Report on VOTE3DEEP first stage implementation: Understanding, visualising and feature extraction from Point cloud data

Objective:

The main objective of the first stage of the implementation is to read in the raw point cloud data, represent the point cloud data as grid cells and extract the feature vectors of the grid cell representation of the point cloud data. For achieving this objective the problem is divided into several substages and a implementation of each of the substage is achieved. The following, Gives a break down view of the various substages of the implementation.

Step 1: Understanding the KITTI dataset
Step 2: Reading the point cloud data from the KITTI dataset.
Step 3: Dividing the point cloud data into grid cells.
Step 4: Grid cell Feature extraction

In depth description of how each substage of the implementation is carried out will be discussed in the sections to follow.

Introduction to 3D object detection:

Object detection forms one of the important part of the current mobile robotics systems. For abling the perception capability to robots many vision techniques are in use which enable the systems to understand the environment of the robot. The input to the vision technique comes from the vision sensors on the mobile robot. Number of sensors are made available to the mobile robots to provide great information on the environment its envisioning. Some of the widely used sensors are the cameras, RGBD, stereo, LIDAR and thermal sensors.

Lately, the 2D object detection from the planar images of the real world have been the state of the art in the object detection arena. But with the need for the vision systems to make visualise the real world for navigating and understanding helped pave the way for the 3D vision sensory systems. Additionally, Camera data is typically directional, meaning it's only into one direction so are unreliable as single data source. Currently, the 3D vision sensors and techniques based of this sensors is being pursued as one of the promising area for enabling robust 3D vision to robots and autonomous vehicles. The self driving car is one such research instance which visualises the environment around the car using the 3D LiDAR sensor. The Lidar sensory information is used to visualise the environment and autonomously navigate through the environment. Some applications use RGB-D and stereo sensors to 3D visualise the objects and navigate the 3D environment. The various 3D vision sensor systems have their own limitations and are restrictive to a specific set of applications.

LIDAR sensors uses light in the form of pulsed laser to measure the ranges in the environment it sees. The differences in the laser return times and wavelengths are used to

generate a 3D point cloud of the world. This point cloud data is the continuous stream of input to the vision system of the robot. One main advantage of using the LiDAR based sensors is that the sensor is reliable in harsh environments like snow, sleet and rain The multiple beam approach of Velodyne's LiDAR sensors with laser beams with millions of laser beams at different angles enables to find "holes" in-between the snowflakes to "see" the environment. Many techniques exist which use the LiDAR point cloud stream to carry out the object detection. VOTE3DEEP is one such 3D perception technique which uses the point cloud data from the LiDAR sensor to do the task of object detection.

VOTE3DEEP is a neural net based implementation. The architecture of the neural network essentially consists of 3 3D CNN (convolutional neural network) layers and one receptive field layer. The neural network is trained for detection on primarily three classes of objects namely person, car, and cycle. KITTI vision benchmark suite velodyne point cloud dataset is used as the training and testing dataset for the 3D object detection. The dataset consists of thousands of point cloud frames which are 3D annotated with the object locations. The point cloud data from the dataset have on average 100k points per frame of the point cloud. Finally the network trained on this dataset is tested and the methodology is evaluated on the KITTI object evaluation tool.

For arriving at a full pledged implementation of the vote 3 deep methodology and also for making the implementation compatible for GPU processing the entire methodology is divisioned into several stages Extracting feature vector, implementing the core architecture, single class training, fine tuning, evaluation, extending to multiclass object detection, including additional features for tunnel boundary detection, training on custom dataset and extending the implementation to GPU. The various stages of the methodology will be studied in depth and will be implemented. This report discusses about, how the first stage of the VOTE3DEEP is carried out and provide detailed coverage of how the feature extraction stage of the implementation is carried out.

Methodology:

This section discuss in detail about the first stage of the vote 3deep object detection algorithm. In the first stage of the object detection task which is obtaining the feature vector representation of the point cloud data the implementation is carried out in several stages. Firstly, the details of the implementation will begin with the short note on the KITTI object detection dataset and it is followed with details on methodology employed and implementation details of all the sub stages of the implementation. The first substage of the implementation is reading the point cloud data and is followed with representing the point cloud data as grid cells and finally extracting the feature vector information for each of the grid cell in the data. With this, Let us now dive into the details of the first stage of the implementation.

Introduction to KITTI 3D Object Detection Dataset:

The kitti 3D object detection dataset consists of 7481 training images and 7518 test images which include a total of 80256 labeled objects. Along with the velodyne point cloud

training frames the bounding box of the labels corresponding to the point cloud frames are also included in the dataset. For training vote3deep the labels and the point cloud frames of the 7481 images are used. The point cloud data files are stored in the form of N*4 float matrix into binary files and the label files are stored as text files where in for each object annotation have fourteen set of feature values which give information on the bounding box dimensions, truncation, occlusion, and many other important information on the labeled object. The detailed coverage on the point cloud binary files is discussed in the following section. Details on the various features in the label files for each object will be including the second stage of the project. In which the bounding box information of the objects in the point cloud will be forming the base of the second stage of the implementation.

Reading Point Cloud Data:

As discussed in the introduction to KITTI object detection dataset, The Point cloud data in the KITTI 3D object detection is stored in form of binary files. Each of the the binary file have four entries for each point in the point cloud. The first three corresponds to the x,y and z location of the point with respect to the sensor frame and the fourth value corresponds to the reflectance information of the point in the point cloud. The reflectance information of the point cloud stores the intensity value of each point in the point cloud frame. The following figure shows a point cloud frame of the lidar stream visualised using the 3D visualization tools in python framework.

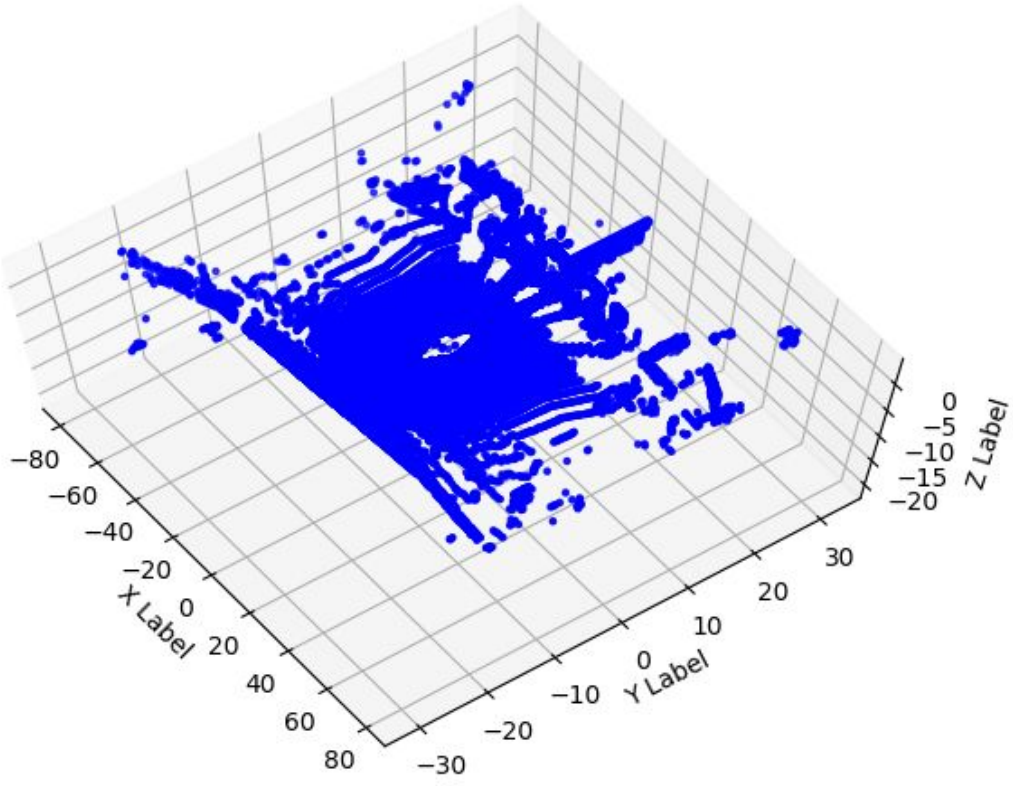

Figure 1: 3D visualization of the LIDAR point cloud data.

Point cloud data to grid cells:

The information on extremes of each frame of the point cloud data is collected i.e maximum and minimum value along each of the X,Y and Z coordinates of the points in the frame. Then the information on the extremes along with the resolution is used to calculate the grid cell size the x,y and z dimension of the grid cell are Obtained by dividing the range along each dimension with resolution. Now for each point in the frame the difference between each coordinate and the minimum value in that dimension is calculated to obtain the distance of the point in that dimension. This value is divided with the grid cell width along the dimension to arrive at the cell to which that particular point belongs to in that dimension.

This process is repeated for all the dimensions of the point and all points in the point cloud frame. The cell/bin information of this points is stored in an array. This makes the four data variables we have for the point cloud seven. The three new entries talk about the x, y and z bin to which the point belongs. By repeating the process of allocating points to grid cells the point cloud data of the lidar scans the points of each of the point cloud binary file are made organised. The organised grid files are then finally used to extract feature vectors representing the point cloud corresponding to each grid cell. The resolution parameter is one of the key parameter which decides the number of grid cells that will be formed along each dimension and also resolution will be one of the interesting parameter that will be fine tuned for carrying out training optimisations.

Sorting the data and Extracting feature vectors of grid cells:

The grid cells information of the point cloud data is the input to the feature vector implementation. This grid cell information is converted into feature vectors in several stages. Firstly, the grid cell data of the point clouds is organised by sorting the grid file data based on the bin values. The X bin value of the grid cells forms the first parameter based on which the sorting is carried out and the y bin and z bin parameters values are used for breaking ties while sorting the grid cell data. By sorting the grid cell data, the data is made more uniform and representative to the previously random data. The final output of sorting the grid cell data is a grid cell array with the entries sorted bin wise. Now the ordered data obtained by sorting is the input to the feature extraction algorithm which forms the second step of the implementation. In this stage of the implementation the sorted grid cell data is processed to get the feature vector for points in each grid cell.

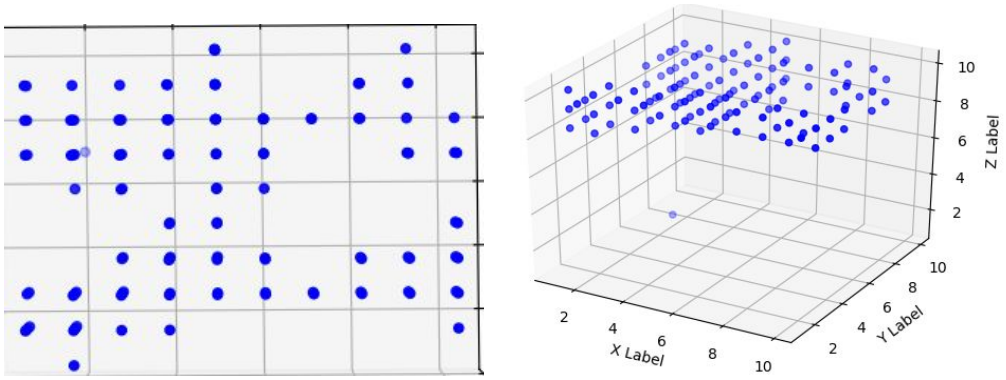

Figure 2: The grid cell and feature vectors of the raw point cloud data. The first and the second figures represent the top view (XY plane) and the orthogonal view of the feature vectors.

Each feature vector of the grid cell have in total eight parameters representing the points enclosed in the grid cell. The parameters at position 1,2,3 represent the x,y and z bin location of the grid cell and the parameters 4,5,6 represent the mean value of the points in the grid cell along the x, y and z dimension. The parameters at position 7,8 reflect the intensity or the reflectance information of the points in the point cloud. This feature vector information of every cell in the point cloud frame is obtained and stored in the grid cell files. This process is repeated for all the point cloud frames in the dataset. The feature vectors of the frames in the dataset forms the starting point or the input to the 3D Convolutional neural network. The feature vector information along with the labels data is used to train the neural network framework for 3D object detection from point cloud.

Neural Network architecture:

The vote3deep neural network architecture comprises of three CNN layers. For each of the CNN hidden layer except the last layer have a total of eight Convolutional filters of size 3*3*3. The last layer of the network has a single 3*3*3 kernel which acts as the receptive field for the network. As discussed earlier the methodology of the vote3deep involves class specific networks as the computation time for computing the detection on large bounding boxes for small objects is large. So, class specific networks forms one of the core principle for the design of architecture of neural network. In addition the class specific neural networks increase the inherent parallelism in the code thus resultantly, enable faster computation. In this implementation of the vote3deep the network will be trained to detect persons from the input point cloud frame.

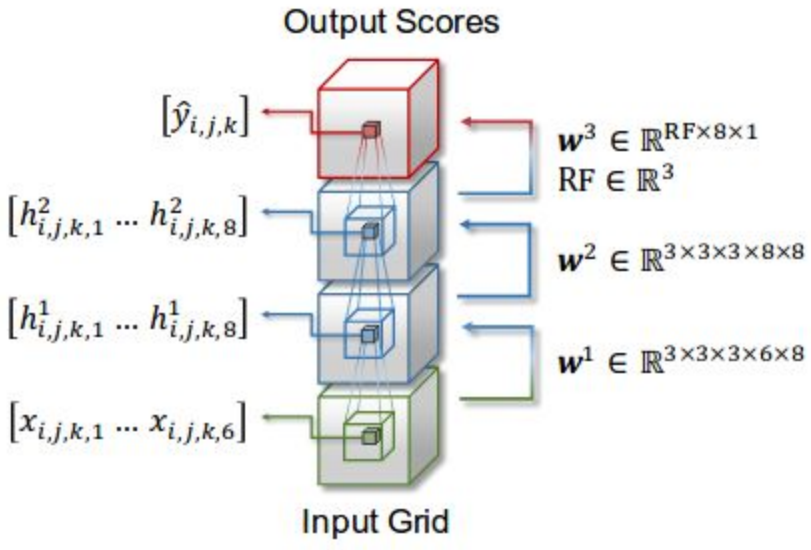

Figure 3: The architecture of the vote3deep implementation

The receptive field in the implementation is essentially is a convolutional kernel the size of the kernel is obtained by first calculating the 95th percentile ground truth bounding box over the training set and approximating the bounding box to the closest kernel size. This eliminates the need to regress the bounding box size by fixing the bounding box for each class in the dataset. In addition to this each CNN layer of the neural network is followed by a RELU nonlinearity to maintain the sparsity in the layers of the neural network.

Batch_size	2
number_epochs	101
Loss function	Mean square error, hinge
Optimisation algorithm	Adaptive movement estimation
Learning rate	1e-4
Training time for the biggest model	8 days (7481 frames)

Table 1: The parameters of the neural network architecture

Voting Scheme implementation:

The voting scheme employed in this methodology lets the non zero feature vectors in the grid cells cast their votes, weighted by filter weights to the surrounding cells in the feature grid. The reason behind choosing nonzero feature vectors is discussed in the upcoming paragraph. The voting weights which are obtained by flipping the convolutional weights are applied and convolved on the input grid. The final convolution result by the voting operation consists of the

cells with the accumulated votes. This voting operation on the feature vector gird is performed in each hidden layer of the network.

The LIDAR point cloud data from the sensor comprises of lot of sparse point cloud information of the environment. This point cloud data can be directly forwarded to the neural network for training and testing. But, looking into the math behind performing the above processing renders an important inference on the basic behaviour of voting scheme on point cloud data. As, described earlier the point cloud data had a lot of sparse points along with very few point clusters of high density. Given this fact when voting scheme is implemented on the point cloud data majority of the process operations involves multiplication with zero. The figure 4 below which is a 2D version of voting scheme visualises the effect of voting scheme on sparse point clouds.

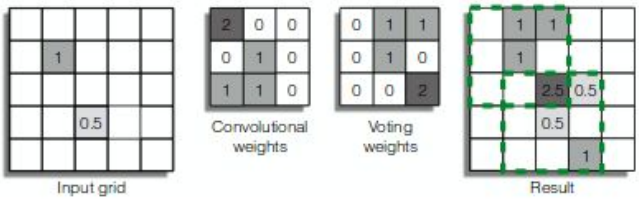

Figure 4: The voting implementation on the feature vectors. If voting scheme employed on all feature vectors only the non zero feature vectors participante in the voting scheme. The remaining of the feature vectors are multiplications with zero.

RELU: How RELU mainitains the sparsity:

After each convolution operation using the voting weights the output consists of dilated representation of the nonempty cells. As this procedure of voting is applied along the depth of convolution the sparsity of the point cloud data will be lost. So in order, to maintain sparsity in the convolution layers layers the RELU non linear activation is performed on the output from the previous convolution. This operation get rid of unwanted features which finally might result into a false detection and also maintains the rate at which the dilation is carried out the filter weights. In general the dilation caused by the filter weights is proportional to the receptive field size of the corresponding filters.

L1 sparsity regulariser:

The RELU activation helps to maintain the psarsity in the feature vector grid at each and every layer and also maintains good representation of the important features. In addition to the RELU functionality the L1 sparsity acts as an additional regulariser by increasing the sparsity and discarding the uninformative features.

Point cloud data Statistics:

The point cloud data from the KITTI dataset consists of lot of information. Each frame of the LIDAR point cloud on average consists of 110k points. This point cloud information is processed to obtain the feature vector information for each cell in the grid. The label data corresponding each point cloud frame talks about the location, size and various other features of objects in that

particular frame. While processing this information and making the data ready for training the following important features are observed in the data

	Number of objects	**Mean Size of the objects**	**95 percentile size of the objects**	**Approximate Kernel resolution**
cars	28742	[1.53,1.65,3.83]	[1.77,1.80,4.53]	5*5*5
pedestrians	4487	[1.73,0.59,1.74]	[1.86,0.86,2.02]	3*3*3
Cyclists	1627	[1.74,0.67,0.77]	[1.90,0.93,1.16]	3*3*3

Table 2: The table above shows the various parameters related to objects in the point cloud data.

As shown in the table above the number of objects in the frames or grid cells are calculated, the average ground truth bounding box size of the object is calculated and the 95 percentile value of the ground truth bounding box is calculated. Using this important inferences from the data the approximate filter or the reception field size is obtained by approximation.

Number of frames	**Number of training instances**	**Training (60 percent)**	**Testing (25 percent**	**Validation (15 percent)**
7481	28742	17246	7186	4310

Table 3: Details on splitting the data into training, validation and testing

The table 2 discusses about the ratio at which the data that is extracted from the grid cells and label files is supplied to the neural network architecture to perform the operations of training, validating and testing.

Conclusion:

The point cloud data from the KITTI dataset which is essentially a collection of binary files with point cloud data is first divided into grid cells by iterating through all the points in the point cloud and the points with in the three dimensional resolution are grouped as grid cells. The grid cell data obtained is then further processed to arrive at the feature vector representation of all the grid cells. The feature vector data extracted will be the main input for training the neural network. So in conclusion the goal of representing the point cloud data into a feature vector data is accomplished by understanding the data and the main idea behind the implementation.

References:

1. Frequently asked questions, http://velodynelidar.com/faq.html
2. Andreas Geiger, Philip Lenz and Raquel Urtasun 2012, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite , Conference on Computer Vision and PatternRecognition(CVPR).
3. Implementation of Vote3Deep algorithm on KITTI Object detection data https://github.com/lijiannuist/Vote3Deep_lidar
4. Martin Enelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner 2016, Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks
5. D.Z. Wang and I. Posner 2015, Voting for Voting in Online Point Cloud Object Detection, Robotics Science and Systems.
6. User manual and programming guide for Velodyne HDL-64E http://velodynelidar.com/lidar/products/manual/HDL-64E%20S3%20manual.pdf
7. User manual and programming guide for Velodyne VLP16 http://velodynelidar.com/docs/manuals/63-9243%20Rev%20B%20User%20Manual%20and%20Programming%20Guide,VLP-16.pdf