https://robotik.informatik.uni-wuerzburg.de/telematics/download/iv2012.pdf

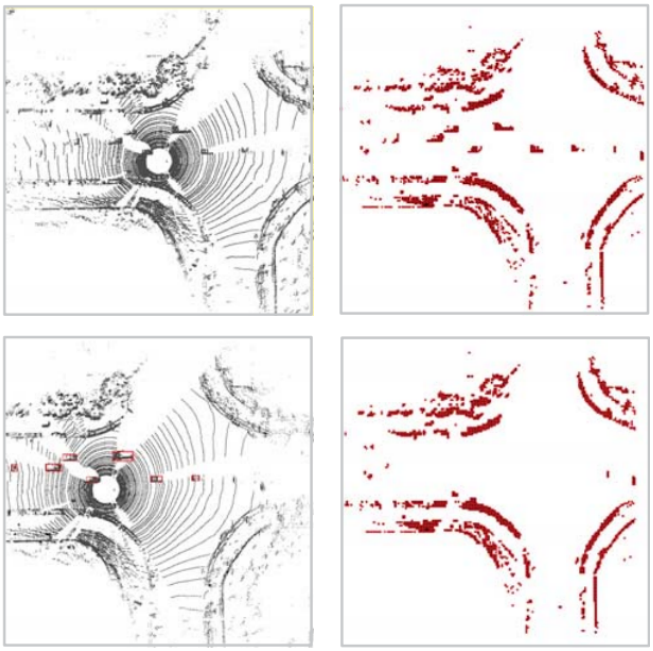
3D LIDAR Point Cloud based Intersection Recognition for Autonomous Driving

This paper proposes an alternate way to detect road intersections in advance by the use of LiDAR mapping and a beam based feature construction.

Preprocessing

Before the beam model is implemented, data needs to be preprocessed, we need a rough outline of a birds eye view of the road without obstacles like trees and pedestrians.

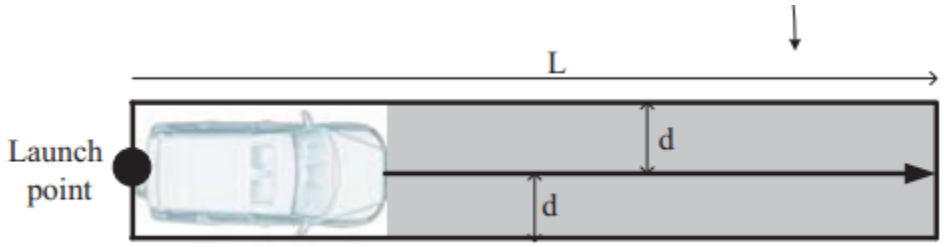
1. First, a dense 64-beam scanning LIDAR maps the surroundings in the form of 3D point clouds.Then, a square grid map for each frame of the point cloud is built. This creates a matrix of cells of size rxr.
2. The variance (σ^2) of each cell is calculated and if this value is above a given threshold value, the cell is marked as 1, otherwise it is 0. This creates a binary matrix of 1s and 0s that resembles a birds eye view of the surroundings
3. We collect the cells who's neighboring cells are 1 as a **connected region,** rebuilding this connected region as an approximate cube can tell us if this region is part of the road boundary or if its a passing vehicle or person. We can infer this from the height and length of this "cube"
4. By removing the vehicles and pedestrians from the map, we are left with a map of the road. This is what is used for detecting what kind of intersection the road has



What the data looks like without obstacles

Beam model:

It works on a similar principle to rangefinders, i.e time-of-flight principle.



The beam model is a sequence of beams with same launching point which is within the adaptive distance in front of the autonomous vehicle, the distance is related to the speed of the autonomous vehicle by the formula

D=5+v*t

Where

D= distance between the vehicle and the launching point

v=speed of the vehicle

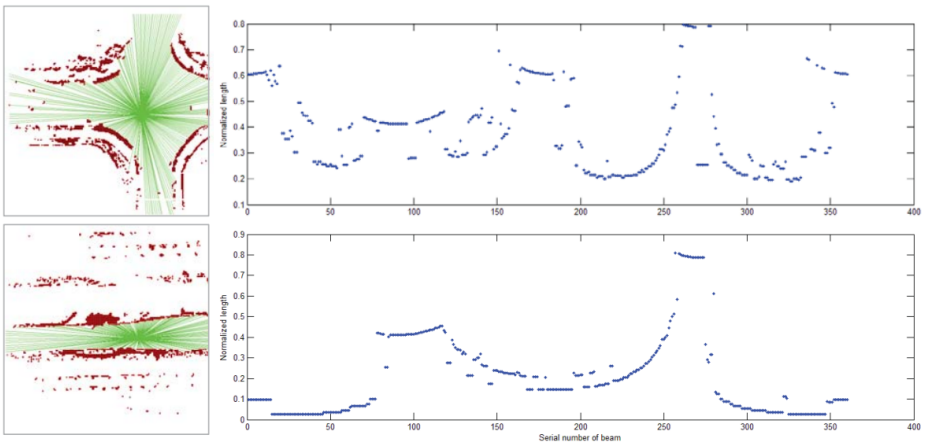
t=time(taken as 1s in the paper).

This is unlike existing technologies that use a fixed length D.

In this technology, the higher the speed is, the longer is the distance and the slower speed is, shorter the distance, which is more optimized.

When a beam from the launch point hits an obstacle, it will get cut off. If the beam doesn't hit an obstacle, its length is capped to a constant L. For intersections and road segments, the distribution of the length of each beam is different. The different lengths are plotted on a histogram.

The histogram data can be processed by a machine learning algorithm. For example, double peaks in the histogram implies a straight road while 4 peaks implies a +-shaped road, and hence, the nature of the road intersection can be determined.



Testing and results

The fourth part of the paper tests the accuracy of the algorithm by testing it on large data sets and plotting the true positive and true negative rates along with the accuracy and the receiver operating characteristic curve (ROC) and the area under curve (AUC) .
The results are as shown:

TABLE I
SVM PERFORMANCE ON INTERSECTION AND ROAD SEGMENT CLASSIFICATION

	TPR	TNR	Accuracy	AUC
Test Data 1	91.25%	96%	93.625%	0.987
Test Data 2	81%	84%	82.5%	0.938

TABLE II
SVM PERFORMANCE T-SHAPED AND $+$-SHAPED INTERSECTION CLASSIFICATION

	TPR	TNR	Accuracy
Test Data 1	93.382%	80.681%	85%
Test Data 2	85.714%	79.545%	83%

My inputs:

The proposed variable adaptive distance **D** in the algorithm seems to be a better alternative, as you will need to estimate the road ahead faster if you are speeding, the ML bot will need to move as fast the car in terms of processing and increasing the D length will in turn increase range of the detector, while optimizing the required processing power as processing too much will cost energy and efficiency.