Final Project Report
LiDAR based 3D Object Detection
EECE 5554: Robot Sensing and Navigation

Authors: *Christopher Anton Dominic, German Gonzalez, Rishabh Singh, Srinivas Peri, Dhavan Sanghvi*

Instructor: *Dr. Thomas R. Consi*
Department: Electrical & Computer Engineering

Contents

1 Abstract

Object detection is a key component in advanced driver assistance systems (ADAS), which allow cars to detect driving lanes and pedestrians to improve road safety. This report describes a modern approach for 3D Object Detection using LiDAR while driving on the road. The system includes a Velodyne VLP-16 LiDAR sensor to capture real-time scenarios. Using Open3d, we perform the following: segmentation, RANSAC, DBSCAN, Voxel-Grid Downsampling, clustering, and detection using bounding boxes. The experimental results confirm the detection of objects such as pedestrians, cyclists, and other vehicles.

2 Project Description

The project's main goal is to investigate real-time object detection and tracking of pedestrians or bicyclists using a Velodyne LiDAR Sensor. Various point-cloud-based algorithms will be implemented using the Open3d python package. The resulting 3D point cloud can then be processed to detect objects in the surrounding environment. The reliance on LiDAR data comes at the expense of variation in point density across distances among popular datasets.

3 Explanation of sensing principle & sensor performance

3.1 Method

LiDAR is an acronym for Light Detection And Ranging. Eye-safe laser beams are used to create a 3D representation of the environment. The LiDAR sensor emits pulsed light waves from a laser into the surrounding area. These pulses bounce off surrounding objects and return to the sensor. The sensor uses the time it took for each pulse to return to the sensor to calculate the distance it traveled. Repeating this process multiple times per second creates a real-time 3D point cloud map of the environment.

3.2 Conventional versus Modern Approach

Cameras can produce 2D images of the environment while LiDAR visualizes the environment in 3D, a huge advantage when accuracy and precision are paramount. Laser-based technology produces real-time, high-resolution 3D point clouds of the surroundings, with accuracy that is unmatched by a camera. For this reason, autonomous or highly automated systems benefit from LiDAR for safe navigation.

3.3 LiDAR Data Attributes

LiDAR data attributes can vary, depending upon how the data was collected and processed. All LiDAR data points will have X, Y, and Z values with most also having an intensity value representing the amount of light energy recorded by the sensor in RGB format.

3.4 Expected sensor performance

The VLP-16 has a range of 100m with a low power consumption of 8W, a light weight of 830 grams, and dual return capabilities. Velodyne's LiDAR Puck supports 16 channels, 300,000 points/sec, a 360° horizontal field of view, and a 30° vertical field of view with +/- 15° up and down. The Velodyne LiDAR Puck does not have visible rotating parts, making it resilient in challenging environments (Rated IP67).

4 Collection of Dataset

The dataset was made available here. The dataset was collected using Velodyne's VLP-16 LiDAR model. Each packet contains the data from 24 firing sequences in 12 data blocks where each data block contains information from two firing sequences of 16 lasers.

5 Principle behind Algorithms

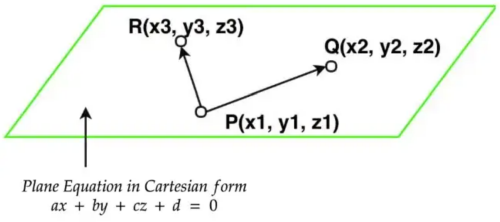
5.1 Voxel-Grid Down-sampling

After extraction, we obtain a large number of points to process which we wish to reduce for computational efficiency. However, we also wish to retain the data's key features. Because of this, we use Voxel grids which are 3D boxes or cells that can hold multiple points. The points present in each voxel (i.e., 3D box), are approximated with their centroid. This represents an accurate structure of the surface by points that are equidistant from each other. While retaining the essential features of the data set, we reduced the number of points seen in each frame to around 6,500 from the original 23,000 data points.

5.2 RANSAC: RANdom Sampling and Consensus

The core RANSAC algorithm is fairly straightforward:

- Select a random set of points (3 points to form a plane).
- Calculate the parameters required for the plane equation as seen in Figure 1.
- Calculate the distance of all the points in the point cloud from the plane.
- If the distance is within the THRESHOLD then add the point as an inlier.
- Store the plane points and points having the maximum number of inliers.
- Repeat the process again for MAX-ITERATIONS.



Plane Equation in Cartesian form

$$ax + by + cz + d = 0$$

If 3 points are provided, then the constants a, b, c, and d can be derived using the following formulas:

$$a = [(y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_2)]$$

$$b = [(z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_2)]$$

$$c = [(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_2)]$$

$$d = -(ax + by + cz)$$

Figure 1: RANSAC equations

After RANSAC is complete, the plane having the maximum number of inliers is the best estimate of the ground plane (i.e road plane). Using this model-based algorithm, the ground plane can be eliminated, hence the obstacles in the field of view of the sensor can be more efficiently localized and detected.

5.3 DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. The algorithm can be summarized as follows:

- Pick a point in the dataset (until all points have been visited).
- If there are at least MINPOINTS points within a radius of EPS to the point selected, then we consider all these points to be part of the same cluster.
- The clusters are then expanded by repeating this calculation for each neighboring point.

DBSCAN is a robust mechanism for clustering by removing noise as outliers. Its advantages over other methods include being able to detect arbitrary clusters and being independent of the requirement to predefine the number of clusters like k-means clustering.

5.4 Bounding box Detection

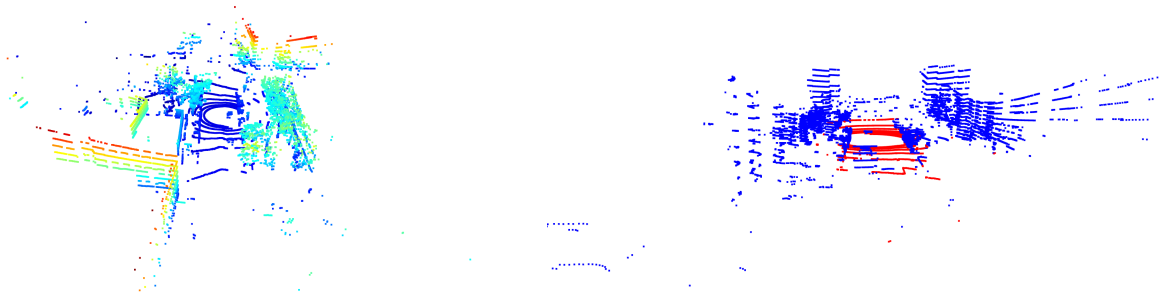
To draw bounding boxes around the clusters of points, we get the labels of each cluster and group them together. Using these labels, we get the indices of the actual points to get the axis-aligned bounding boxes.

6 Analysis of approach

We can summarize our approach in Figure 2.

Figure 2: Method used.

To begin our approach, we can first visualize the data in point clouds. By extracting our data with Velodyne-decoder and using Open3d for point cloud processing, we can see the resulting point clouds of a single frame in our data in Figure 3a. To segment the road plane out of the data, we use the previously described RANSAC algorithm. As seen in Figure 3b, the road plane, labeled in red, is well-defined and can be separated from the data to reduce the complexity and possible noise.

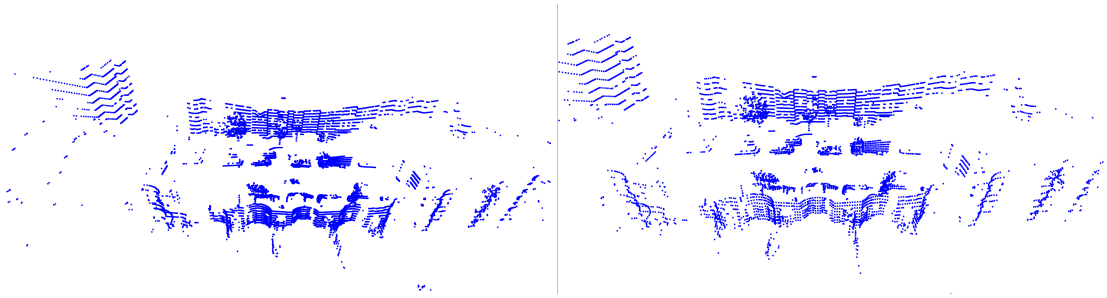


(a) Initial point clouds of a single frame.

(b) Result after segmentation.

Figure 3: Initialization & Result of Segmentation.

With only the outlier points, we still have around 23,000 points in a frame, therefore, using Voxel downsampling helps us reduce this complexity to around 6,500 points. We can see a before and after in Figure 4. Despite reducing the complexity by a factor of 3, the contents seen in the point clouds are not affected as clusters of points are not affected, rather the number of points that form those clusters.



(a) Before running voxel downsampling.

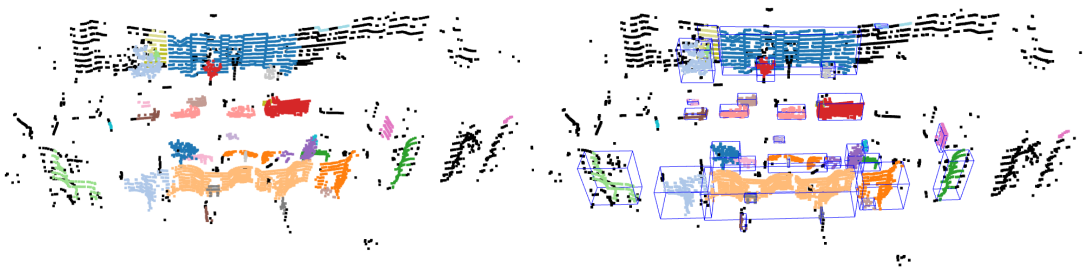
(b) After running voxel downsampling.

Figure 4: Effect of using voxel downsampling.

Finally, using DBSCAN, we can cluster the points into objects. We can see the final objects detected in Figure 5a. Given these clusters, we can place bounding boxes on the detected objects as shown in 5b.

7 Final Results

As we can see in Figure 5b using the recorded data, we were able to detect the objects accurately in real time. As seen from the final results, it may not always be perfectly accurate *because of the down-sampling we did with voxels, which may result in data loss due to saturation of point density.* <- **Not entirely accurate** But the current results seen show that our task of detection can be achieved with satisfying results.



(a) Results after clustering. **(b)** Clusters with bounding boxes.

Figure 5: Final Results.

8 Challenges

- The initial challenge was with data extraction. Extracting the files from the ROS bag file was expensive and time-consuming. Therefore, we used the Velodyne decoder, which converted velodyne packets into readable point cloud arrays directly from the bag file rather than using .csv files.
- The second challenge dealt with detecting objects across all time instances. When drawing bounding boxes, we can only do so for a single frame. We could not get bounding boxes to show on a sequence of images even with loops. Nevertheless, we can still detect clusters.
- There were numerous issues with the back-end software as different parts of code were executed on different laptops, which were resolved by updating them to the required versions.
- We began trying to use Open3d-ML for deep learning-based methods, but we could not get the build to work on Windows and we could not get GPU access through the virtual box.

9 Future Scope

From the images in Figure 5, it is clear that we can detect and set bounding boxes to objects seen by the LiDAR sensor. Now, the project's future focus will be on classifying the data and running machine learning algorithms to make decisions autonomously. This can be accomplished by utilizing Open3D-ML within Open3D. Additionally, because we are performing these algorithms independently for each individual frame, having an algorithm that can take multiple frames at once and run the object detection process on this sequence of images can result in more coherent results. Therefore, we believe using Deep Learning methods such as Point Pillars and Complex Yolov4 can result in improved performances.

References

[1] Velodyne lidar puck. https://www.amtechs.co.jp/product/VLP-16-Puck.pdf.

[2] Downsampling a pointcloud using a voxelgrid filter. https://adioshun.gitbooks.io/pcl/content/Tutorial/Filtering/pcl-cpp-downsampling-a-pointcloud-using-a-voxelgrid-filter.html, 2022.

[3] Downsampling a pointcloud using a voxelgrid filter — point cloud library 0.0 documentation. https://pcl.readthedocs.io/en/latest/voxel_grid.html, 2022.

[4] What is lidar? learn how lidar works, velodyne lidar. https://velodynelidar.com/what-is-lidar/, 2022.

[5] Nagesh Chauhan. Dbscan clustering algorithm in machine learning - kdnuggets. https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html, 2022.

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

[7] Martin Valgur. Velodyne decoder. https://github.com/valgur/velodyne_decoder, 2022.

[8] Leah A. Wasser. The basics of lidar - light detection and ranging - remote sensing. https://www.neonscience.org/resources/learning-hub/tutorials/lidar-basics, 2022.

[9] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.