CSE 4088 Introduction to Machine Learning Course Final Project Report

3D Object Detection Using Lidar Point Clouds

Onur Can Yücedağ - 150116825

Emre Erdem - 150115501

Alperen Bayraktar – 150116501

ABSTRACT

Detection object in the wild is hard, especially when we think about different weather conditions there exists in our normal living life. Human eyes are particularly powerful about focusing objects that is far in distance and isolating the object of interest. We also have an immense memory system that is gathering information in real-time.

For a machine to do this job just like a human is hard and we have a long way to go. But we also have different sensors that can see what we can't see or absorb. Camera and LiDAR sensors are the examples that nowadays we use to detect objects. 2D object detection is relatively mature compared to the 3D object detection.

Main reason that this task is still behind the traditional 2D Object Detection research is;

1. Using 2D convolution on images is very cheap process when compared to the 3D convolutions.
2. 3D data such as point clouds and radar is a high price tag entry to begin with in this research.
3. 3D unconstructed data such as point clouds are not trivial as RGB raw images, they are required to many transforms to work with the current development of feature extractors such as convolutions, pooling, residual connections.
4. Finally, 3D object detection is used in autonomous driving systems. These systems must be fast as a real time system.

Camera will mostly be used for 2D object detection (-although we have 3D object detection researches, they are not powerful as 2D detection tasks[1]) and won't work in the rainy, foggy, dark days and times or any calibration mistake will be paid big time. For that reason, in this project we are using LiDAR point clouds to detect objects in 3D space.

LiDAR point clouds are sparse 3D data that let anyone can think that operating 3D convolution on them natural. But as you can imagine 3D convolutions are heavy to compute and systems like autonomous vehicles needs to run at real-time, so that's why we need to create an embedding to operate 2D convolutions to this data.

3D object detection task is an embedding/input representation task. Most novelty in any research comes from the novelty in the input representation. Lately PIXOR[2], Point Pillars[3] and PointNet[4] papers changed the accuracy and execution time leaderboard in this area. Therefore, we choose researching these three papers in depth and implementing the most successful one that we think.

Overview – Project Accomplishment

This research is a survey of the three different 3D object detection methods such as: PIXOR, Point Pillars and PointNet. Everyone of them are essential for us because they include much explanations and implementation ideas in their papers.

Unlike 2D pixel arrays such as images, 3D voxel arrays such as point clouds have an unstructured nature because data is simply a set of points during scan of a LiDAR. In order to use existing methods (2D and 3D convolutions) researchers often discretize the point clouds by taking multiple views of the data in z-axis to create multiple 2D images or creates voxels out of the data.

Point cloud data is fully described by x, y and z values for each point.

Properties of the point cloud data:

- ***Permutation Invariance:*** Single scan in a 64 channel LiDAR can generate 64! different permutations and subsequent points processing needs to be invariant to the different order.
- ***Transformation Invariance:*** Classification results must be unchanged if point cloud data undergoes a transformation such as rotation and translation.
- ***Point Interactions:*** Interaction between neighboring points is important. Therefore, single point can't be isolated from its neighbor.

PointNet

Classification network uses a shared MLP to map n points from three dimensions (x, y, z) to 64 dimensions. Single MLP is shared for each of the n points (mapping is i.i.d). This procedure repeated to map input n points from 64 dimensions to 1024 dimensions. Network representation can be seen in Figure 2.

Max pooling used to create global feature vector. Lastly, three-layer fully connected network used to estimate k output classification scores.

As we mentioned above for n input points n! permutations exist. For solving this problem, authors suggested symmetric functions such as sum, average and max/min. PointNet uses max function to create global feature vector which is directly used for outputting the classification scores.

Also, there is another problem, Transformation Invariance. Classification results needs to be invariant to the transformation of the point clouds which can be happen often because it is a highly unordered data structure. In the "input transform" and "feature transform" parts of the network, PointNet uses a small network for pose normalization of the given point cloud data.

This idea is influenced by Spatial Transformer which can be seen in Figure 1.

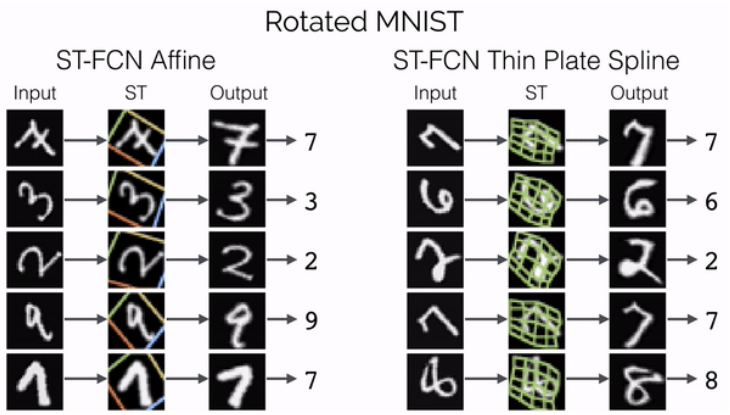

Figure 1 Various inputs and corresponding outputs of a Spatial Transformer [5]

For a given input point cloud, apply an appropriate rigid or affine transformation to achieve pose normalization. Because each of the n input points are represented as a vector and are mapped to the embedding spaces independently, applying a geometric transformation simply amounts to matrix multiplying each point with a transformation matrix.

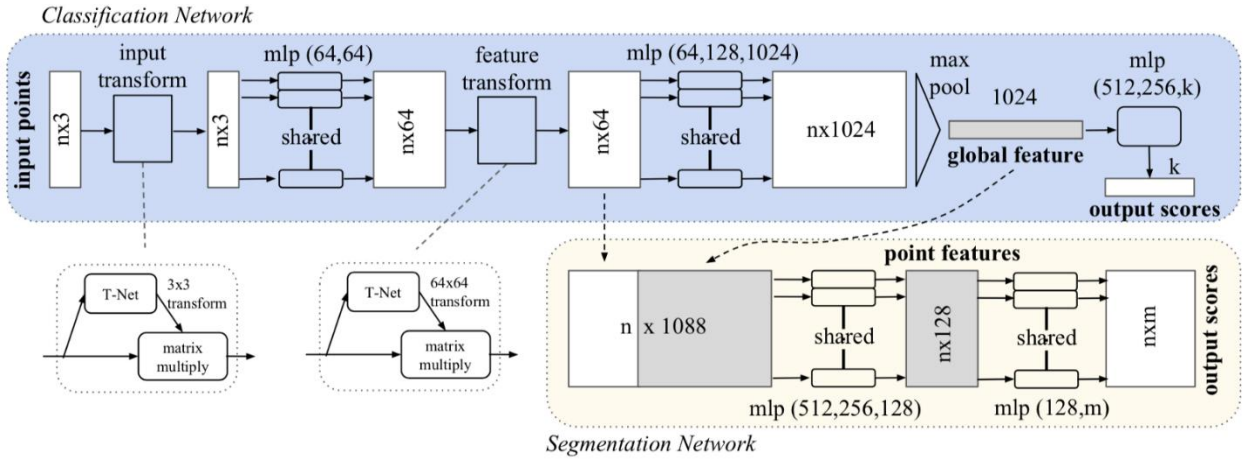

Figure 2 PointNet classification and segmentation network [4]

The points that do contribute to and define the global feature vector are referred to as the critical point set and encode the input with a sparse set of key points.

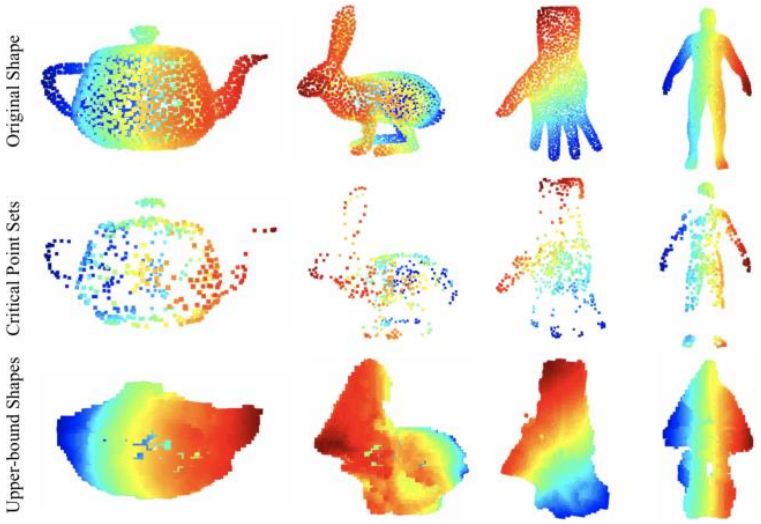

Figure 3 Visualization of critical point sets and upper-bound shapes [4]

Point Pillars

Point Pillars model was particularly interesting because it is directly focusing on input representation as we described as a most effective novelty in these researches. It uses a simplified PointNet architecture for gathering the transformation relationship.

It will divide the point clouds in the vertical columns called pillars. These pillars have features that is describing each of them; distance to the arithmetic mean of the pillar, offset from the pillar and global x, y, z values.

For each pillar it will run the PointNet architecture to perform prediction of transformation matrix to eliminate the unordered nature of these points in the point cloud. These two distinctive features of the Point Pillars are the most important novelties of the model.

Learned features that is coming out from the PointNet model will be modified to create a Psuedo Image representation. This representation will allow us to use 2D convolutions instead of 3D which is heavy to compute.

Pseudo Image will be sent to Backbone 2D CNN architecture to create embeddings as well as up sampling with transposed convolutions.

Lastly, a Detection Head used to predict the bounding boxes with their classes in 3D coordinate system. This model selected as SSD[6] by the authors. They also tried YOLO[7] which worked nearly good as SSD.

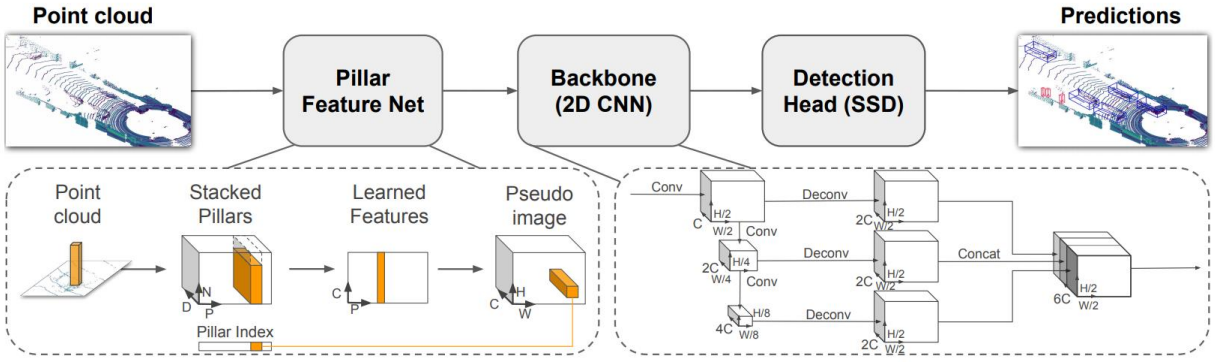

Figure 4 Network overview of Point Pillars model [3]

PIXOR

For this report we only show the results from the PIXOR model since it is a different model in many aspects:

- It uses BEV (Bird's Eye View) for input representation. This representation gives the model robustness in case of scaling of the objects. Also, it mainly uses a priori; all objects need to be in x-y plane such as autonomous driving domain. BEV perspective also helping the occlusions between the objects very much since the point cloud data structure changing rapidly from one perspective to another.
- It doesn't use any voxelization or vertical columns for the input representations, but PointPillars use Pillar and VoxelNet[8] uses voxel grids which make the computation fast but loses fair amount of information.
- It is single-stage detector, meaning that it doesn't rely on proposals for object detection.
- It uses focal loss with smoothing which makes class-imbalance problem less effective.
- It uses occupancy grid matrix once it forms the BEV perspective and uses 2D convolutions only to make computation very cheap when compared to SECOND[9].

You can see an output representation in Figure 5.

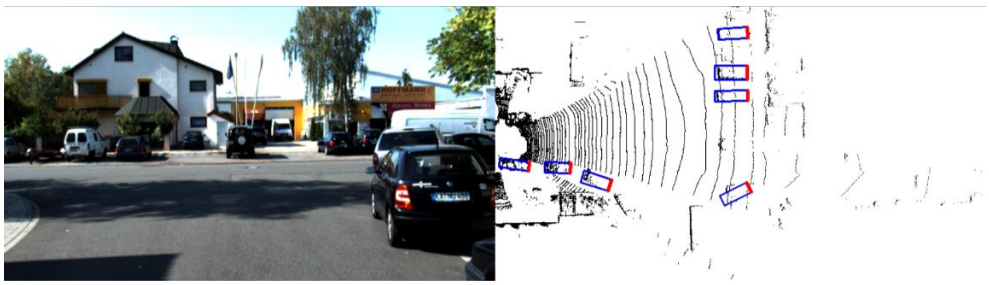

Figure 5 PIXOR model's output example [10]

SUMMARY

Object detection tasks relatively an important part in the real-time systems such as autonomous vehicles. These systems needs to compute and evaluate their estimation results in 30 FPS or higher because any delay that can occur during in a critical part of the work they do, people can lose their lifes.

Just as execution time, accuracy is another important aspect of these methods. Accuracy of their findings will lead to different results which can lead to different trajectories for an EGO vehicle.

In this project, our main focus was experiencing with these systems and researches. Implementing these models was very powerfull feeling and it was the core of ouw excitement for the field.

In the project presentation, we will present the demo of our implementation of the PIXOR model. It is very fast and accurate method but it needs to be developed to create better estimations in lesser time like every method in machine learning field.

As mentioned in the earlier in this paper, these 3D object detection using point clouds methods are strong as their representation of input. Input transformation needs special attention. For Point Pillars, this representation is combined with pillars; for PointNet it was pose transformations; for PIXOR, it was the BEV transformation of the point cloud data. Everyone of the methods are contributed to this field.

We need to study encoding of the inputs and pay much attention to fusing different sensors such as camera, radar or vector map informations with the LiDAR point clouds.

Sparse nature of the point clouds make the applying regular convolution operations (like on images) much harder compared to the images. We can’t just plug in the point cloud data as input to the YOLO and expect that something is going to work. Because images are dense, its easily understandable for computers and cheap to compute comparing with the LiDAR point cloud data.

REFERENCES

[1] Kim, Youngseok, and Dongsuk Kum. "Deep Learning based Vehicle Position and Orientation Estimation via Inverse Perspective Mapping Image." *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019.

[2] Yang, Bin, Wenjie Luo, and Raquel Urtasun. "Pixor: Real-time 3d object detection from point clouds." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.

[3] Lang, Alex H., et al. "PointPillars: Fast encoders for object detection from point clouds." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[4] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[5] Review: STN — Spatial Transformer Network (Image Classification) (2019). Towardsdatascience.com. Retrieved 25 December 2019, from towardsdatascience.com/review-stn-spatial-transformer-network-image-classification-d3cbd98a70aa

[6] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[7] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[8] Zhou, Yin, and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[9] Yan, Yan, Yuxing Mao, and Bo Li. "Second: Sparsely embedded convolutional detection." Sensors 18.10 (2018): 3337.

[10] PyTorch Implementation of PIXOR (2019). github.com. Retrieved 21 November 2019, from github.com/philip-huang/PIXOR