Using Faster-RCNN to Improve Shape Detection in LIDAR

TJ Melanson

Stanford University

INTRODUCTION

In traditional structure from motion mapping, a camera uses a series of images to calculate disparity and depth to directly compute a point cloud for scene reconstruction. However, in cases such as autonomous vehicles, the 3D reconstruction alone is not enough to gather the necessary scene information. In the case of autonomous vehicles, vision is needed to distinguish obstacles as well as recognize intent. My project, is about using object recognition to augment reconstruction by replacing partially reconstructed meshes of objects with full models of a generic model type recognized by CNNs. The results are evaluated visually, by comparison with the original scene, as well as numerically, by comparing depth, size, and object type to real collected data.

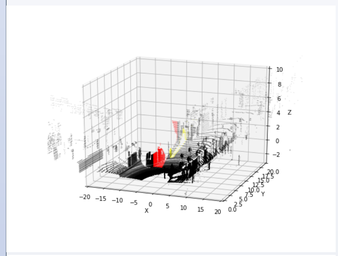
OBJECTIVES

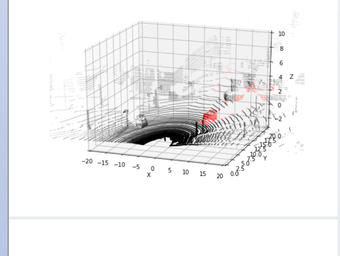
The purpose of this project is to detect objects in 3D space by properly segmenting the portions of the Point Cloud related to three classes: Person, Car, and Cyclist. Although other classes, such as Plant and Train/Tram, are included in the detection suite, there is not a significant enough set of objects outside of the three major class types to warrant a full training/testing dataset.

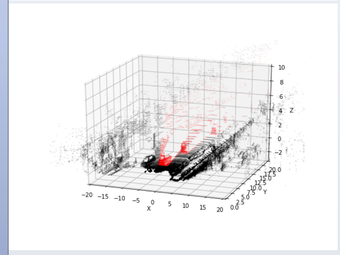
The purpose of using a neural network is to detect the shapes of the vehicle within the 3-D image so it can be processed or segmented out. For this ,the goal is to compare the effectiveness of using camera data to detect the sources of foreground data versus a method that only used volume-based neural networks such as V-CNN.

Once these foreground obstacles are detected, they then could be tracked by the vehicle over time, and can then be either recorded or acted upon by a vehicle decision.

RESULTS







METHODS

For the initial run, Faster-RCNN trained on default VGGNet was used to segment and detect the objects within the camera image. Afterwards, the velodyne data was projected onto the camera via its transformation and projection function, and the data within the object region was labeled accordingly.

KITTI has a number of benchmarks for both tracking and object detection. In the 2D object detection case, KITTI finds the average precision of the dataset, based on the PASCAL defined metric. Essentially, this involves averaging the percentage of correctly detected features over the total possible detectable features.

For 3D shape detection, PointNet was used, which is a series of augmented fully connected networks (with feature transformation as well as max pooling for feature aggregation)

CONCLUSIONS/FUTURE WORK

The default configuration of Faster-RCNN (trained on VGGNet) proved sufficient to detect the majority of vehicles within a region.

The PointNet structure was unable to determine the location of the objects given the input data. Given the lack of provable evidence of a vehicle or person shape within a single LIDAR scan, the best next step would be to combine several LIDAR scans / sets of image data into a single, continuous stream. This would change the frame of the objective from one of object detection to that of object tracking. While an initial attempt could be to use a LSTM or similar state-driven approach, there are multiple sources on KITTI for current tracking methods and benchmarking sources within a tracking section of the KITTI dataset.

Additionally, the shapes could be run individually into PointNet, although there don't seem to be enough recognizable feature s on the point cloud to recognize objects within it.

REFERENCES

1. Kundu et al. Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. Georgia Institute of Technology, 2014. https://web.engr.oregonstate.edu/~lif/HybridSFM-ECCV2014.pdf
2. http://www.robots.ox.ac.uk/~mobile/Papers/2016IROS_maddern.pdf
3. http://graphics.usc.edu/cgit/publications/papers/point_cloud_3dcnn.pdf
4. http://www.roboticsproceedings.org/rss12/p42.pdf
5. http://hci.stanford.edu/cstr/reports/2012-01.pdf
6. http://ai.stanford.edu/~haosu/papers/iccv_renderforcnn.pdf
7. https://arxiv.org/pdf/1702.04405.pdf
8. https://arxiv.org/abs/1612.00593
9. https://arxiv.org/abs/1506.01497
10. PyKitti http://www.cvlibs.net/datasets/kitti/eval_object.php
11. http://www.cvlibs.net/publications/Geiger2012CVPR.pdf

ACKNOWLEDGEMENTS

I would like to thank Ross Girschik, Charles Qi and the others in creating py-faster-rcnn and pointnet for providing the baseline for the project, as well as my mentor, Leo, as well as the CS231N Course Staff