Grand Challenge: Real-Time Object Recognition from Streaming LiDAR Point Cloud Data

Sambasiva Rao Gangineni, Harshad Reddy Nalla, Saeed Fathollahzadeh and Kia Teymourian

13th ACM International Conference on Distributed and Event-Based Systems - DEBS 2019

Table of contents

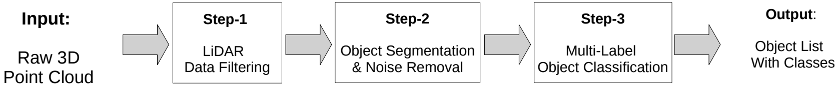
- Training data has input file and output file, input file has the coordinates and output has object names and the count of the each object.
- But there are no annotations.
- There are single-object scenes and multiple object scenes in the training data.
- Because of this problem, we cannot use the multiple-object scenes in the training phase. Also, this helped us to design our data processing pipeline.

Data Processing Pipleline

Architecture

Steps for data processing:

- **Step 1:** Data Filtering (Training and Testing)
- **Step 2:** Object Segmentation (Testing)
- **Step 3:** Object Classification (Training and Testing)



Step 1: LiDAR Laser Line Data Filtering

- Filter out the LiDAR laser lines that build a cylinder 3D shape from the laser standing point $(x = 0, y = 0, z = 0)$.
- Figure 1 visualizes the LiDAR data for a single scene with LiDAR laser lines and Figure 2 visualizes the data after filtering out the Laser lines.

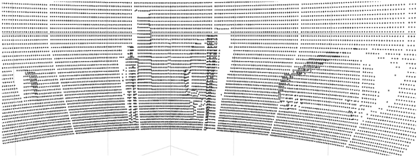


Figure 1: LiDAR Raw Point Cloud Data

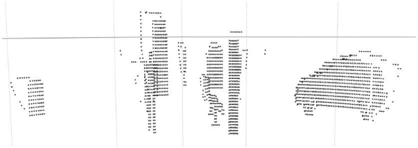


Figure 2: Data After Filtering the LiDAR Scan Lines

Understanding the 3D cylinder

- In the given data each point is annotated with the laser number.
- LiDAR used for collecting this data is mounted with the 64 lasers, each with different angle of elevation. Each cylinder line is formed by a single laser.
- In an empty scene and flat ground, the distance of the points in each cylinder line from the LiDAR is always constant.
- Thus, all the boundary points for each laser will always correspond to same distance given that the vehicle used to mount LiDAR is same.

segment the point cloud to chunks of data

- **3D to 2D Projection:** projected the 3D data in 4 different ways to a 2D plane and reduced the data dimensionality
- **Perspective projection**:

$$d = \text{Distance to a projection plane}$$

$$x' = x(\frac{d}{z}) \quad , \quad y' = y(\frac{d}{z}) \quad , \quad z' = z(\frac{d}{z}) = d$$

- Object points have varying density when the surface of the object is not notmal to the LiDAR. To make the object points dense 2D projections are used.

Distance based vs Density based Clustering

- **Object segmentation using Clustering:** different clustering methods to cluster the data
  1. **K-means and Mini Batch K-means** on the 3D and project 2D data.
  2. **Meanshift** on 3D and 2D data
  3. **DBSCAN** on 3D and 2D
- Figure 3 visualizes the data after filtering the LiDAR lines and Figure 4 visualizes the objects after clustering

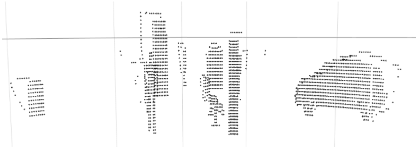


Figure 3: Data after Filtering the LiDAR Scan Lines

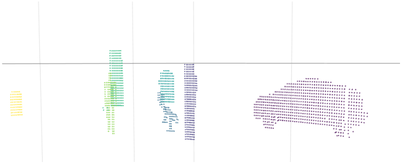


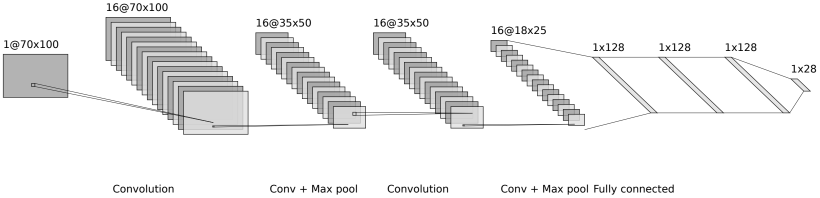
Figure 4: Clustered Point Cloud Data

Step 3: Multi-class Object Classification

Used for classification of point cloud data Convolutional Neural Network (CNN)

Layers:

- Convolutional layer
- Max Pooling layer
- Dropout Layer
- Fully Connected Layer



How to achieve real-time stream processing?

- **Step 1:** Data Filtering
- **Step 2:** Object Segmentation
- **Step 3:** Object Classification

Fast algorithm and efficient implementation.

- Choose appropiate alogirthm
- Be caution about implementation details, e.g. which PL? (C++)

Evaluation

Experiment Setups

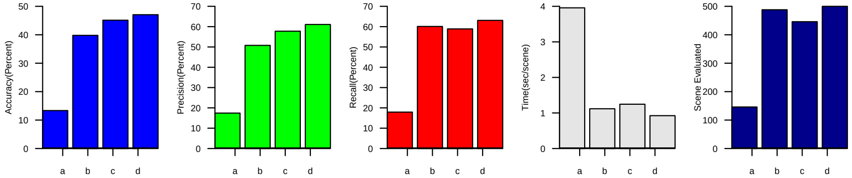
We evaluated our implementation [1] using the 4 different experiment setups:

- 2-Layer CNN on projected data to 2D (Single View) and Object Segmentation with 3D DBSCAN
- 2-Layer CNN on projected data to 2D (Using perspective projection) and Object Segmentation with 3D DBSCAN
- 4-Layer CNN on projected data to 2D (Single View) and Object Segmentation with 3D DBSCAN
- 4-Layer CNN on projected data to 2D (Using perspective projection) and Object Segmentation with 3D DBSCAN

[1] Github Repository of our Implementation `https://github.com/kiat/debs2019`

Evaluation: Experiment Settings on DEBS2019

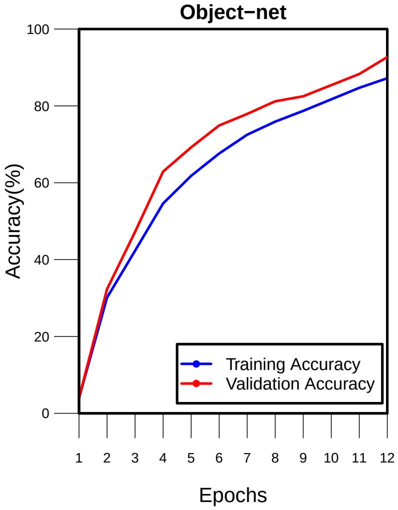
Precision, Recall, Accuracy and Processing Time of 4 different our Experiment Variation

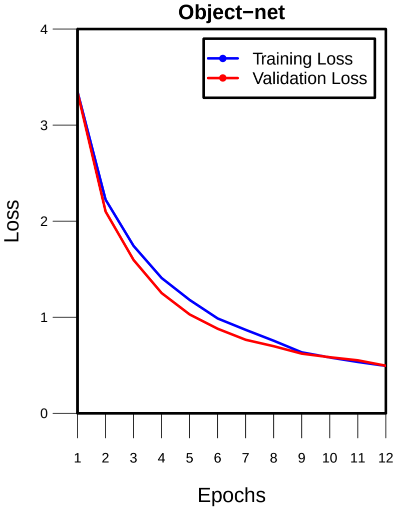


- a= 2−Layer CNN on projected data to 2D (Single View) and Object Segmentation with 3D DBSCAN
- b= 2−Layer CNN on projected data to 2D (Using perspective projection) and Object Segmentation with 3D DBSCAN
- c= 4−Layer CNN on projected data to 2D (Single View) and Object Segmentation with 3D DBSCAN
- d= 4−Layer CNN on projected data to 2D (Using perspective projection) and Object Segmentation with 3D DBSCAN

Evaluation: Accuracy and Loss

Training and Validation Accuracy and Loss





Related Work

Related Work

In this brief section, we review some of the most related publications regarding LiDAR point cloud object recognition problem.

- [Yavartanoo et al., 2018] introduces multi-view stereographic projection; it first transforms a 3D input volume into a 2D planar image using stereographic projection.
- [Zhou and Tuzel, 2018] is the best-ranked model on KITTI [Geiger et al., 2012] for 3D and birds-eye view detections using LiDAR data only
- [Wu et al., 2018] present SqueezeSeg which projects point cloud to the front view with cells gridded by LiDAR rotation
- [Riegler et al., 2017] design more efficient 3D CNN or neural network architectures that exploit sparsity in the point cloud
- [Huang and You, 2016] take a point cloud and parse it through a dense voxel grid, generating a set of occupancy voxels which are used as input to a 3D CNN to produce one label per voxel
- [Maturana and Scherer, 2015] used deep learning models is to first convert raw point cloud data into a volumetric representation, namely a 3D grid

Conclusion

Conclusion

Lesson learned from our implementation are:

- ▷ Classification of LiDAR point cloud can achieve high accuracy and real-time processing time by projecting the 3D data into 2D view.
- ▷ Classification using CNN on point cloud does not need a large number of hidden layers to achieve high accuracy.
- ▷ CNN may fail to classify if the scene includes tiny objects or objects have variable density like "Tree Objects".
- ▷ If multiple objects are in a scene and they are hiding each other (completely or partially) then object segmentation using DBSCAN or other traditional clustering methods may fail to separate objects.

Thank you!

References

Geiger, A., Lenz, P., and Urtasun, R. (2012).
Are we ready for autonomous driving? the kitti vision benchmark suite.
In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE.

Huang, J. and You, S. (2016).
Point cloud labeling using 3d convolutional neural network.
In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2670–2675. IEEE.

Maturana, D. and Scherer, S. (2015).
Voxnet: A 3d convolutional neural network for real-time object recognition.
In *IROS*, pages 922–928. IEEE.

Riegler, G., Ulusoy, A. O., and Geiger, A. (2017).
Octnet: Learning deep 3d representations at high resolutions.
In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6620–6629. IEEE Computer Society.

Wu, B., Wan, A., Yue, X., and Keutzer, K. (2018).
Squeezeseg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud.
In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1887–1893. IEEE.

Yavartanoo, M., Kim, E., and Lee, K. M. (2018).
Spnet: Deep 3d object classification and retrieval using stereographic projection.
CoRR, abs/1811.01571.

Zhou, Y. and Tuzel, O. (2018).
Voxelnet: End-to-end learning for point cloud based 3d object detection.
In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*