Report on Center-based 3D Object Detection and Tracking

Sanjay Mullapudi[1], Rahavee Prabakaran[2], Jaya Bhargavi Mannem[3], Udaya Gopi Kappala[4]

vmullapu@asu.edu[1] Rprabhak1@asu.edu[2] jmannem@asu.edu[3] ukappala@asu.edu[4]

***Abstract*—3D Detection and tracking is one of the most important and researched topics in the field of Artificial Intelligence. 3D objects are often represented as 3D boxes in a point cloud. This representation in inspired from the well-studied 2D detection problem. But objects in 3D will have orientation which gives rise to the difficulty in fitting axis aligned box to rotated object. CenterPoint uses a different approach where objects are represented as points rather than anchors. CenterPoint first detects the center of the objects and represents them as heatmap which regress to 3D bounding box. In the second stage, the 3D bounding box is refined to address the issue of lack of depth information from map-view features generated by the backbone networks. It is trained on NuScenes dataset i.e., v1.0-mini dataset and evaluated on v1.0-test dataset. It achieved a mAP of 60.455 and NDS of 67.62.**

Keywords—3D Detection, Tracking, Voxels, Gaussians, Autonomous Driving, NuScenes.

I. INTRODUCTION (*HEADING 1*)

Autonomous vehicles are a growing and much needed technology for 21st century. The first of many problems that needed to dealt with when trying to attain complete autonomy is Autonomous driving and comprehending the Traffic environment. To achieve this there are many intertwined problems which are needed to addressed like Detecting objects like Vehicles, Pedestrians, road blocks, etc. **CenterPoint** [1] tries to solve this issue of detecting and tracking most of the frequently encountered items when driving in a general traffic scene. There has been much research in the field of 2D object detection like the systems that detects objects present in the image but these systems cannot be implemented in the field of Autonomous driving. Camera data always lack the details of orientation with respect to global co-ordinates like cyclists and pedestrians are almost planar, buses and limousines are elongated. Inclusion of LIDAR in the sensor suite can give a lot of information which lacks in a camera. LIDAR along with a simple monocular camera can do wonders in the field of autonomous driving [2].

In contrast to 2D object detection, in 3D object Detection the approach to represent objects using anchors i.e., predefined bounding boxes of a certain height and width, the approach of representing objects using a point greatly simplifies the task of 3D recognition.

Figure 1 shows that although in straight driving scenes, both anchor-based and point based methods are accurate (at t = 1). But at critical turns anchor-based methods fail to align the bounding boxes perfectly whereas point representation does the task with ease.

The center-based representation has many advantages over traditional anchor-based methods. Firstly, unlike bounding boxes have no inherent orientation. This drastically decreases the search space for the object detector while

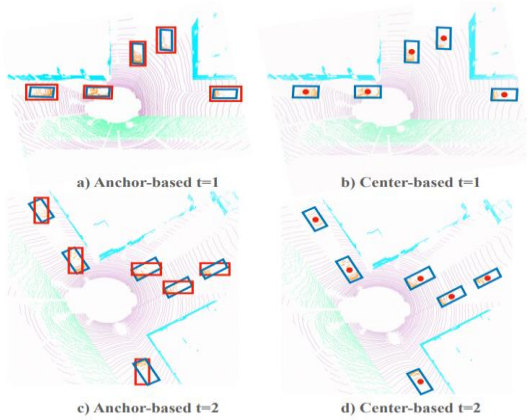

Fig 1. Comparison of Anchor based and center based methods

allowing the backbone to learn about object rotational invariance and relative rotational equivariance. Secondly, a center-based representation significantly eases the task of tracking as it tracklets are just paths in space and time.

The proposed methodology CenterPoint consists of two stages where the first stage detects centers and their respective properties and second stage refines these estimates. The first stage uses a standard LIDAR based backbone network like VoxelNet [3] or Point Pillars [4] to build representation of point-cloud data from LIDAR. This representation is flattened into an overhead-map view and image-based detectors are implemented on this flatted representation to find object centers. For, all the detected centers, it regresses to various properties like 3D size, orientation and velocity from a point feature at center location. CenterPoint predicts relative velocity of objects between consecutive frames.

The module is trained on NuScenes v1.0-mini dataset and evaluated while generates some visual samples. The Pre-trained models are made available in this link. The pretrained models are tested on NuScenes v1.0-test dataset and the resulting detections are uploaded to NuScenes detection challenge website to calculate metrics like mAP and NDS which are found to be 60.46 and 67.62.

II. METHODS

Let $P = \{(x, y, z, , r)_i\}$ be an order less point-cloud of 3D-location (x, y, z) and r reflectance measurement. The final output of the 3D detectors is expected to be a set of bounding boxes $\{b_k\}$ in bird-eye view from this point-cloud data. Each bounding box $b = \{u, v, d, w, l, h, \alpha\}$ which consists of center location (u, v, d) with respect to ground plane, 3D size (w, l, h) and rotation expressed by yaw α.

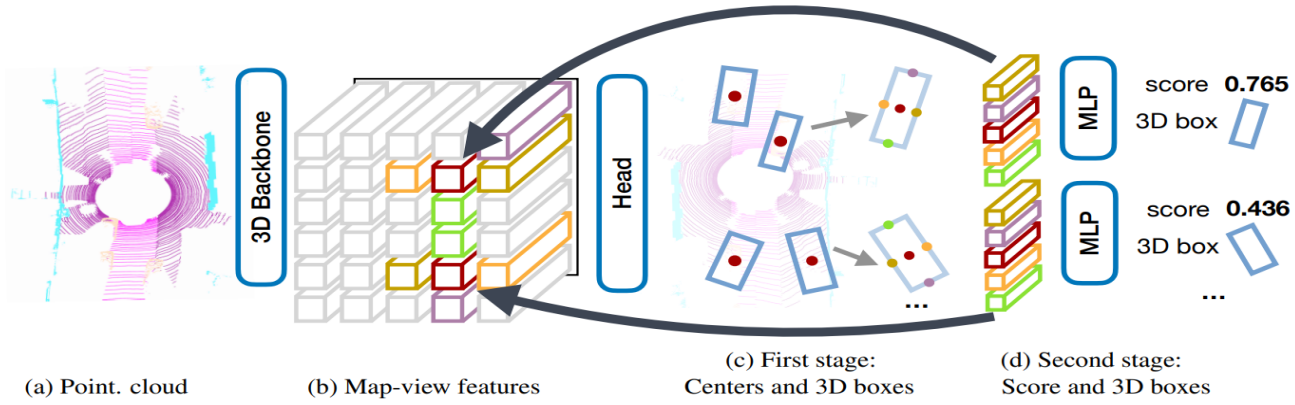

Figure 2. Overall Architecture of CenterPoint.

The overall architecture is shown in Figure 2. The architecture is divided into 3 main parts. They are Backbone Network to generate map-view features, Head to generate class specific heatmap and regress to 3D bounding box and second stage to refine the estimates from head.

A. Backbone Network

VoxelNet is one of the backbones used in the implementation of CenterPoint. VoxelNet divides a point-cloud into equally spaced paced 3D bins called voxels. A point-based network is used to extract the features for all points in each voxel and each feature representation is encoded using a 3D encoder. Output of Voxel net is map-view feature map. The architecture for Voxel Net Feature Extractor (map-view feature extractor) is shown in Figure 3.

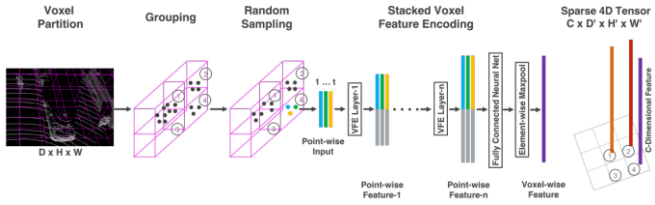

Fig 3. Architecture of Voxel net Feature Extractor

As shown in Figure 3, whole point cloud is divided into voxels of fixed size. That may result in such a way that every voxel may not have same number of points like shown in Voxel-1, Voxel-2, Voxel-4. Only voxels with greater number of points than threshold are selected and points in the selected voxel are randomly sampled. Then each point from each voxel is augmented with its deviation from centroid of voxel. These vectors form Point-wise input to Voxel Feature encoding layers. Each Voxel Feature Encoder layer converts the Point-wise inputs into Point-wise concatenated feature vectors which contain a "m" dimensional feature and element-wise maxpooled vector as shown in Figure 4.

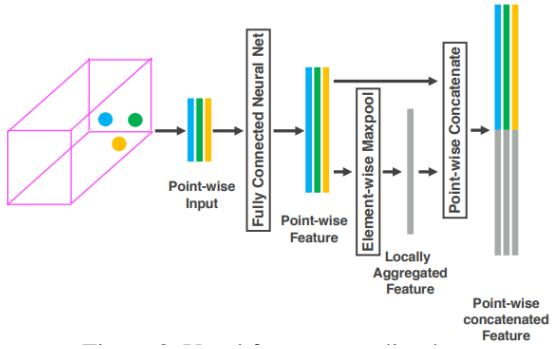

Figure 4. Voxel Feature Encoder (VFE)

These concatenated vectors are given to a Fully Connected Layer (FCN: Linear Layer x Batch Normalization x ReLU) and element-wise maxpooled to obtain a 4D tensor. As most of the point cloud data is sparse, the grouping results in most voxels to not be considered after thresholding. So, the 4D Tensor is a sparse tensor as shown in the end of Figure 3. This 4D tensor is converted into map-view features $\mathbf{M} \in \mathrm{R}^{\mathrm{WxLxF}}$ of width W, Length L and F channels.

B. Class specific heatmap and regression head to 3D Bounding boxes

There are many Regression heads in this stage namely, Center heatmap head, Regression heads (3D Bounding Box), Velocity and Tracking head.

a) Center heatmap head: The Center heatmap head's goal is to generate K-channel heatmap for each of K-classes. The heatmaps are supposed to show peaks at the center of the detected objects. The target is to match a 2D Gaussian produced by projection of 3D centers onto map-view as shown in Equation 1.

$$. Y_{xyc} = \exp\left(-\frac{(x-p_x)^2+(y-p_y)^2}{2\sigma_p^2}\right)$$

where σ_p is object size-adaptive standard deviation and $\{p_x,p_y\}$ is the ground truth center.

This produces a sparse heatmaps where most locations are considered as background. To curb this effect, we set the gaussian radius minimum to 2.

b) 3D Bounding Boxes Regression head: To generate 3D bounding boxes, the output of Center heatmap head is

used to regress to height above ground $h_g \in R$, 3D size $s \in R^3$ and orientaion angle α in terms of $(\sin(\alpha),\cos(\alpha)) \in R^2$.

Each output uses its own regression head to calculate the properties. At training time, only ground truth centers are supervised using L1 loss. Combining h_g with center location on map-view feature gives 3D center. The outputs from all these regression heads gives 3D bounding box.

c) Velocity and Tracking Head: This head tries to predict difference in object position in consecutive frames. It is supervised using L1 loss at the ground truth object's location at the current frame. The negative velocity estimate is used to project the current frame's object centers back to the previous frame, then use closest distance matching to match them to the tracked objects. Using SORT [5], unmatched tracks upto 3 frames are kept before deleting them.

All the properties in this stage are obtained from map-view features which may not have sufficient information to localize object efficiently. So, a second stage is used to refine these estimates.

C. Second stage (Refinement):

This stage extracts additional point-features from the output of backbone. The point-features of 3D centers from each face of estimated bounding boxes are extracted. A bilinear interpolated feature from backbone map-view output **M**. These both features extracted are concatenated and fed into MLP (Multi Layer Perceptron). MLP predicts class-agnostic confidence score and bounding box refinement of Regression head as shown in Figure 2.

For class-agnostic confidence score, target **I** is generated using 3D IoU with ground truth box as shown in Equation 2.

$$I = \min(1, \max(0, 2\,x\,IoU_t - 0.5)) \qquad (2)$$

where IoU_t is IoU between t-th detection and ground truth.

The training is supervised using binary cross entropy loss is as shown in Equation 3.

$$L_{score} = -I_t \log(\hat{I}_t) - (1 - I_t)\log(1 - \hat{I}_t) \qquad (3)$$

where $\hat{I}_t$ is the predicted confidence score.

The model predicts a refinement on top of first-stage outputs in box regression, and the model is trained using L1 loss.

D. NuScenes Dataset and Metrics used.

NuScenes dataset contain a total 1000 driving scenes Boston and Singapore. Each is scene is approximately of 20 seconds. The dataset is annotated with 23 classes along with 3D bounding boxes at 2Hz frequency. A sample LIDAR scan from bird eye-view is shown in Figure 5.

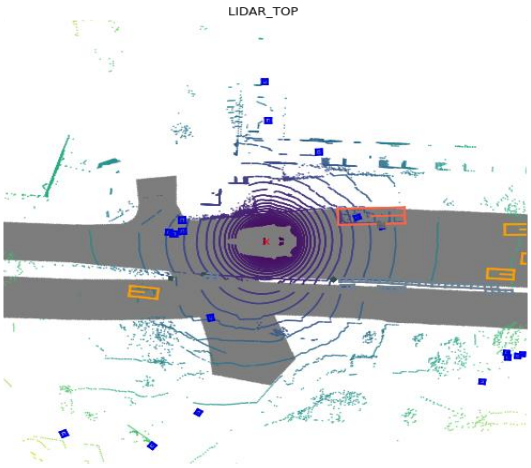

Fig 5 Sample LIDAR scan

The same sample's Camera images are shown in Figure 6 and Figure 7. The truck seen in this can be rendered separately by using annotation's token as shown in Figure 8. Figure 9 shows how LIDAR data is seen in Front camera perspective.

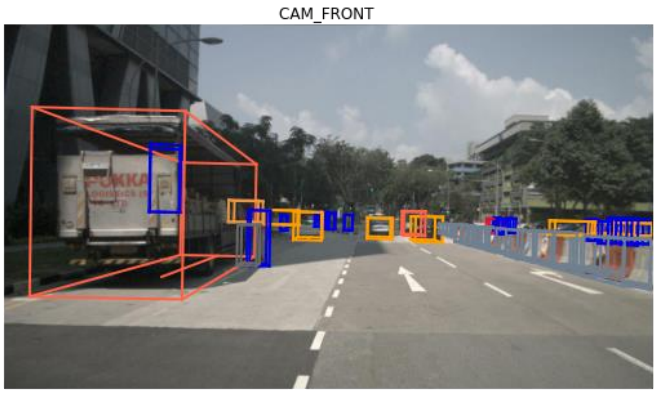

Fig 6. Front Camera data sample

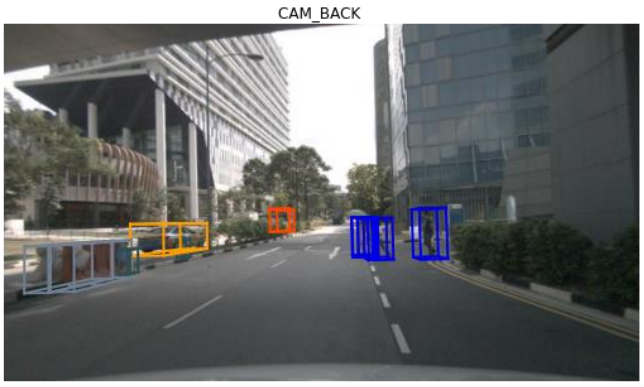

Fig 7. Back Camera data sample.

The dataset is divided into 700, 150, 150 scenes for train, validation and test set. The official evaluation metric for the task is average among classes. For 3D detection, the metric is **mAP** and **NDS** (NuScenes detection score).

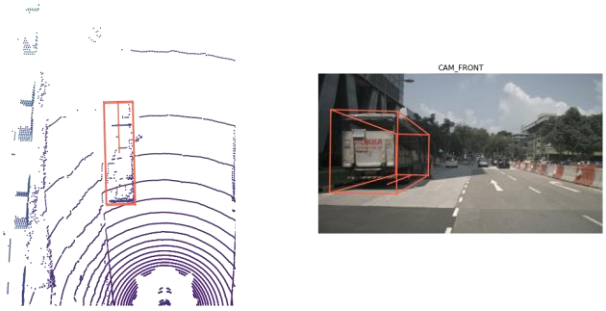

Fig 8. Only Annotated truck is visualized.

Mean Average Precision (mAP) is one of the most famous metrics for object detection. To calculate mAP, Average precision for each class is calculated and it is aggregated over all classes.

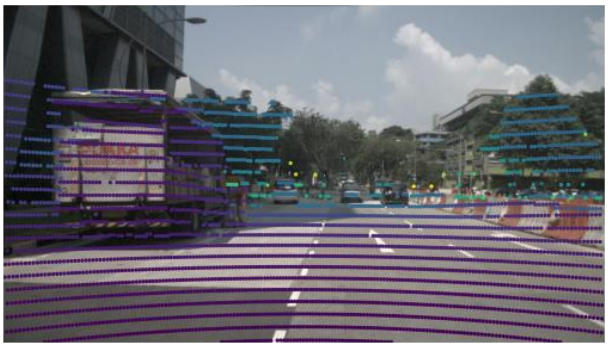

Fig 9. LIDAR Data rendered on Camera image.

True positive metrics are used to measure translation / scale / orientation / velocity and attribute errors.

- **Average Translation Error (ATE)**: Euclidean center distance in 2D in meters.
- **Average Scale Error (ASE):** Calculated as 1 - IOU after aligning centers and orientation
- **Average Orientation Error (AOE):** Smallest yaw angle difference between prediction and ground-truth in radians. Orientation error is evaluated at 360º for all classes except barriers where it is only evaluated at 180º. Orientation errors for cones are ignored.
- **Average Velocity Error (AVE):** Absolute velocity error in m/s. Velocity error for barriers and cones are ignored.
- **Average Attribute Error (AAE):** Calculated as 1 - acc, where acc is the attribute classification accuracy. Attribute error for barriers and cones are ignored.

The TP metrics are defined per class, and then a mean is taken over classes to calculate mATE, mASE, mAOE, mAVE and mAAE.

NuScenes Detection Score (NDS): the above metrics are consolidated by computing a weighted sum: mAP, mATE, mASE, mAOE, mAVE and mAAE. [6]

III. Implementation and Simulation

The CenterPoint is implemented on NuScenes dataset. The model is trained on NuScenes v1.0-mini dataset. And tested on v1.0-test using pre-trained models.

Software Requirements:

- Ubuntu 18.04 LTS
- Python 3.6.9
- Pytorch 1.10.1+cu111
- CUDA 11.1
- Spconv-cu111
- Numpy
- Scikit-Learn
- Nuscenes-devkit

In Ubuntu 18.04 LTS, NVIDIA drivers to access the GPU are installed. Docker is installed and NVIDIA container toolkit is installed alongside. This gives the docker containers to access GPU. The GPU used is NVIDIA RTX 3050 Ti with NVIDIA 510 GPU drivers.

All the executions are done using Linux terminal. A Docker file alongside the code will be available in this **link**. Required bash scripts are also provided to install the requirements.

a) Training: While training on single GPU, a batch size of 1 and 20 epochs is used because GPU couldn't handle bigger tensors while loading the data.The v1.0-mini dataset is downloaded from this **link**. Annotations files are created using create_data.py code which generate train data grount truth label files as shown in Figure 10. The training on v1.0-mini took approximately 4 hours which can be seen in Figure 11.

b) Testing: For testing the model, v1.0-test is downloaded (~36GB) from this **link.** Test dataset conatins 150 scenes which contains around 6008 samples. So, the testing took around 70mins.

IV. Results

All the Code files, Results and Figures provided in this section are also available in this **link.**

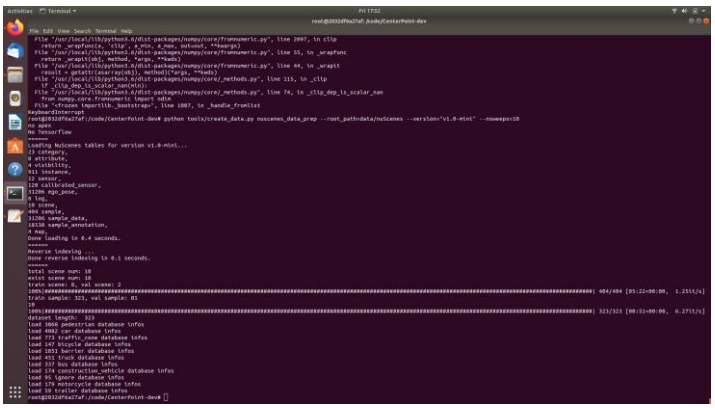

Fig 10. Creation of ground truth datafiles.

The generated pickle files are used in configuration files while evaluating. Figure 11 shows start of Epoch 1/20 and Figure 12 shows end of training i.e., Epoch 20/20. The time can be seen in the screenshots provided.

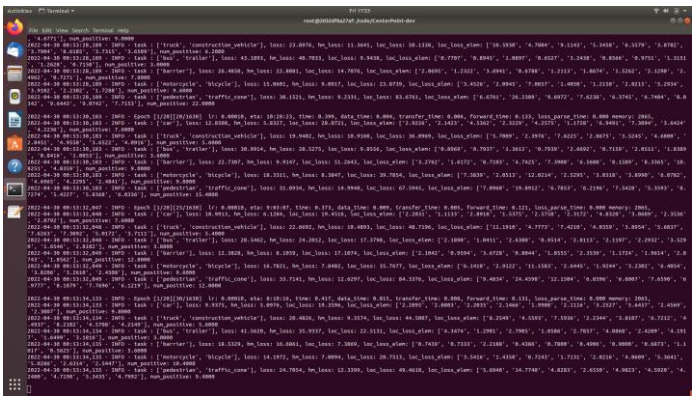

Fig 11. Training started: Epoch 1/20

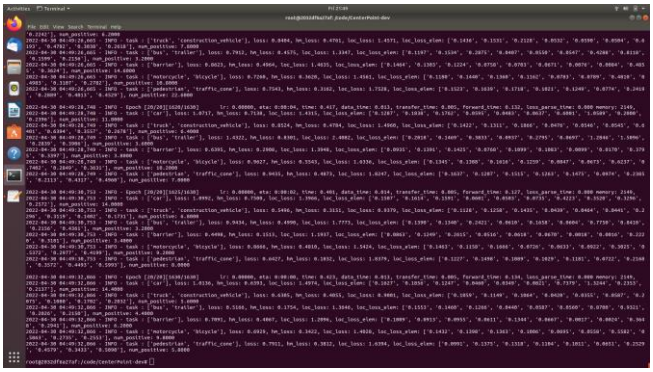

Fig 12. Training ended: Epoch 20/20

Figure 13 shows sample detection in birds eye view. Figure 14,15,16 shows same sample to visualize the LIDAR

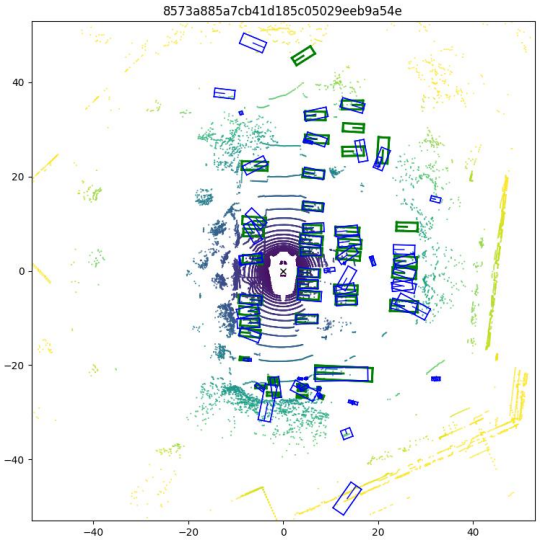

Fig 14. Detection result for a sample token. Blue boxes are predicted bounding boxes and green boxes are ground truth boxes.

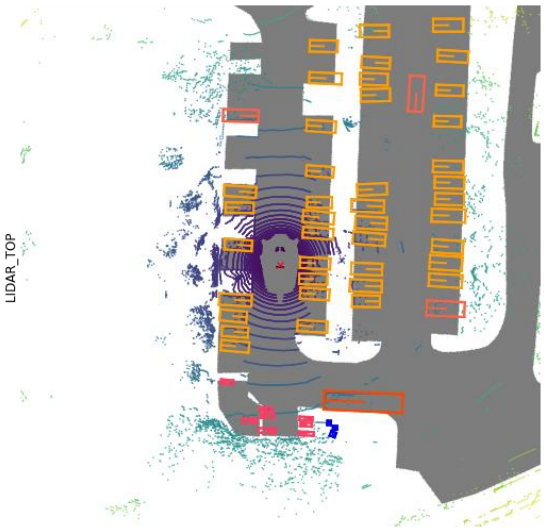

Fig 15. Shows the sample token's only ground truth boxes.

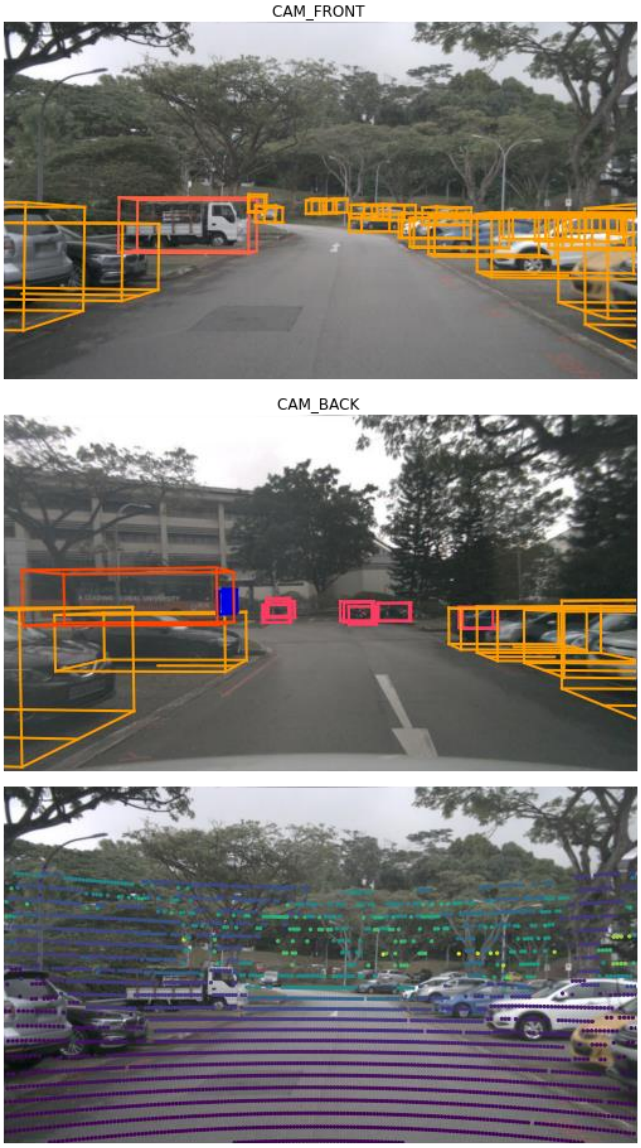

Fig 16. Camera data of sample token. Front Camera (top). Back Camera (middle). LIDAR points plotted on camera image to infer the predictions (bottom).

Figure 17 shows the testing script running successfully and generating json file which contains detected outputs on v1.0-test dataset.

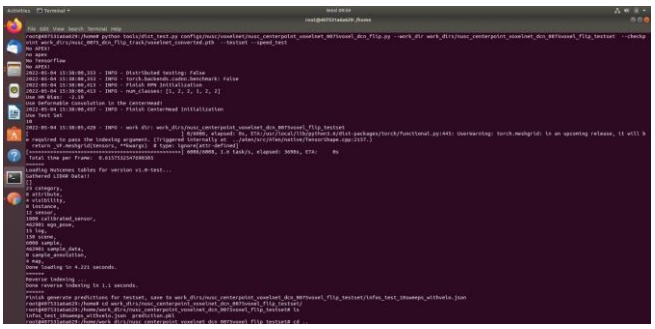

Fig 17. Testing on v1.0-test and json generation

NuScenes test annotations are not publicly available. They are only available to submit in NuScenes detection challenge website on this **link**. Figure 18 shows the submission results. Submission screenshots are provided in this **link.**

Test results show a mAP of 60.455, NDS of 67.16 as shown in Figure 18.

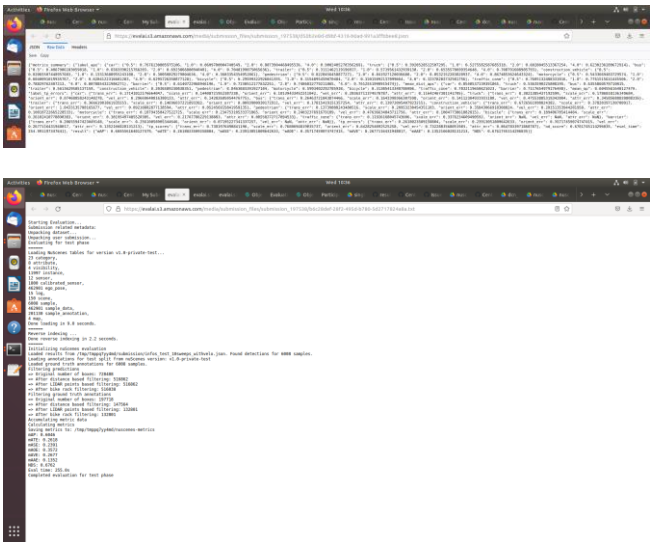

Fig 18. Metrics summary in json format (top) and Evaluation logs in txt format (bottom).

V. Conclusion

LIDAR data is always very sparse which results in memory wasted if loaded without any preprocessing. So, the idea of Voxels and Voxel Feature Encoding contributed a lot to handle the problem of sparsity. The key idea is to represent objects as centers rather than anchors which reduced much of the time and computational complexity. CenterPoint is very light weight and near real time 3D detector and tracker. It is based on many other open-source projects like VoxelNet, PointPillars, and CenterNet. In NeurIPS 2020, NuScenes 3D detection challenge, CenterPoint is adopted in 3 of 4 winning entries.

VI. References

[1] Tianwei Yin, Xingyi Zhou, Philipp Krähenbühl, Center-based 3D Object Detection and Tracking https://arxiv.org/abs/2006.11275

[2] https://www.louisbouchard.ai/waymo-lidar/

[3] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CVPR, 2018

[4] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. CVPR, 2019.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. ICIP, 2016.

[6] https://www.nuscenes.org/object-detection/#evaluation-metrics

every team member is required to complete this form, which will be reviewed and signed by other team members

by signing this form, you agree that the student's descriptions of contributions are factual

effort/contribution levels determine an individual's score which can be different from the rest of the team

This form does not contribute to page count.

Assignment name/key words Final_Package_submission

Team #12

Student name:Rahavee Prabakaran	worked on literature	worked on implementation (data, platform, test run, debug, compatibility...)	generated results (run results, result data processing, presenting results	wrote report (Intro, method, result, discussions, ...)	other significant contributions	peer approval 1	peer approval 2	peer approval 3
specific & detailed evidence is required to support claims of contributions (make reference to specific paragrphs, equation #, figure #, code line #'s sections, etc...)	Found papers related to CenterNet. Worked with team members to understand the background literature.	Downloaded Nuscenes v1.0-mini for training and helped to implement training inside Docker environment.	Involved in creating ground truth files for v1.0-mini dataset for training.	Wrote Intro, discussions and Abstract.	NA	Udaya Gopi Kappala	Sanjay Mullapudi	Jaya Bhargavi Mannem

Student name:Sanjay Mullapudi	worked on literature	worked on implementation (data, platform, test run, debug, compatibility...)	generated results (run results, result data processing, presenting results	wrote report (Intro, method, result, discussions, ...)	other significant contributions	peer approval 1	peer approval 2	peer approval 3
specific & detailed evidence is required to support claims of contributions (make reference to specific paragrphs, equation #, figure #, code line #'s sections, etc...)	Found papers related to VoxelNet.Worked with team members to understand the background literature.	Implemented whole system on Ubuntu 18.04 using Dockers. All the requied packages are installed using bash scripts and generated results and visual renderings.	Implemented the train on v1.0-mini dataset and test codes to generate json detection results on test dataset to submit the detections on nuscenes detection challenge for evaluation	Wrote about VoxelNet Architecture, and CenterPoint architecture. And conlcuison	NA	Jaya Bhargavi Mannem	Udaya Gopi Kappala	Rahavee Prabakaran

Student name:Jaya Bhargavi Mannem	worked on literature	worked on implementation (data, platform, test run, debug, compatibility...)	generated results (run results, result data processing, presenting results	wrote report (Intro, method, result, discussions, ...)	other significant contributions	peer approval 1	peer approval 2	peer approval 3
specific & detailed evidence is required to support claims of contributions (make reference to specific paragrphs, equation #, figure #, code line #'s sections, etc...)	Found papers related to PointPillars. Worked with team members to understand the background literature.	Helped to develop the bash scripts required for installation of packages, Studied Nuscenes devkit which helped to understand the dataloader of v1.0-test dataset. Helped in implementation of testing	Generated visual renderings of LIDAR and camera to infer on detctions.	Wrote Implementation, Gathered data about Nuscenes dataset	NA	Rahavee Prabakaran	Sanjay Mullapudi	Udaya Gopi Kappala

Student name:Udaya Gopi Kappala	worked on literature	worked on implementation (data, platform, test run, debug, compatibility...)	generated results (run results, result data processing, presenting results	wrote report (Intro, method, result, discussions, ...)	other significant contributions	peer approval 1	peer approval 2	peer approval 3
specific & detailed evidence is required to support claims of contributions (make reference to specific paragrphs, equation #, figure #, code line #'s sections, etc...)	Found papers on Point based detectors for 3D object Detection. Worked with team members to understand the background literature.	Worked on generating evaluation metrics on test dataset and generating pickle files for test set	Generated json detection files and pickle files to generate test evaluation metrics.	Wrote Results, Metrics used for evaluation. Wrote implementaion.	NA	Sanjay Mullapudi	Jaya Bhargavi Mannem	Rahavee Prabakaran

This table is required and to be included in the final report
(this table does not contribute to the page count)
Please don't change the format of the table, i.e., only fill in the blanks
end of semester reflection - lessons learned from working on the final project
Team 12 and names of team members: Sanjay Mullapudi, UDAY GOPI KAPPALA, JAYA BHARGAVI MANNEM, RAHAVEE PRABAKARAN

Project title CENTER-BASED 3D OBJECT DETECTION AND TRACKING

	literature (not well written or self-contained, not specific on implementation, no data source indicated, no source code indicated…)	setting up the environment and obtaining data	to have the first successful test run (issues during debugging, compatibility problems…)	obtaining results (algorithm/method is dificult to implement, hyper parameters difficult to tune...)	obtaining results (cannot duplicate what was reported in paper, if so, why?)	reporting (Intro, method, result, discussions, …)

specific & detailed evidence is required to support claims (e.g., links, repository sites, equation #, figure #, paragraphs, sections, etc…)	Building det3d package was issue and not much literature was found to solve. Found a solution in one of the closed Issues on Github page. https://github.com/tianweiy/CenterPoint/issues/46	Data is publicly availble and Understanding the Nuscenes data format and usage of nuscenes devkit helped a lot to debug the code. https://www.nuscenes.org/nuscenes?tutorial=nuscenes Docker is one of the most useful tool which I learnt during the project which helped a lot. It is still an open issue about the Dockerfile. https://github.com/tianweiy/CenterPoint/issues/17	Downloading the v1.0-test (~36GB) and arranging it in the required format. Creating a data loader for evaluation was an issue but learnt a lot about GPUs and their parallel computing capabilites.	Detections are generated in form of json form but metrics were not generated which troubled a lot but found a solution to it in an issue. https://github.com/tianweiy/CenterPoint/issues/138 The eval.ai website was useful for generating metrics. Understood how competetions and challenges work in the conferences and workshops. It got me excited to participate by developing a novel approach for 3D Detections.	Couldn't work on Waymo Dataset because it needed Tensorflow and had a problem installing both PyTorch and Tensorflow with their respective GPU and CUDA supports.	The report writeing helped me a lot to dig into the references given in the orginal paper and study a lot background research. In the course of writing the report made me read 4-5 papers which got me excited to learn more about the fusion of LIDAR, RADAR and camera for 3D detections in Autonomous Vehicles.

footnote (e.g., reference citation.. Original Paper https://arxiv.org/pdf/2006.11275.pdf