21	47.84	47.39	72.80	56.35	56.68

Table 3: Performance of detection model trained on Completed point cloud

Test on Velodyne-64						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	15.67	17.67	14.21	68.42	56.60	57.03
bev	29.37	38.47	36.63	66.68	48.87	48.94
3D	19.78	22.19	20.83	52.51	34.00	33.99
Test on Cepton-alike						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	31.51	37.70	38.73	73.94	59.47	59.91
bev	39.56	50.63	50.89	69.18	57.55	57.61
3D	21.62	35.28	35.23	57.87	38.67	38.96
Test on Completed						
AP	Car(loU=0.7)			Pedestrian(loU=0.5)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
bbox	24.46	38.12	39.09	84.63	73.42	73.57
bev	49.41	59.82	59.56	80.11	69.43	69.46
3D	47.20	56.39	50.78	76.67	62.19	62.33

Though our project hasn't reached developing methods to solve the dataset mismatch problem as we initially planned, our project demonstrates the importance of domain adaptation in 3D detection.

A Appendix

- **Project direction**: own research project (prepare to submit)
- **Workload**
  - Divyanshu: Generate the FOV Dataset from CMU AIO dataset
  - Mengdi: Model training and evaluation
  - Chaeun (Auditing) : Discussion and data collection

References

[1] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[2] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[6] *Waymo open dataset: An autonomous driving dataset*, 2019.

[7] X. Weng, Y. Man, D. Cheng, J. Park, M. O'Toole, and K. Kitani, "All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds," *arXiv*, 2020.

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[9] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–8.

[10] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[11] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1355–1361.

[12] Y. Zhou and O. 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*, 2018, pp. 4490–4499.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[14] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[15] Y. Wang, X. Chen, Y. You, *et al.*, "Train in germany, test in the usa: Making 3d object detectors generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 713–11 723.

[16] Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov, "Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 446–15 456.

[17] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 368–10 378.

[18] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: A point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7203–7212.

[19] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5958–5967.

[20] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2790–2799.

[21] H.-K. Hsu, C.-H. Yao, Y.-H. Tsai, *et al.*, "Progressive domain adaptation for object detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 749–757.

[22] K. Saleh, A. Abobakr, M. Attia, *et al.*, "Domain adaptation for vehicle detection from bird's eye view lidar point cloud data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

[23] Y. Shan, W. F. Lu, and C. M. Chew, "Pixel and feature level based domain adaptation for object detection in autonomous driving," *Neurocomputing*, vol. 367, pp. 31–38, 2019.

[24] O. D. Team, *Openpcdet: An open-source toolbox for 3d object detection from point clouds*, `https://github.com/open-mmlab/OpenPCDet`, 2020.