MS THESIS FALL SEMESTER REPORT

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF WORCESTER POLYTECHNIC INSTITUTE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Sanket Gujar
January 2019

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Prof. Michael Gennert) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science.

(Prof. Jacob Whitehill)

Approved for the Worcester Polytechnic Institute Committee on Graduate Studies

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The camera is the cheapest and computationally real time option for detecting or segmenting the environment for an autonomous vehicle, but it does not provide depth information and is surely not reliable during night, bad weather, tunnel flash out etc. The risk of an accident gets higher for autonomous cars when driven with a camera in such situations. The industry has been relying on LiDAR for the past decade to solve this problems and focus on depth information of the environment. Industry leaders like Waymo and Uber ATG now even started using there in-house manufactured LiDARs instead of relying on other manufacturers.

There has been lot of attention given to research and investment in the autonomous vehicle industry which lead to significant improvement in LiDAR technology. The LiDAR market is going to hit **5,204.8 Million** USD soon and new silicon valley startups like Luminar Technologies and Ouster have released their LiDAR product which can detect up to 200m and are giving a good competition to the currently dominating LiDAR company Velodyne.

The Investment, resources and the drastic improvement ensures the technology will be a dominating factor for the autonomous vehicle industry in the future, but the issue commonly faced with

Figure 1.1: Camera Challenges: Bad weather and dynamic lighting condition are a big challenge for camera based autonomous driving.

LiDAR is more on the computational efficiency of perception stack. Running the current pointwise algorithm [11, 12] in real time with the input feed of more than 1M points per second needs lots of hardware optimization. Pointwise classification or segmentation with so many points is computationally expensive and will not be real time.

It is possible to build a self-driving car with camera alone but it will not always be safe and having safe autonomy means driving with fewer errors. Different sensor modalities have different strengths and weaknesses. Cameras suffer from difficulty to adjust and detect object in low-light and high dynamic range scenarios. Radar suffers from limited resolution and artifacts due to multi-path and doppler ambiguity. LiDAR sees obsurants as it is sensitive enough to detect snow, making it difficult for robustness in projection based methods.

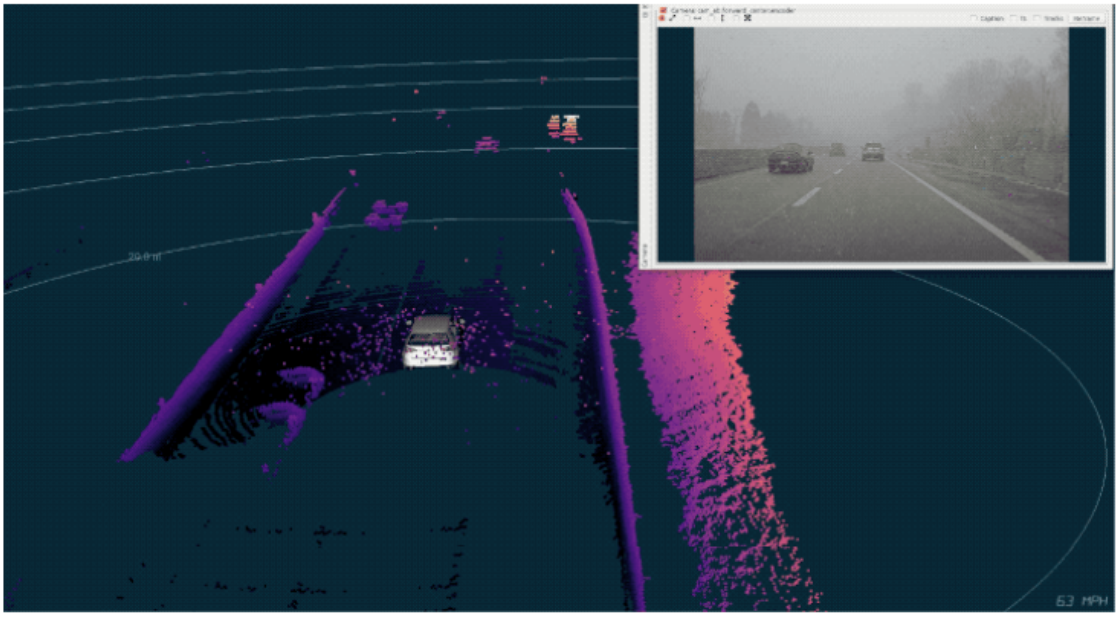


Figure 1.2: LiDAR sensitive to snow which makes it difficult for projection methods to detect vehicles

1.1 Projection methods

The Projection methods [14, 1]are the fastest methods for object detection for autonomous cars but are not robust enough. In projection methods, the points are projected on a plane to obtain a 2d projected image and the model tries to localize the target given the projected image. The most common projection used are birds eye view projection and front projection. The front projection is made by projecting all the points on the front view plane. The points will not overlap over each other but, if two target objects lie close enough they might be projected as a single entity in the

front projection confusing the model. Birds eye projection is made by projecting all the points on the ground plane. Here points may overlap over each other and the priority of the class of the points decides which point will be projected on the plane. It is fast for real time application and has shown good results [7] with detecting vehicles but it still has drawbacks (fig: 1.3, 1.4) making it not reliable. [1] takes Birds eye projection as well as the Front projection and fuses the features with RGB image features to regress the 3d box coordinates

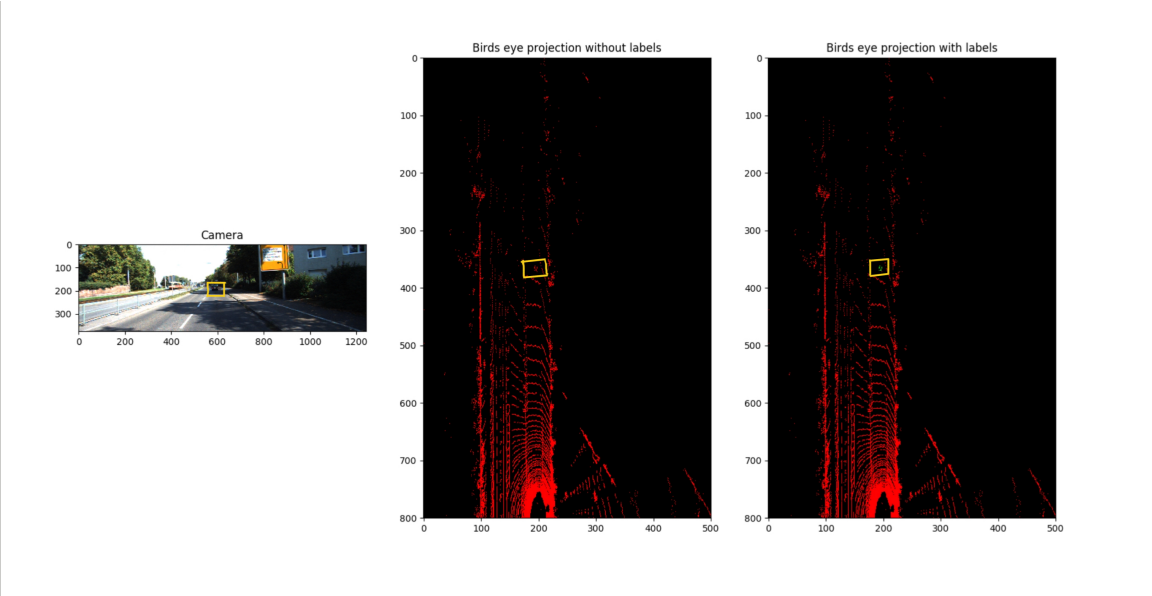


Figure 1.3: Birds eye view projection drawbacks
The car (yellow box) is represented by a small number of points in the point cloud and birds eye projections samples down it even further making it difficult to detect by the model.

1.1.1 When will projection based methods fail

The projection methods are fast but are likely to fail given:

1. The vehicles are close to one another making their projections to look like a single object in the front projection.

2. Pedestrians and poles are represented as a small compact cluster in birds eye view projection making it hard for the model to detect them (fig 1.4)

3. The difficulty of differentiating between large and small vehicles just on projections as the back part of vehicles is projected just as a line for any vehicle irrespective of their dimensions.

4. Weather conditions like rain and snow as Lidar is sensitive enough to detect this point adding a uniform noise distribution in the projections.

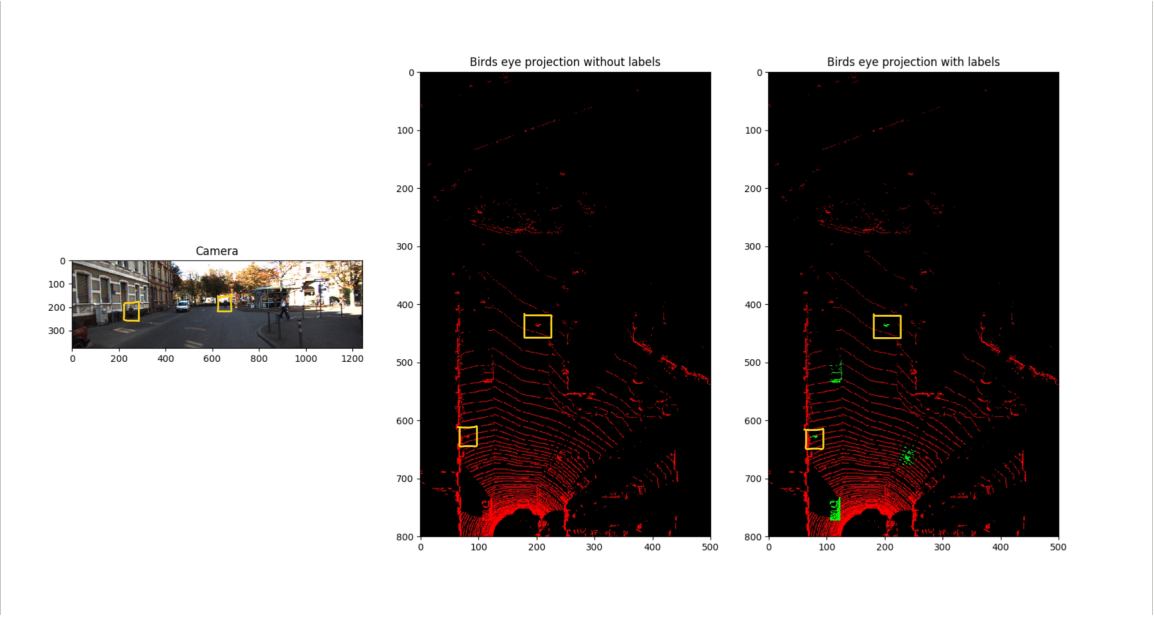


Figure 1.4: Birds eye view projection drawbacks
The pedestrains (yellow boxes) are respresented as a small compact cluster in the projection.

1.2 3D Point wise Methods

Alternative to projection methods are the pointwise methods, where the point is represented by just its 3 coordinates (x,y,z) and additional dimensions can be added such as normal, intensity etc, depending on the availability.

Pointnet [11] was the initial approach for a novel type of neural network that directly consumes unordered point clouds, which also takes care of the permutation invariance of points in the point cloud. Pointnet can do object classification, part segmentation, and scene semantic parsing. The main feature of Pointnet is its robustness with respect to input perturbations and corruptions. Pointnet has 3 main key modules (Fig 1.5), which were use of symmetric function, a local and global information combination structure, and two joint alignment networks that aligns both input points and point features. The most important key was the use of the single symmetric function, MaxPooling, which inputs a single vector which is invariant to the permutations of the input vector orders.

Pointnet failed to capture local structure and generalize to complex scenes, so the authors modified the architecture resulting in Pointnet++ [12]. The intuition of Pointnet++ came from the basic CNN structure where its lower level neurons have smaller receptive fields whereas larger level have larger receptive fields. The ability to abstract local patterns along the hierarchy allows better generality to unseen cases.

Pointnet++ is a hierarchical network that applies Pointnet recursively on a nested portioning

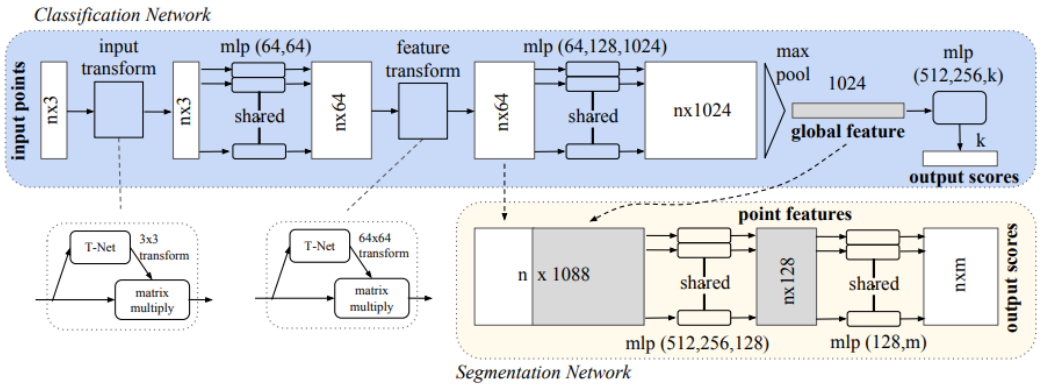


Figure 1.5: Pointnet architecture

of the input point cloud. It proposes novel set learning layers to adaptively combine features from multiple scales from varying densities. Similar to CNNs, Pointnet++ extracts local features from small neighborhoods and groups them into larger units and processes the groups to produce higher level features. This process is recursive until we obtain the feature of the whole point set.

The two main issues addressed by Pointnet++ were the partitioning of point set, and abstraction of points or local features through a local feature learner. Both these issues are correlated as the partitioning of the point set has to produce common structure partitions, so that the weights of the local features can be shared. Pointnet++ uses Pointnet as the local feature learner.

The hierarchical structure (Fig : 1.6) is composed by a number of set abstraction levels. The set abstraction layers consist of three layers: Sampling layer, Grouping layer and Pointnet layer.

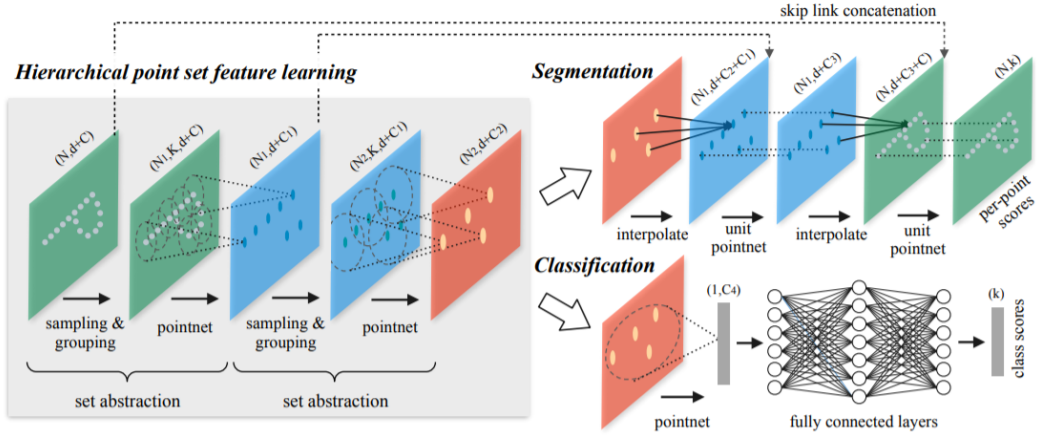
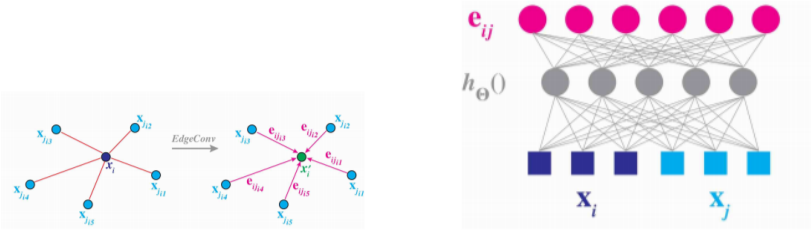


Figure 1.6: Pointnet++ Architecture

Recently, there was one more architecture inspired from Pointnet known as EdgeConv[15]. EdgeConv, instead of working on individual points, exploits the geometric structure by constructing a local neighborhood graph and applying convolution-like operation on the edge connecting the neighborhood pair of points. It thus has the property of translation-invariance and non-locality.



(a) EdgeConv Operation: The output of EdgeConv is calculated by aggregating the edge features associated with edges from each connecting vertex.

(b) Computing edge feature E_{ij} from a point pair x_i, x_j

Figure 1.7: Edge Conv Operations

Here the graph is not fixed as it is dynamically updated after each layer of the network, that is the k-nearest neighbors (kNN) of a point changes after each layer is calculated from the sequence of embedding.

EdgeConv (fig :1.7) applies channel-wise symmetric aggregation operation (ψ) on the edge features associated with all the edges emanating from each vertex. The edge features are defined as $E_{ij} = H(x_i, x_j)$, where H is some parametric non-linear function parametrized by the set of learnable parameters. Here, H will be MLP for the model. So the output of EdgeConv at the i^{th} vertex is $x_i^{'} = \underset{j:(i,j)\in\varepsilon}{\psi} H_\theta(x_i, x_j)$

The choice of edge function has a crucial influence on the properties of the resulting EdgeConv operation (table 1.1)

Edge Function	Property
$H(x_i, x_j) = \theta_j x_j$ $\theta = (\theta_i, ..., \theta_k)$	Classical Euclidean Convolution
$H(x_i, x_j) = H(x_i)$	Encoding global information of the local neighboring structure. (Pointnet)
$H(x_i, x_j) = H(x_j - x_i)$	Encodes the local information, considering the shape as collection of small patches and losing the global information.
$H(x_i, x_j) = H(x_i, x_j - x_i)$	Asymmetric edge function which combines the global shape structure(Xi) and local shape features $(x_j - x_i)$

Table 1.1: Edge Functions

Besides changing the input, there were approaches to change the convolution operators for point

cloud too. Parametric Continous Convolution[18] is a learnable operator that operates over non-grid structured data and exploits parametrized kernel functions that span the full continuous vector space. It can handle arbitrary data structures as long as its support relationship is computable. The continuous convolution operator is based on Monte-Carlo integration, so given particular function f and g with a finite number of input points y_i sampled from the domain so the convolution at any arbitrary point x will be approximated as $h(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy \approx \sum_i^N \frac{1}{N} f(y_i)g(x-y_i)$

Function g is parameterized such that each point in the support domain is assigned a value (Kernel weights), such parameterization is infeasible for continuous convolution, since the function g will be defined over an infinite number of points. The solution is to model g using parametric continuous functions using multi-layer perceptrons (MLP) as the approximator, because MLPs are expressive and capable of approximating continuous function, so $g(z;\theta) = MLP(z;\theta)$. The kernel function g(z,) spans the full continuous support domain while remaining parameterized by a finite number of parameters.

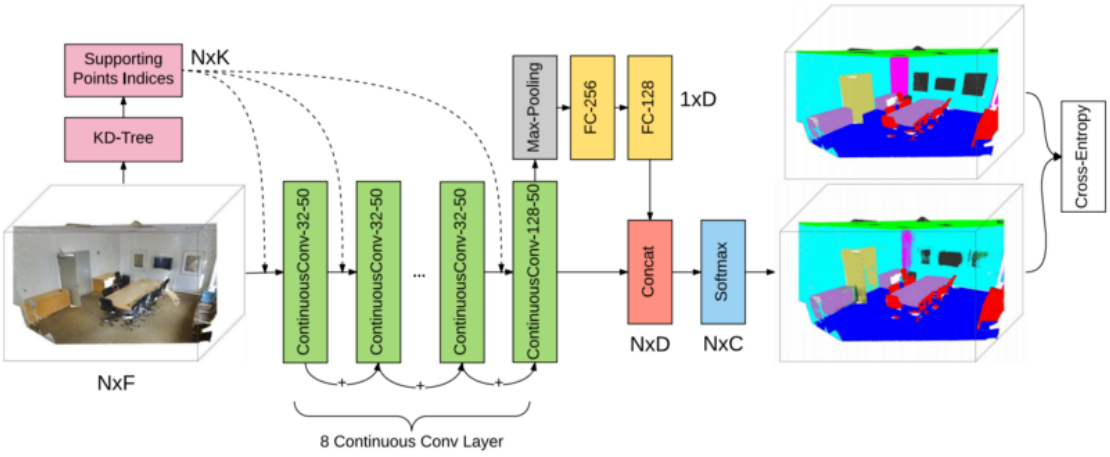


Figure 1.8: Continous Parametric Convolution Architecture

1.3 3D Instance Segmentation

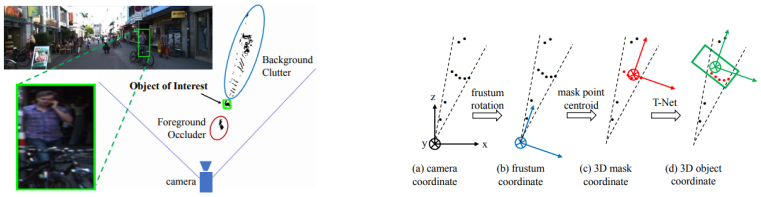
Instance segmentation is also an important problem for autonomous vehicle, where we are not only segmenting the points but also counting the number of objects per class and classifying the point belong to them. There had been a very few approaches in this direction but mostly with the use of RGB image along with LiDAR. Frustum Pointnet [10] is a novel framework for RGB-D data based object detection. Instead of solely relying on 3D proposals, this method leverages both mature 2D object detection and advanced 3D deep learning for object localization. It leverages mature 2D object detector to propose 2D object regions in RGB images as well as to classify objects. So, with a known camera projection matrix, a 2D box can be lifted to a frustum that defines a 3D search

space. We collect all the points in frustum to form a frustum point cloud.

The frustum may be oriented towards any direction leading to a large variation in the placement of point clouds. So we normalize the frustums by rotating them towards center view such that the center axis of the frustum is orthogonal to the image plane. Similar to Mask-RCNN, which achieves instance segmentation by binary classification of pixels in the image region, it does 3D instance segmentation using a Pointnet-based network on point clouds in frustums. It predicts the 3D bounding box center in the local coordinate system: 3D mask coordinates.

The network takes frustum point cloud as input and predicts a score for each point for how likely the point belongs to the object of interest. In multi-class detection case, it also leverages the semantics from a 2D detector for better instance segmentation. So the network can use this prior to finding geometries that look like the object of interest. In the architecture, they encode the semantic category as a one-hot vector (k dimensional for the pre-defined k categories) and concatenate the one-hot vector to the intermediate point cloud features.

After 3D instance segmentation, points which are classified as the object of interest are extracted. They further normalize the points to boost the translational invariance of the algorithm and translate the point clouds into a local coordinate by subtracting XYZ values by its centroid. The coordinate transformation and frustum rotation are critical for 3D detection results.



(a) Frustum proposal generation (b) Coordinate System for point cloud

Figure 1.9: Frustum Pointnet

Given the segmented object points, this module will estimate the object amodal oriented 3D bounding box by running a box regression Pointnet together with a preprocessing transformer network. A light-weighted regression Pointnet (T-net) is used to estimate the true center of the complete object and transform the coordinate such that the predicted center becomes the origin. The T-net can be thought of as a special type of spatial transformer network (STN). The box estimation network predicts amodal bounding boxes for objects given an object detection in 3D object coordinate. They parametrized the 3D bounding box by its center (c_x, c_y, c_z), size (h, w, l) and heading angle θ.

The center residual predicted by the box estimation network is combined with previous center residual from the T-net and the masked points centroid to recover an absolute center.

1.4 Capsule Network

Convolutional neural networks have been a dominant approach to object detection but still have many drawbacks. A notable example is CNN faces difficulty in generalizing to novel viewpoints. Capsules[4] solves some inefficiences by converting pixel intensities into vectors of instantiation parameters of recognized fragments and then applying transformation matrices to the fragments to predict the instantiation parameters of larger fragments. Here each layer is divided into many small groups of neurons called capsules. A capsule [15] is a group of neurons which performs computation on the input and encapsulates it into a vector which is capable of representing a different property of the same entity.

In the capsules, the length of the activity vector represents the probability that the entity exists and orientation represents the instantiation parameters. Capsule network is equivariant to the input instead of being invariant. So if the object is rotated at an angle, the capsule will be able to identify that the object is rotated at an angle without extra augmented training required. CapsNet preserves the geometric dependence of features to be translational invariant.

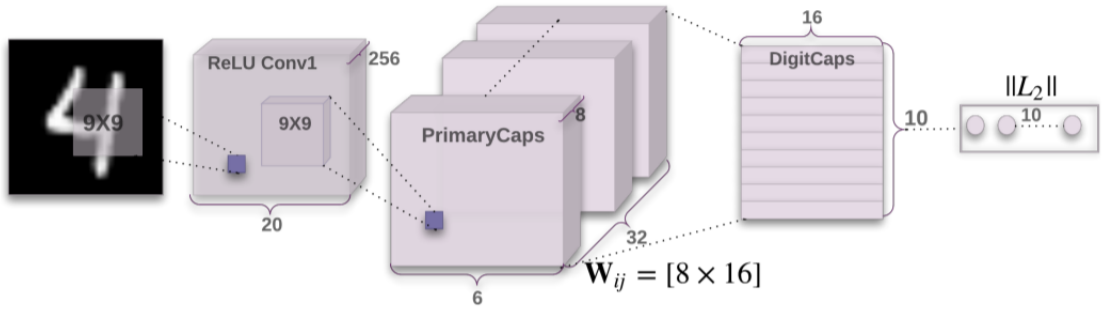


Figure 1.10: CapsNet architecture with 3 layers for MNIST data

1.4.1 Routing

The output of a capsule is a vector making it possible to use powerful dynamic routing to ensure all the output of the capsules are sent to an appropriate parent. Each parent capsule computes a prediction vector by multiplying its output by a weight matrix. If the prediction vector has a large scalar product with the output of a possible parent, there is a top-down feedback which increases the coupling coefficient for that parent and decreases for other parents[15].

Where c_{ij} is the coupling coefficient and is determined by the iterative dynamic routing process. The coupling coefficient between capsule i and all the capsules in the above layers sums up to 1 and is determined by a routing softmax whose initial logits b_{ij} are the log probabilities that capsule i should be coupled to capsule j.

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{1.1}$$

The log priors can be learned discriminatively at the same time as all other weights. The agreement is measured as the scalar product $a_{ij} = v_j \hat{u}_{j|i}$. The agreement is treated as a log likelihood and is added to the initial logits b_{ij} before computing the new values of the coupling coefficient.

For all the first layers of capsules the total input s_j is a weighted sum over all prediction vectors $\hat{u}_{j|i}$ from capsules in the layer below and is produced by multiplying the output u_i of a capsule in the layer below with a weight matrix W_{ij}

$$s_j = \sum_i c_{ij} \hat{u}_{j|i}, \qquad \hat{u}_{j|i} = W_{ij} u_i \tag{1.2}$$

1.4.2 Squash

The length of the output vector of a capsule represents the probability that the entity represented by the capsule is present in the current input. A non-linear squashing function is used to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1. Squash is the activation function applied to the capsule. For a capsule total input s_j the output v_j is defined as

$$v_j = \frac{\| s_j \|^2}{1 + \| s_j \|^2} \frac{s_j}{\| s_j \|^2} \tag{1.3}$$

1.4.3 Results

CapsNet achieved state-of-the-art results on MNIST dataset and also proved to learn more robust representation for each class than a traditional CNN. It also achieved more accuracy on affine transformed dataset as compared to the traditional CNN. Although it achieved a 10.6% error rate on CIFAR10[5] dataset when tested with an ensemble of 7 similar models, it is similiar to the error rate when standard CNN were applied to CIFAR10[17].

1.5 Application

Lots of applications uses lasers to scan and analyze the environment or product entities. When they use laser they create a demand for 3D object detection algorithms for better analysis. The industry where 3D object detection are of critical importance are:

1. **Self Driving Industry**: 3D object detection is critical to the perception stack of a self driving car to detect and localize the obstacles in real world coordinates. They need to correctly

identify vehicles and pedestrians in very possible scenario to avoid unexpected outcomes. With new LiDAR in the market, they enable the vehicle to see up to 200m, helping the vehicle to maintain a high speed and still have enough time to respond to unexpected states.

The main advantage of LiDAR is the vehicle can drive at night with the same precision in the day irrespective of the lighting conditions. This makes the autonomous stack more stable and reliable while driving during night or sudden change in light intensity.

2. **Urban Planning and Surveys**: LiDAR scans are used more often to analyze urban areas, remote terrains, damage inspections etc. Drones are now widely used to inspect buildings, bridges, caves etc which are not safely reachable for humans. The 3D object detection algorithm can segment the buildings, bridges, vehicles creating a faster pipeline for analysis for researchers.

3. **Health care**: 3D reconstructions are popularly used in diagnosis systems in digital microscopy and analysis of CT scans [8, 9].

4. **Manufacturing industry**: laser scanners are used to check fault in productions in real time. They are essential for quality assurance in industries producing for aerospace, defence, and medical sectors. 3D object detection algorithms can be used for localizing the object pose for the robot to make it more efficient for grabbing and placing in difficult situations.

Chapter 2

Proposal

2.1 Aims

We believe a independant LiDAR perception stack will be more robust to the challenges faced by autonomous vehicles given its scan invariance to the environment lighting condition. If we summarize the section 1, most of the methods just evaluate their model on a point cloud with less than ten thousand points and also, there is no method that did 3D semantic segmentation for a large point cloud with more than 60 thousand points per frame. Given the LiDAR point cloud will have at least have more than 60 thousand points per frame, we need a more scalable algorithm to deal with this scenario. Learning the current problems faced we will try to focus on the following aspects during the thesis:

1. **Object segmentation using point cloud**

   Considering scenarios where lighting conditions are not appropriate for camera model to work robustly, we need a pipeline that should be invariant to the environment conditions to work appropriately.

   Implementing a pipeline that works on LiDAR data alone will provide us the same result irrespective of the lighting conditions. The pipeline will be more reliable for driving at night as the scans will be not affected by the day/night conditions. Most of the autonomous vehicles pipeline also deploys two independent models for perception using camera and LiDAR and use their output to cross-verify the detections.

   Here we will mostly focus on **segmenting vehicles and pedestrians** from the lidar point clouds. More challenging scenarios involve segmentation of roads, lanes and traffic signals. It all depends if we are able to preprocess and extract ground labels for the data.

2. **LiDAR based instance segmentation**

Along with segmentation, instance segmentation is also an important issue for LiDAR data. We believe a centroid is a distinctive character for each instance in the point cloud. We will try to predict the segment id as well as a vector $(\hat{x}, \hat{y}, \hat{z})$ which will point towards the centroid of the object.

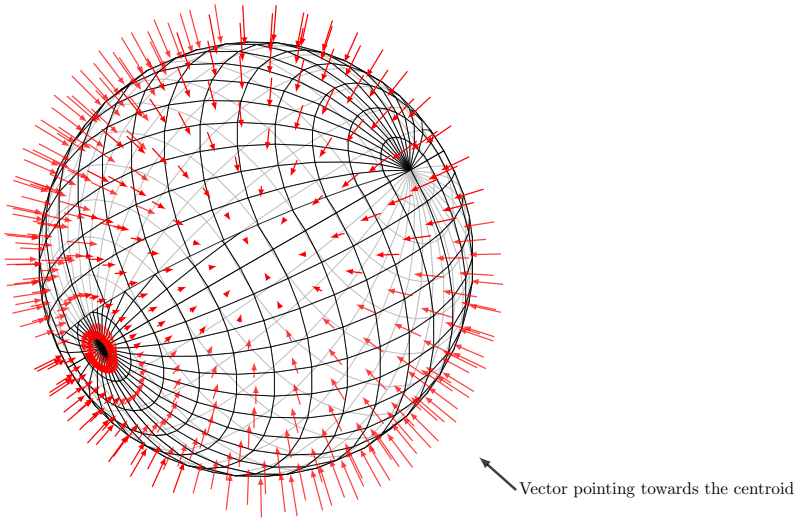


Figure 2.1: Sphere point cloud mesh

3. **Object detection using 3D capsule network**

   The previous proposed methods use convolutions to learn features, but CNNs fails to address the spatial relationship between features because of max-pooling, which is also the important symmetric functions used by the state-of-the-arts methods [11, 12]. Also, the convolutional architectures also need huge datasets to generalize.

   We propose a model architecture of extension of Capsule network [15] for 3D object classification. We believe that capsules will help us preserve the orientation and spatial relationship of the extracted features.

2.2 The Proposed Approach, Experiments and Evaluations

1. **Object segmentation using point cloud**

   We believe that the a good classifier needs to learn both the local and global features in order to classify a object well. We propose implement a fusion of a global and hierarchical network where the global features takes care of the global scenario and the hierarchical network forms clusters of KNN with the seeds or with the voxelized input. We will concatenate the features to produce the final segmentation output. We believe that this will give the network more

knowledge about the local as well as the global features present and will help to produce better segmentation results.

We will try to test with weather conditions where the projection methods might fail by simulating weather conditions in Carla simulator or by augmenting uniform noise points in the frame. We will evaluate this experiment with the projection methods[13, 7, 6]

We will carry experiments with Kitti and Carla dataset and try to compare it with the existing methods. We will try to evaluate the experiment results using mean pointwise accuracy and per-class pointwise accuracy.

2. **LiDAR based instance segmentation**

   We will try to predict a single vector directing to the centroid for the group of points belonging to the same instance. We will try to predict the segment id as well as a vector ($\hat{x}$,$\hat{y}$,$\hat{z}$) which will point towards the centroid of the each instance (fig 2.2, 2.3). We plan to include the position information (x, y, z) along with the high level segmentation features to help the network learn to differentiate between objects based on their locatians in the frame.

   We will carry out experiments on Kitti[3] dataset and as there does not exist any other method to do instance segmentation with LiDAR we will not have a baseline to compare to. We will evaluate the model with mean pointwise accuracy and loss with mean square error for the vectors and cross-entropy with segmentation ids.

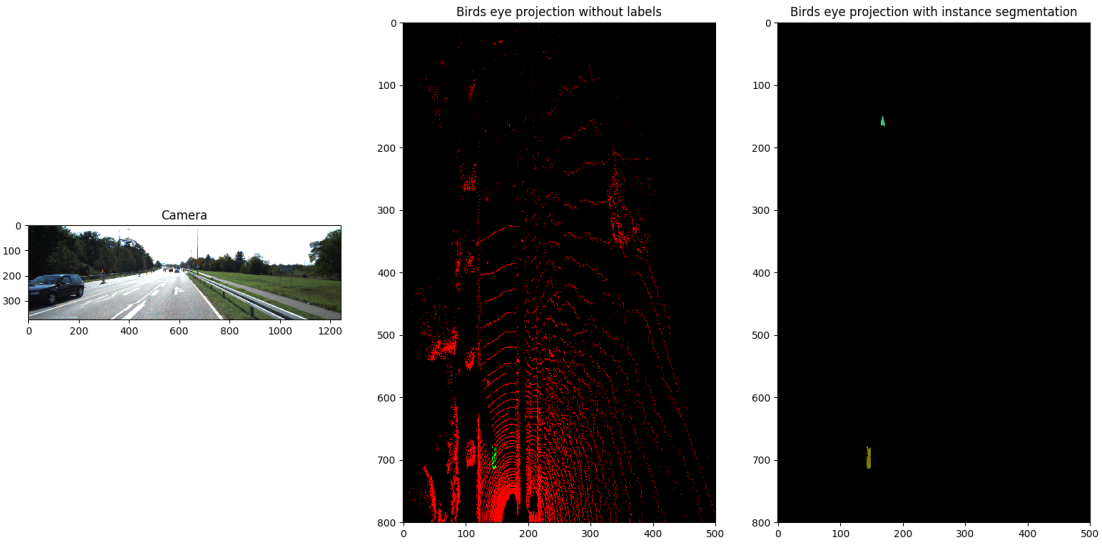


Figure 2.2: Instance Segmentation on Kitti dataset

3. **Object detction using 3D capsule network**

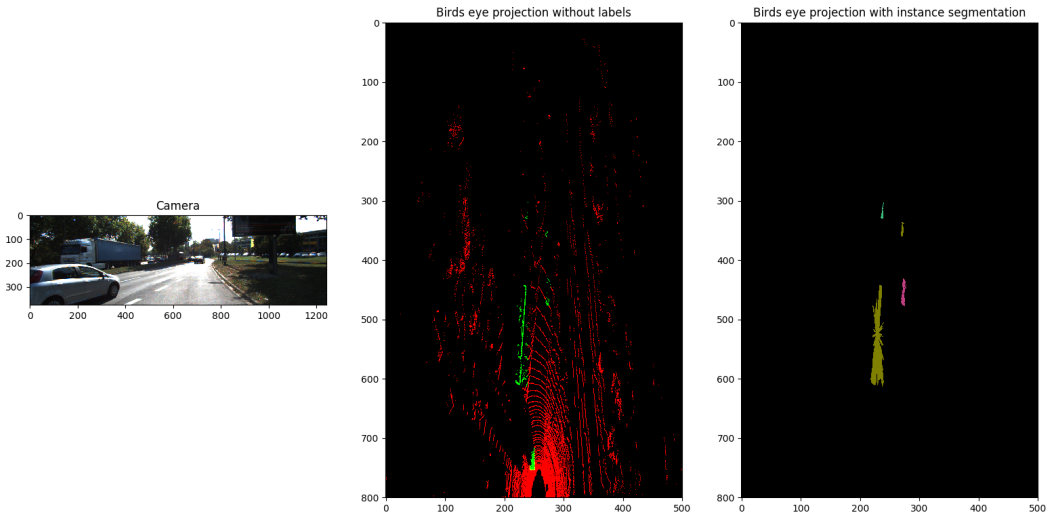


Figure 2.3: Instance Segmentation on Kitti dataset

As of there are no papers of applying CapsNet to 3D data, we propose to use edge features of the point cloud as the input feature to the CapsNet and carryout classification experiments with ModelNet10 dataset. As CapsNet preserves the spatial relationship and orientation of the extracted features, we believe it will help the network learn faster with less samples and also to generalize to multiple viewpoints which is a dominant problem for 3D point cloud. We will try to experiment with rotationally augmented samples of Modelnet10 top compare the accuracy with CapsuleNet and Pointnet[11]. We will try to test the architecture with ModelNet10 and if we get interesting results we will try to carry out experiments with Kitti dataset.

2.3 Dataset

We will be using multiple dataset for training and evaluation of out methods:

1. **KITTI Vision Benchmark Suite**

   Kitti[3] have been the most popular dataset in the autonomous vehicle research industry. The dataset was captured by driving around the mid-size city of Karlsruhe, in rural areas and on highways. The 3D object detection benchmark (fig: 2.4) consists of 7841 trained images and 7518 test images as well as their corresponding point clouds. For evaluation they use precision-recall curves and for ranking the methods they use average precision. The annotated objects in the dataset are cars, vans, trucks, pedestrians, person sitting, cyclists, and trams.

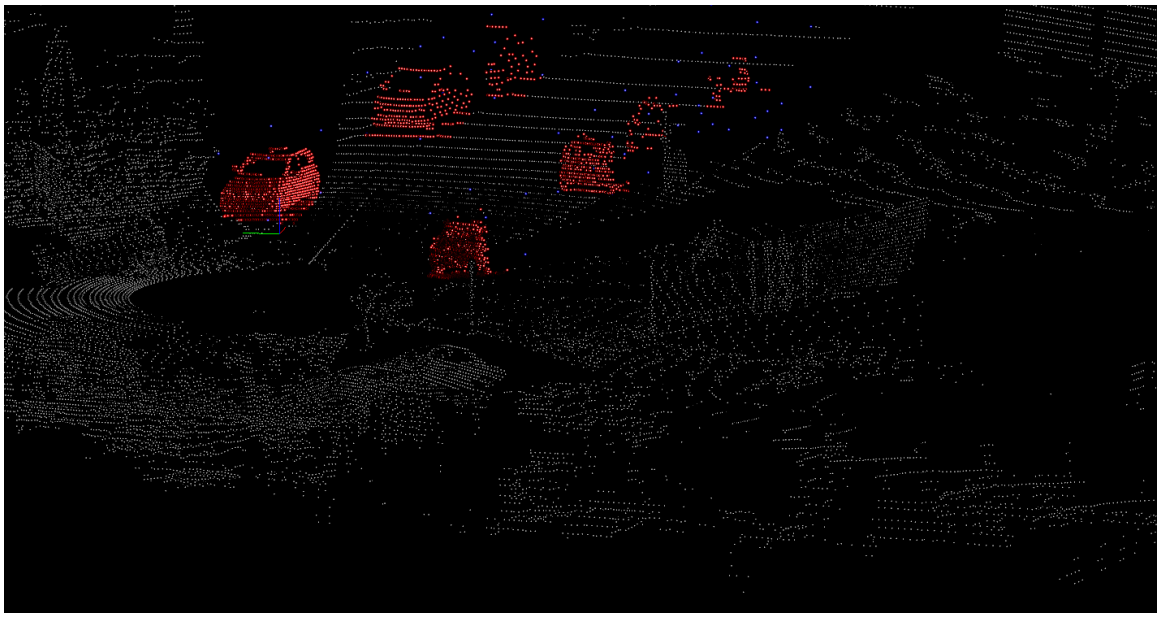


Figure 2.4: Point cloud semantic labelling in Kitti

2. **Carla Simulator**

   Carla[2] is a open source platform to support development, training, testing, and validation of autonomous urban driving systems. The simulation platform supports flexible specification of sensor suites and environmental conditions. It gives us the flexibility to gather as much data as we want from different scenarios, weather conditions and traffic conditions.

   Carla has recently released a rotating Lidar feature with ray-casting. The points are computed by adding a laser for each channel distributed in the vertical FOV, then the rotation is simulated computing the horizontal angle and doing the ray-casting. When server sends a packet with all the points the physics do not update, so all the points in a packet reflect the same static scene. Carla also provides semantic segmentation labels for the front camera view which can then be projected on to the point cloud to get semantic labels for each point cloud present in the front view (fig: 2.5). There are a total of 13 classes annotated right now.

3. **ModelNet40**

   ModelNet[16] provides comprehensive clean collection of 3D CAD models for objects. It is available in two subsets of 10 popular class subset (ModelNet10) and 40 class subset (ModelNet40). ModelNet had been popularly used to benchmark the models on point clouds and also analyze the cost variations after certain oper ations[15, 12, 15]. We will be using ModelNet to test Capsule Net operations before applying to the large point clouds.

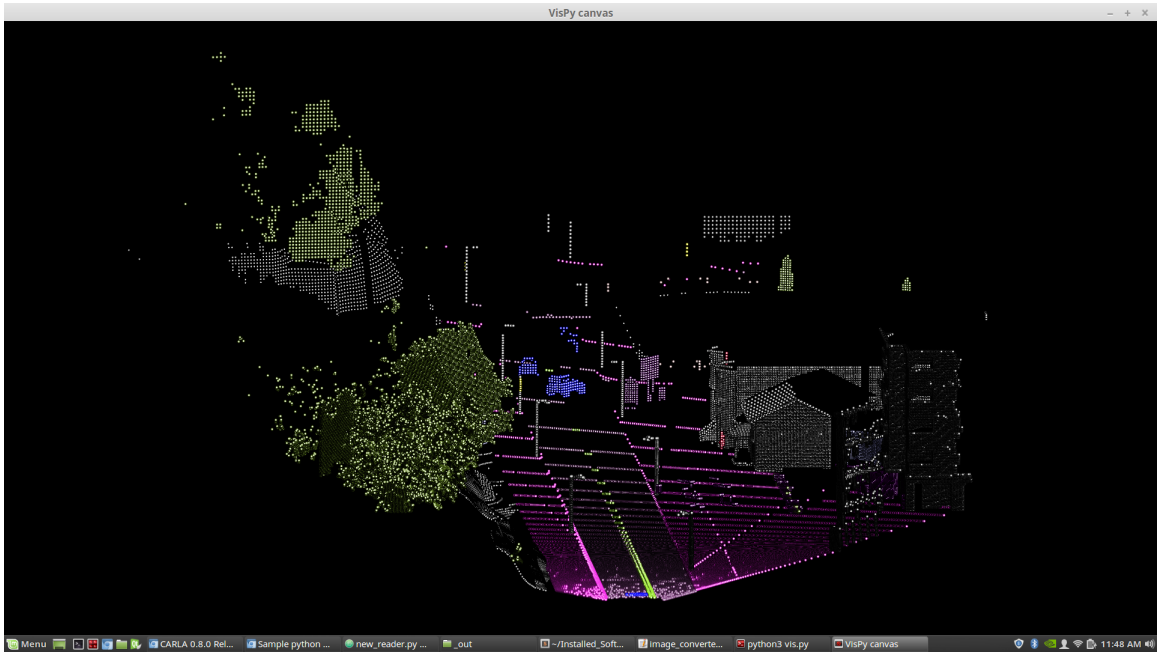


Figure 2.5: Point cloud semantic labelling in Carla

2.4 Results

2.4.1 Pointnet

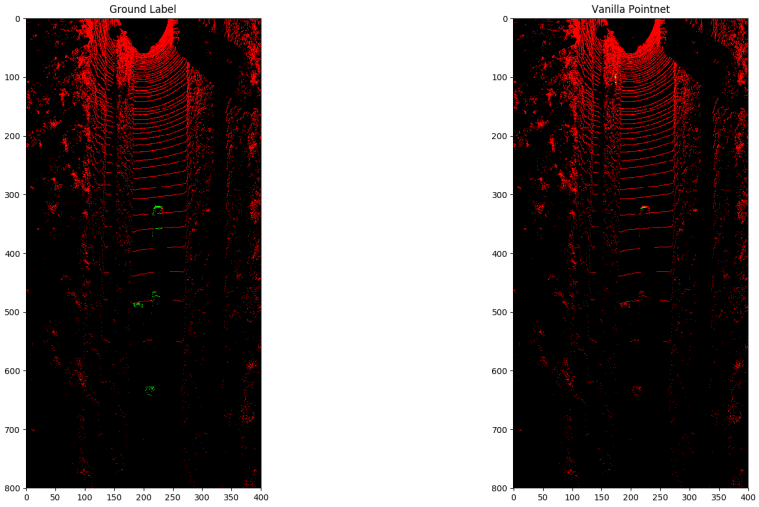


Figure 2.6: Pointnet results

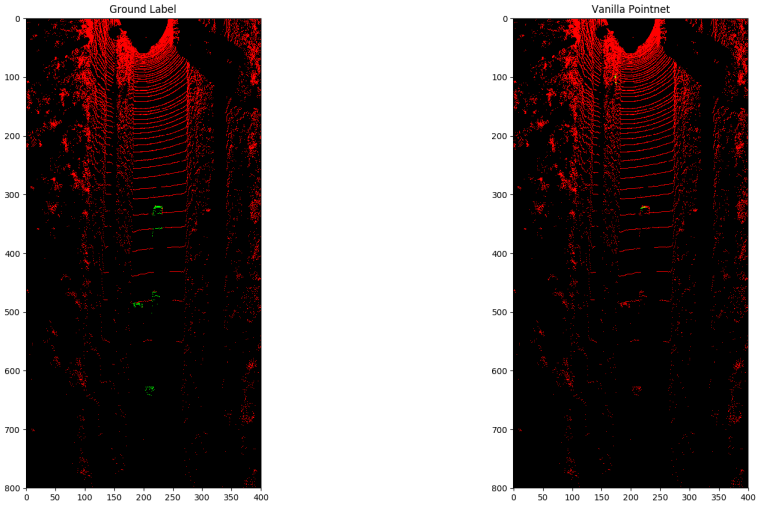


Figure 2.7: Pointnet results

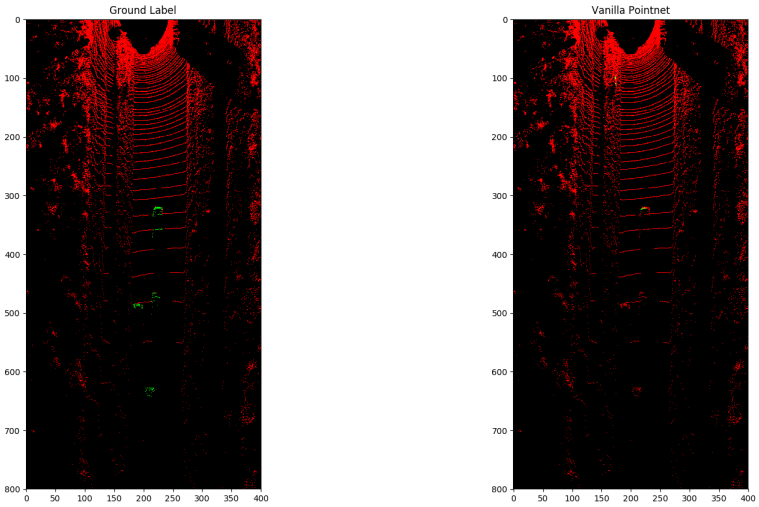


Figure 2.8: Pointnet results

2.4.2 Pointnet++

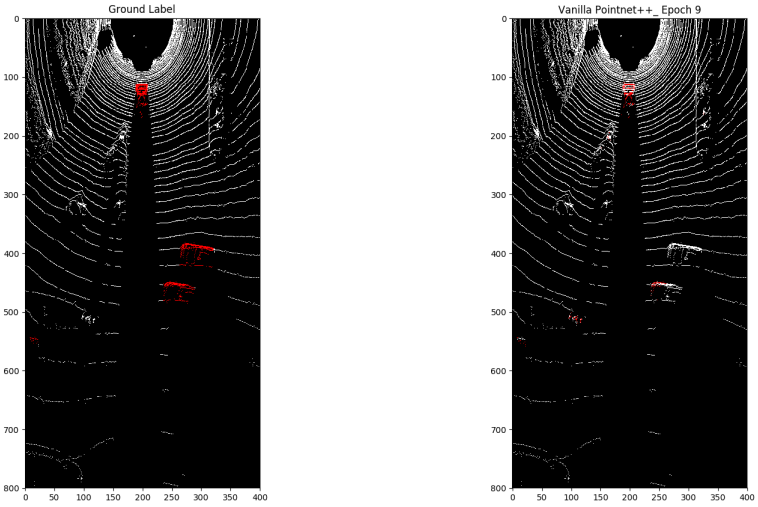


Figure 2.9: Pointnet++ results

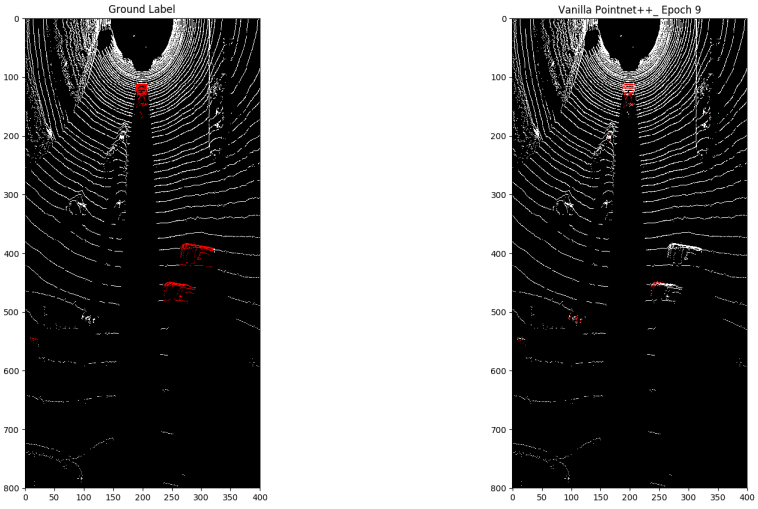


Figure 2.10: Pointnet++ results

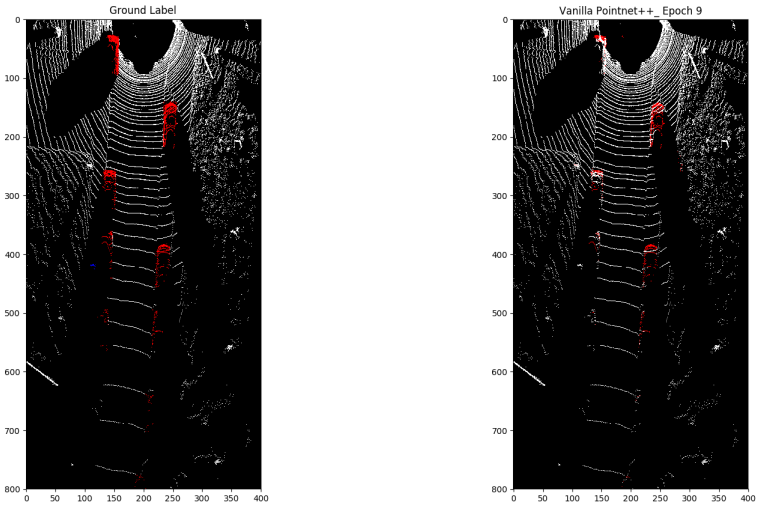


Figure 2.11: Pointnet++ results

2.4.3 EdgeConv

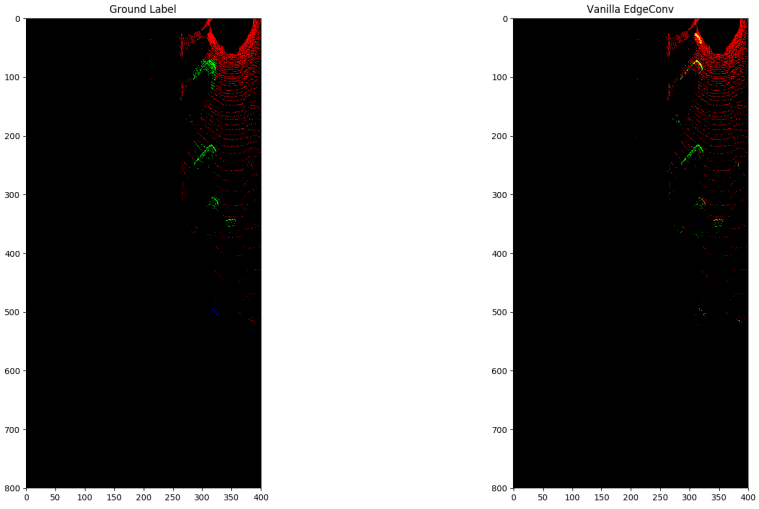


Figure 2.12: EdgeConv results

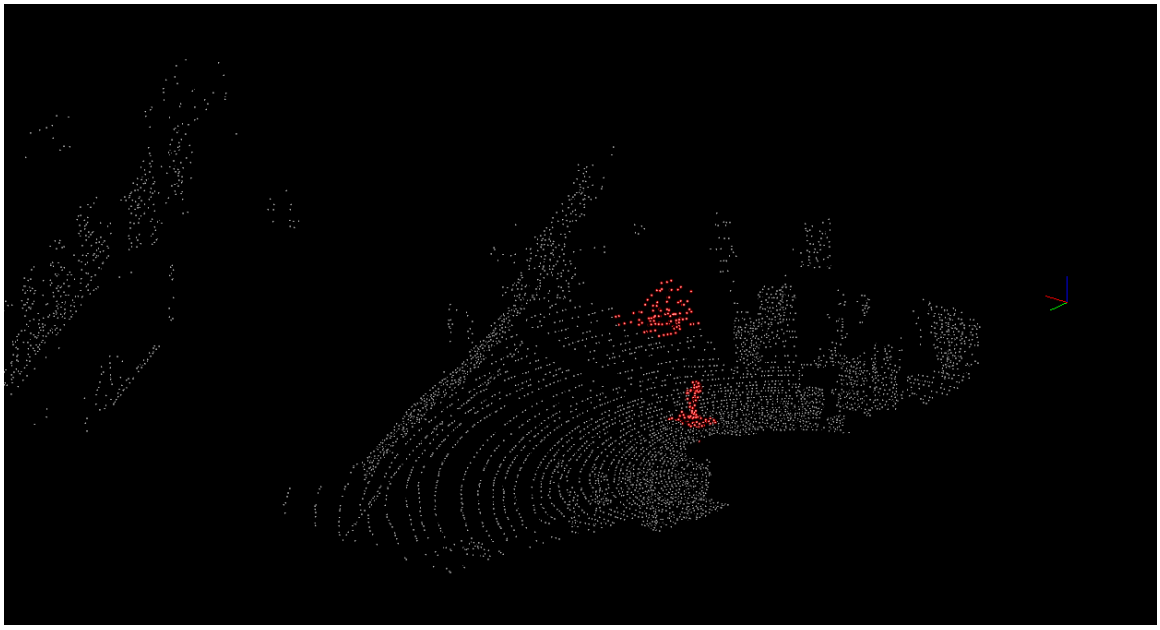


Figure 2.13: EdgeConv results

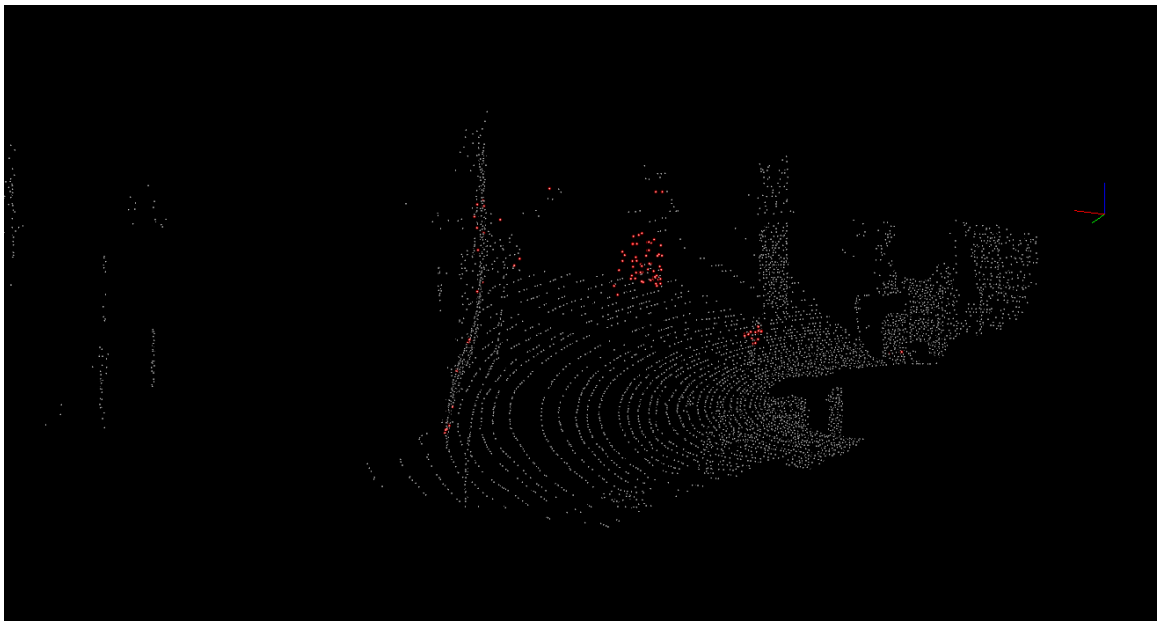


Figure 2.14: EdgeConv results

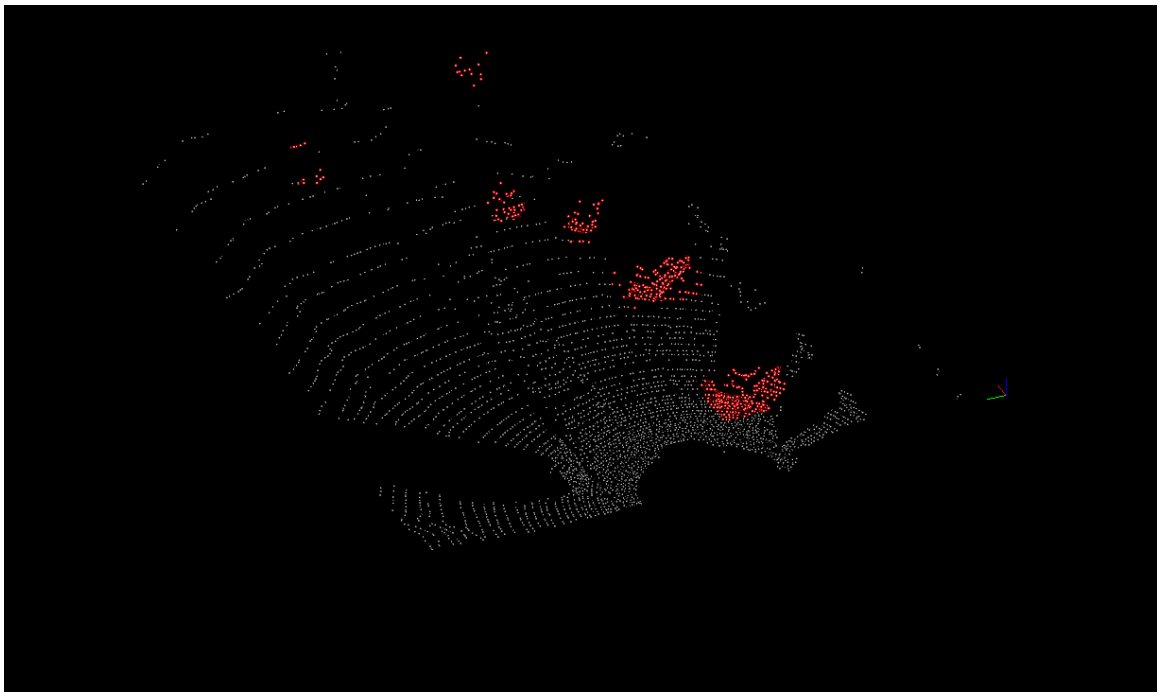


Figure 2.15: EdgeConv results

Bibliography

[1] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016.

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[3] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3354–3361, Washington, DC, USA, 2012. IEEE Computer Society.

[4] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *ICANN*, 2011.

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[7] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Peter M Maloca, J. Emanuel Ramos de Carvalho, Tjebo Heeren, Pascal W Hasler, Faisal Mushtaq, Mark Mon-Williams, Hendrik P.N. Scholl, Konstantinos Balaskas, Catherine Egan, Adnan Tufail, Lilian Witthauer, and Philippe C. Cattin. High-performance virtual reality volume rendering of original optical coherence tomography point-cloud data enhanced with real-time ray casting. *Translational Vision Science Technology*, 7(4):2, 2018.

[9] Jeremy P Metcalf and Richard C Olsen. Photogrammetric point cloud and lidar fusion for improved building delineation (conference presentation). In *Algorithms and Technologies for*

Multispectral, Hyperspectral, and Ultraspectral Imagery XXIV, volume 10644, page 106440E. International Society for Optics and Photonics, 2018.

[10] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

[11] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[12] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[13] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: Real-time 3d object detection on point clouds. *CoRR*, abs/1803.06199, 2018.

[14] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[15] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[16] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920. IEEE Computer Society, 2015.

[17] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[18] Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds.