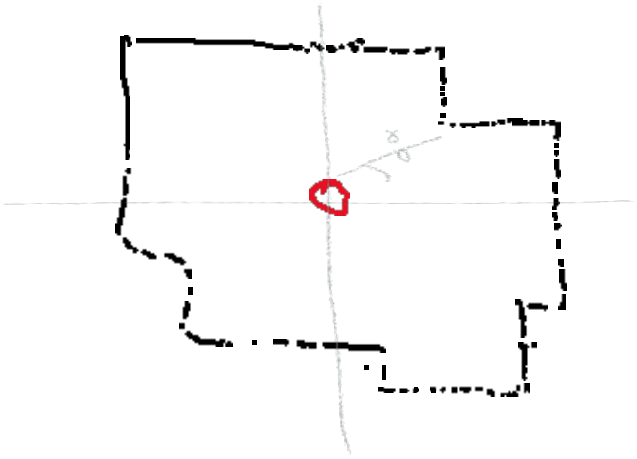
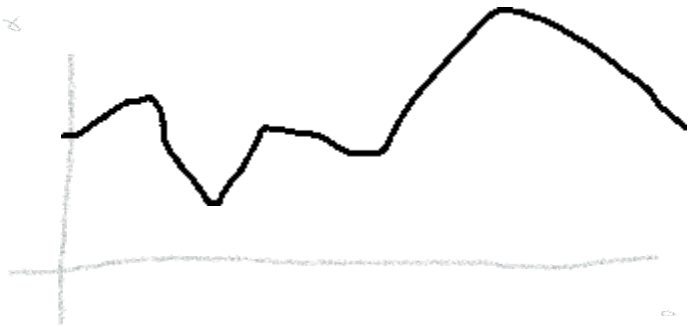
Make a brief documentation on your understanding of the above paper. The documentation can include insights, results, alternatives and any suggestions/improvements that you can think of. We essentially want you to explain the algorithm to us, discuss how well it will/won't work.

LiDAR – stands for light detection and ranging.

Suppose we have a laser pointer which tells us at what distance from us (the laser pointer) the projected point is. Although this laser pointer is imaginary we could accomplish this via having a receiving module along with the emitter and timing the time it takes the light to reach back to the receiver. Coming back to the point let's say now we turn the laser pointer by one degree and take a measurement and jot it down, then again turn it one degree and then take one measurement continuing this process lets say we take the measurement of each degree from 1 to 359 of a room/roadway and while taking these measurement we are stationary. Now suppose a polar system of coordinate and we are standing on the origin. Lets say we plot these points on the polar axis each point represented by its radius and theta. Now if we see this plot from above we would be able to see the shape of the room/roadway.



This if plotted on a (theta) vs (r)



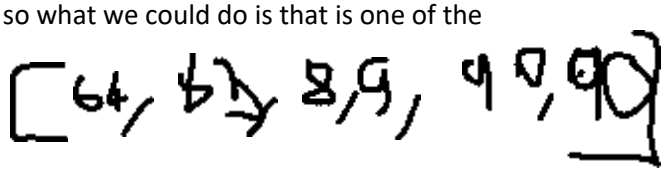
Now let's suppose our data of the road has people and cars in it
An approach could be to remove any sudden variations in the above graph.

And converting it into to reduce the suddenness of the graph.

Another approach could be using kernels but this has to be speculated.

Also according to the paper the sensor has a width of sensing so what we could do is that is one of the measurement taken has a large variation.

Like this



Then we ignore this measurement if the mean deviation or something is higher than threshold

If our LiDAR has 3d capability then we could filter by the height of objects too and in the process create a mask of the areas we are not going to input to our model.

Before inputting it would be wise to perform a feature scaling;

A simple would look like (theta-180)/180

And r/100 sort of.

Then our trained machine learning model would first learn from the training data.

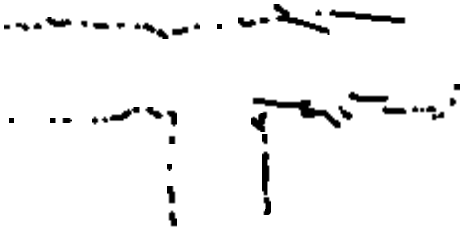
Then we may test it out on the real world.

In general it would work very well specially if the architecture style road width and other things are of the same design.

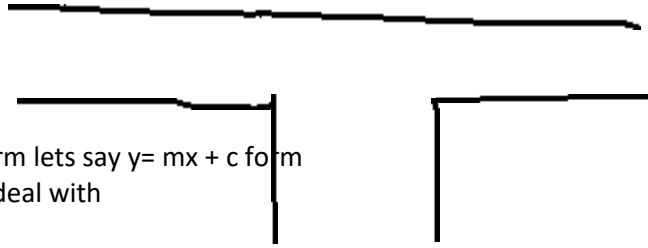
What we could do to improve it?

We could apply Hough line transform on the data and replace the points with a straight line but how we will input it in the model will have to be seen to.

What else can we do is that we could declare points to be irrelevant and only input lines in a model
So that a data like



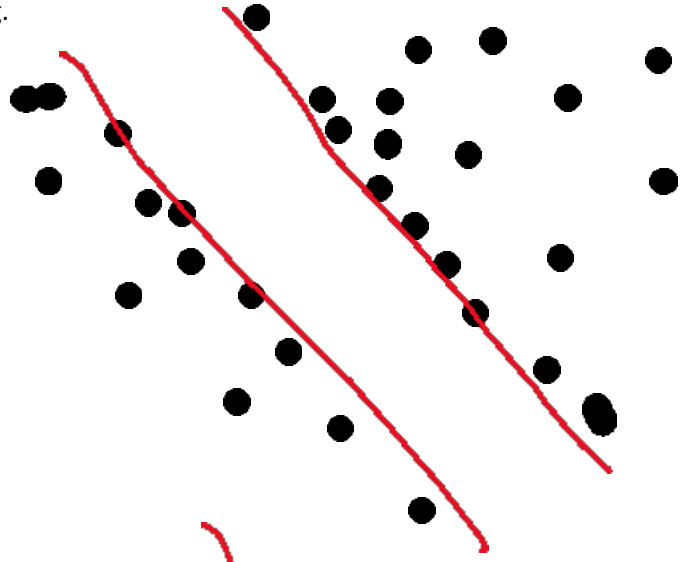
becomes



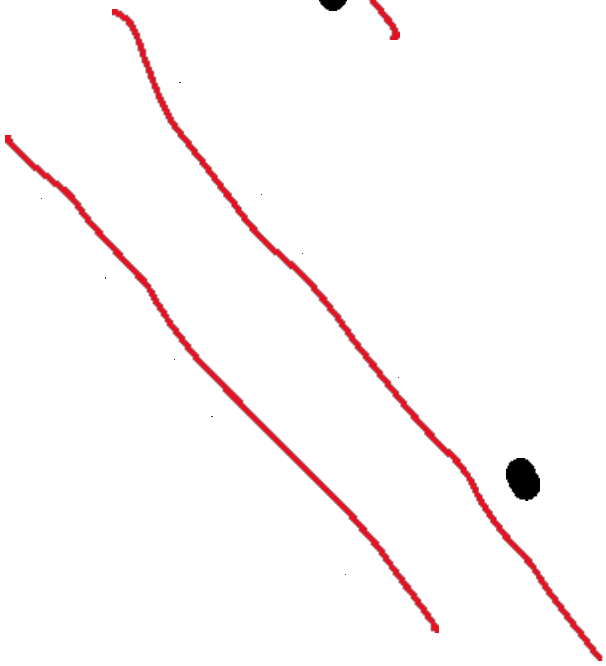
and instead of inputting lots
and lots of points we are now
giving the model 5 lines in any form lets say y= mx + c form
which is a lot less parameters to deal with

The features or each feature/degree of data has to be computed and that computation is in general intensive. As given we are using SVM type classifiers we could in some way get rid of excess data and only keep the support vectors while inputting.

Instead of



We input only



The practicality or usefulness of these improvements would have to be speculated.