Pedestrian Detection with LIDAR
Midterm Progress Report

Elbert Lin
el168@stanford.edu

Greg Katz
gkatz@stanford.edu

1. Introduction

Jack Rabbot is a robot equipped with many sensors that are used to navigate crowded areas whilst maintaining a human-like sense of social etiquette. One of these sensors is lidar, which provides a 2D scan of objects in an area in front of the robot. For our project, we are taking this data (and only this data), and attempting to detect which of those objects are in fact pedestrians. We use different methods of segmentation and feed those data segments into appropriate neural networks to train for detection. This would help Jack Rabbot avoid collisions with people, and aid in recognizing situations in which it must behave differently. The results can also be used to augment data and inferences made using other sensors. Given time, we could extend our project to work in the temporal dimension, in 3D, or both.

2. Problem Statement

For this project we train and test on data recorded on Jack Rabbot. Ground truth pedestrian bounding boxes are provided by the results of an optical pedestrian detection algorithm. Thus we have a series of labeled real world 2D lidar scans to train and evaluate our algorithms, and our expected results would be estimated bounding boxes.

In this project, failing to detect a pedestrian is worse than a false positive, so we want to set our detection threshold to ensure at least 95% recall. Our evaluation criteria will then be the false positive rate. We could also provide images of the bounding boxes we obtain, to visually check that the boxes are reasonably assigned.

3. Technical Approach

We follow an incremental approach to ensure a working baseline before moving on to more complicated methods. Our first task will be to detect pedestrians from a single snapshot. We propose two approaches for this task. The first approach is to use a sliding window to find regions of interest based on human-sized point clusters and then pass these clusters into a basic neural network which will output a probability that the cluster is a pedestrian, similar to [3]. The second approach is to modify the YOLO neural network [4] architecture to take in lidar scan data. This architecture is designed to find bounding boxes and perform classification in a single network.

Our second task is to detect pedestrians from a sequence of lidar scans. By expanding the input data to include temporal information accuracy should be greatly improved based on anecdotal evidence that it is easier to identify the lidar signature of pedestrians when they are moving. Including temporal information increases the data processing challenge because now a region of interest must be tracked in space. One approach is to package sequences of several lidar scans and pass this higher dimensional data into the neural network. Another would be to employ a recurrent neural network architecture.

A final task would be to extend this work to 3D lidar scans, for example with the KITTI dataset [2]. This could be done by separating the scan into 2D planes, processing them separately, and averaging the results from each layer. Alternatively, the approaches developed for the 2D problem could be expanded to the third dimension. This would mean finding 3D point clusters and expanding the input dimension of the neural networks.

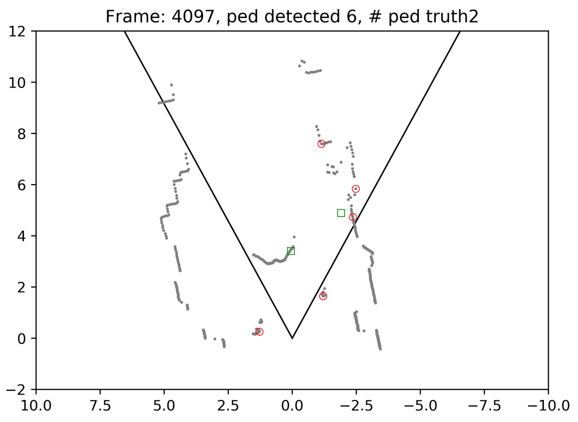
4. Preliminary Results

Figure 1 shows a sample of our initial results for a single lidar measurement instance. This image shows several examples of the challenges with pedestrian detection.

- The green square in the center is a pedestrian visible in 3-dimensions but occluded in the plane of the LIDAR
- The highest red circle is a correct pedestrian detection but a wrong localization. Additionally YOLO did not pick up on this pedestrian in this frame.
- The two red circle farthest to the right are correct pedestrian detections, but they are duplicate.
- The farthest right green square shows a mismatch between the depth of the YOLO pedestrian detection and the LIDAR range data.

- the two lower red circles show successful pedestrian detections outside of the field of view of the visual camera.

Figure 1. Red circles are pedestrians predicted from LIDAR. Green squares are pedestrian estimates from YOLO.



4.1. Segmentation and Training

For our preliminary results we have implemented a simple neural network that we are using with a sliding window approach. Our initial network architecture has two hidden layers of 300 nodes each with ReLu activation and a single output node with a sigmoid activation. Our loss function is binary cross-entropy. For the preliminary results we are using the simplest possible segmentation strategy in which we simply specify a window size and a stride to slide across our data set. Each segment becomes a feature vector and we assign a label of 0 or 1 based on whether our ground truth data set from YOLO indicates a pedestrian anywhere in this segment. Since we consider a pedestrian anywhere in the segment a positive example, this leads to a high rate of duplicate detections in the test data set.

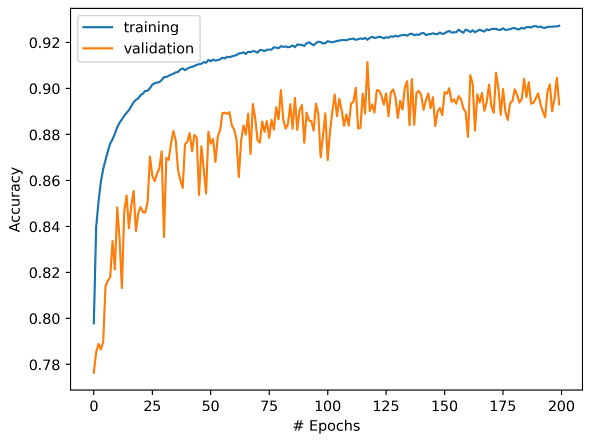
We noticed that when we segment this way, more than 90% of the segments do not contain a pedestrian. To compensate for this bias we use oversampling [1].

Before segmentation the data is split into training, validation, and test sets. See Figure 2 for a plot of the accuracy during training. There is room for improvement in both underfitting and overfitting. For underfitting we plan add more nodes and more hidden layers to the network. For overfitting we will try adding a regularization term to the cost function and dropout layers.

4.2. Testing and evaluation

To run our model in testing, we segment the test data set and feed each segment in a forward pass through the network. For each segment we predict a pedestrian only if the output of the model is greater than some threshold. Then within that segment we select the smallest range measurement that is within 0.6 meters of the median range in the segment as the locus of the predicted pedestrian. Selecting the locus of the pedestrian within a segment is a process that could be significantly improved. No effort is made to remove duplicate pedestrians from overlapping or nearby segments, which likely contributes to a high rate of false positives.

Figure 2. Accuracy during training



To score the results of the model predictions, we associate each YOLO pedestrian estimate with the nearest pedestrian prediction from our model. If these are within 1 meter this counts as a successful prediction. Any YOLO estimates that don't have a prediction within 1 meter are counted as false negatives and any remaining predictions are counted as false positives. We then calculate the F1 score as the harmonic mean of the precision (proportion of predictions that are correct) and recall (proportion of YOLO estimates that are correctly predicted). Figure 3 shows the effect of selecting the threshold for a detection on precision, recall, and F1 score. Figure 4 shows the trade-off between precision and recall.

There is a significant drop-off in performance between the network training accuracy, 90%, and the test accuracy, 55%. There are several contributing factors to the overall error rate. First, with the sliding window method is that multiple windows will see and predict the same pedestrian. A second factor is that it is not clear where in the segment the pedestrian is located. These are two major sources of error that are causing our aggregated results to be significantly worse than the training accuracy. A third source of error is that in some cases the location of the pedestrians found by YOLO do not match any lidar range measurement.

Figure 3. Effect of threshold on precision and recall

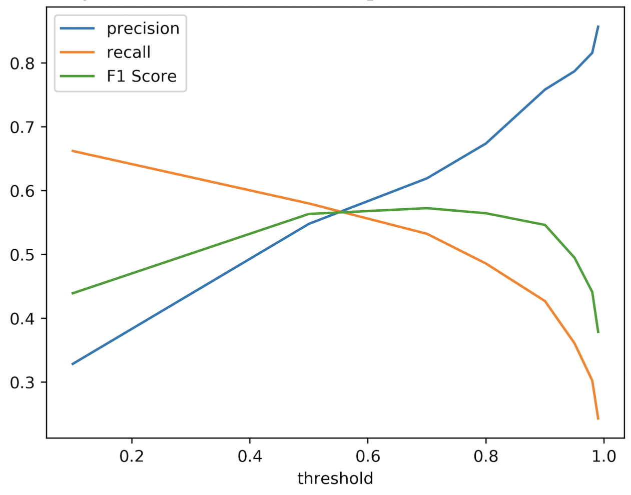
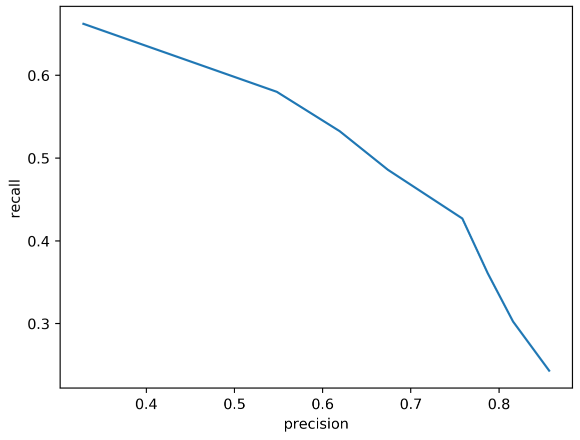


Figure 4. Precision vs Recall



5. Next Steps

Moving forward, we plan to do error analysis to see the contribution of each of these factors to the overall F1-score. Additionally, we hypothesize that an end-to-end neural network that directly predicts pedestrian locations will significantly improve the score. Finally, we plan to use post-processing techniques, for example finding straight lines for walls, and using those to reject unlikely pedestrian candidates.

References

[1] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *CoRR*, abs/1710.05381, 2017.

[2] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] D. Matti, H. K. Ekenel, and J. Thiran. Combining lidar space clustering and convolutional neural networks for pedestrian detection. *CoRR*, abs/1710.06160, 2017.

[4] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.