ESE 650 – Path Planning – Imitation Learning

Vishnu Purushothaman Sreenivasan

Introduction

Problem Statement

The objective of the project is to use the aerial map image from Google maps of the Penn Campus and to learn a model for predicting the optimal path which will be taken by a car and a pedestrian by using imitation learning (a type of reinforcement learning). So in the project a parameterized cost map is generated using an exponentiated sum of some weighted features. Then using expert examples of optimal policies and employing the Learch Algorithm, to get the weights for the cost map that will produce behavior similar to that of the expert policy.

Description of approach

A large chunk of this project work was focused on feature engineering and choosing a good seed for the feature weights to feed into the Learch algorithm for imitation learning. Throughout this report I make references to the Learch algorithm which is based on the paper, "Learning to Search: Functional Gradient Techniques for Imitation Learning" by Ratliff et al. First the image was resized to one eight the original size and then I extracted a bunch of features for every image pixel. By looking at the features I did an educated guess of the weights to be used as a seed to the Learch algorithm which works better and faster when the initial cost map is close to the ideal one. Now, I got a better estimate of the weights using Learch, though I found in some cases the initial seed was a better estimate when the learning rate is not properly set. In every iteration in the Learch algorithm, I compute the cost map with the recently computed weights which I augment by adding a loss field function which increases the costs at the desired path by a fraction of the maximum value of the current cost map (10%). Then I used dijkstra search, to get an optimal path. Now I take all the points in the optimal path and the ones in the desired path of all the MDP's perturb them randomly upto 3 pixels along x and y. I run a binary classification between the features of the points on the perturbed optimal path and the desired path with a L2 regularized Linear Support Vector Machine using Liblinear. I update the old weights with the weight obtained from the classification after scaling with an exponentially decaying learning rate. This procedure was done for both the car and the pedestrian.

Methodology

The methodology can be split into the feature extraction and the training and testing part:

Feature Extraction

a) I first resized the given image to one eight the original size as the algorithm runs way faster on the resized image.
b) I then extracted the following features for every pixel of the image.
    a. Horizontal Edge Response with sobel.
    b. Vertical Edge Response with sobel.
    c. Corner response with Harris corner detector with a standard deviation (sigma) of 2.
    d. Laplacian of Gaussian of the image using a kernel of size of window size 18 and a sigma of 6. At this value the building and trees had a good response but usually the roads had lesser.
    e. 10 clustered colors in the LAB color space.
    f. 11 clustered colors in the HSV color space.

Features

Image Clustering in the HSV space

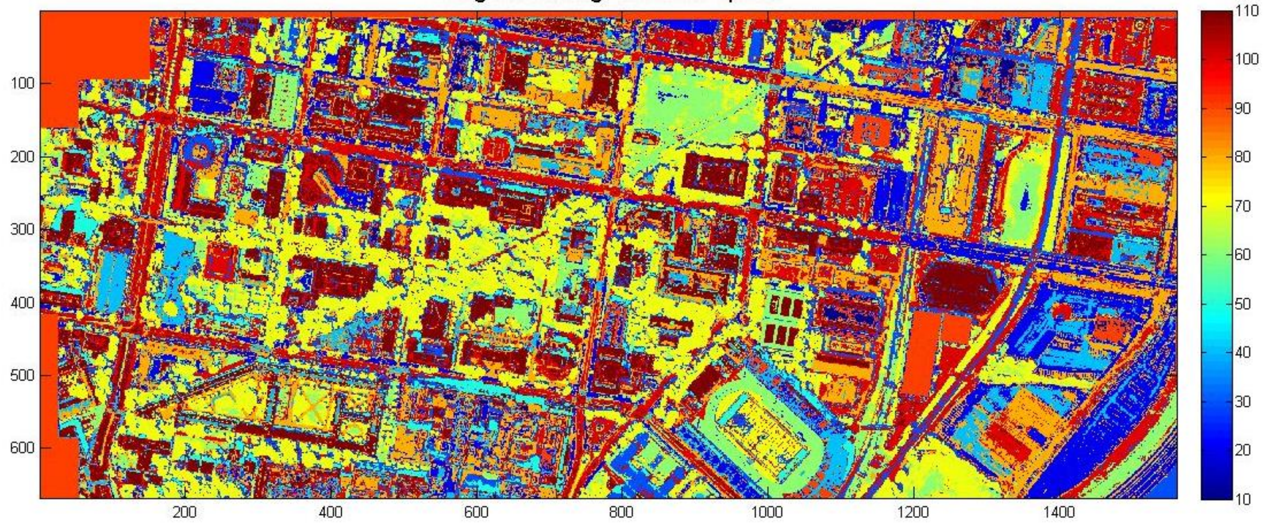
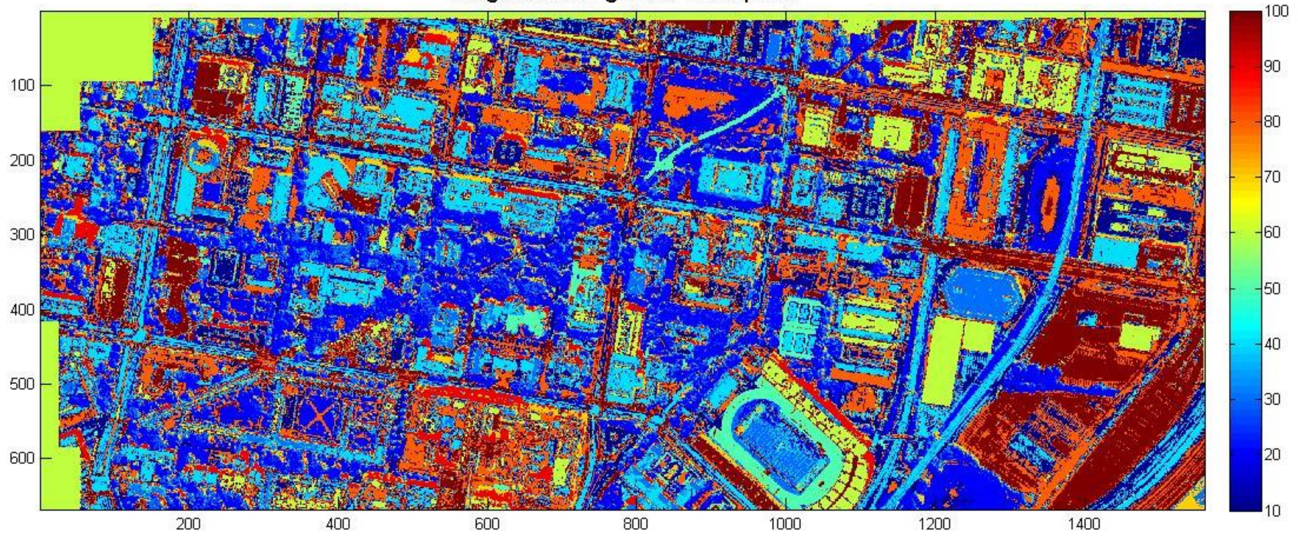
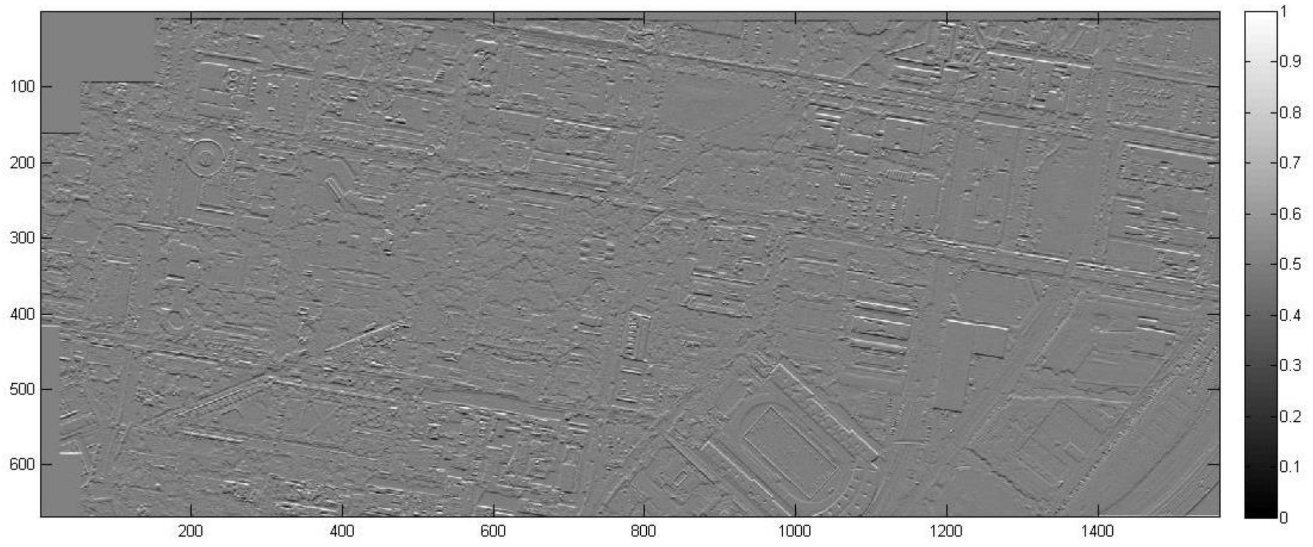


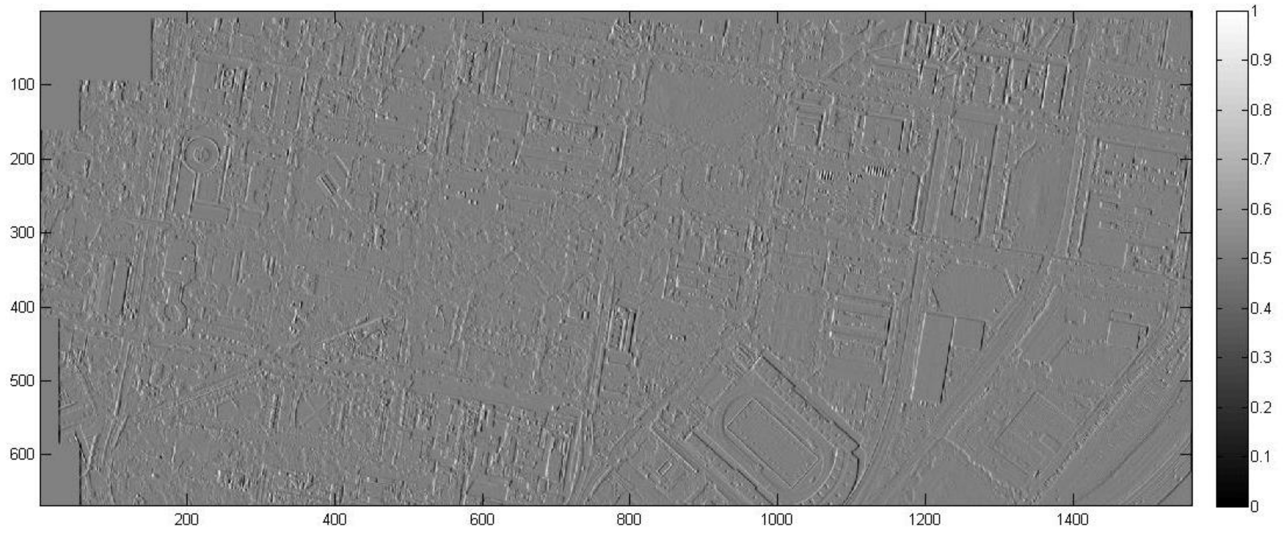
Image Clustering in the LAB space



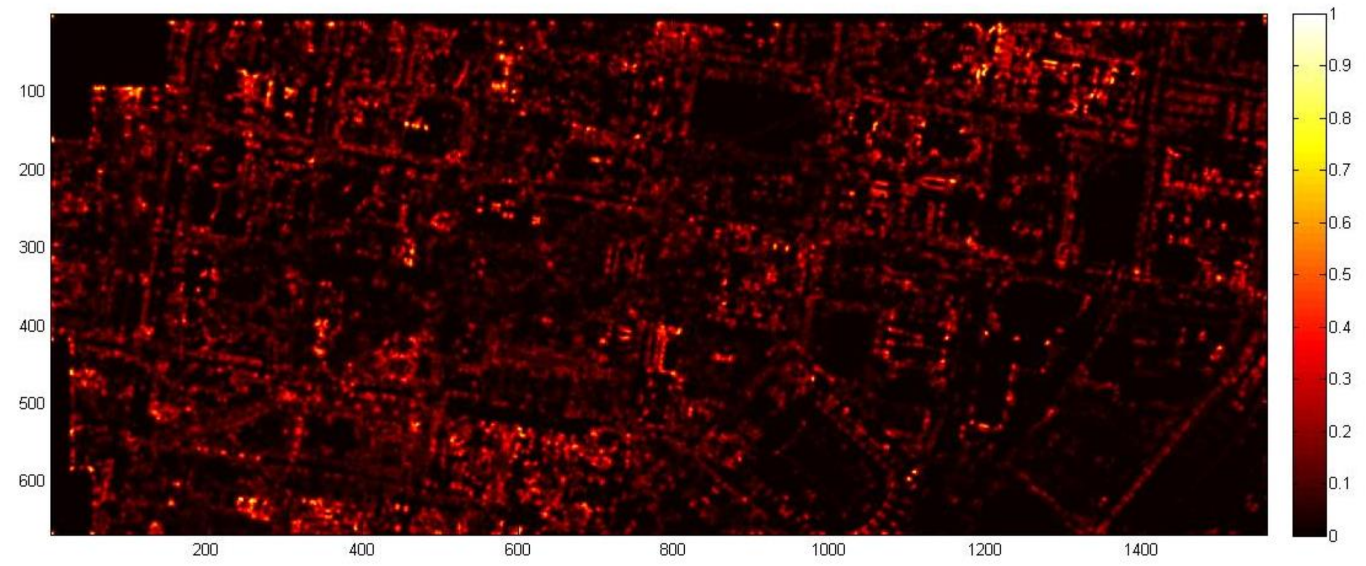
Horizontal Edge Response



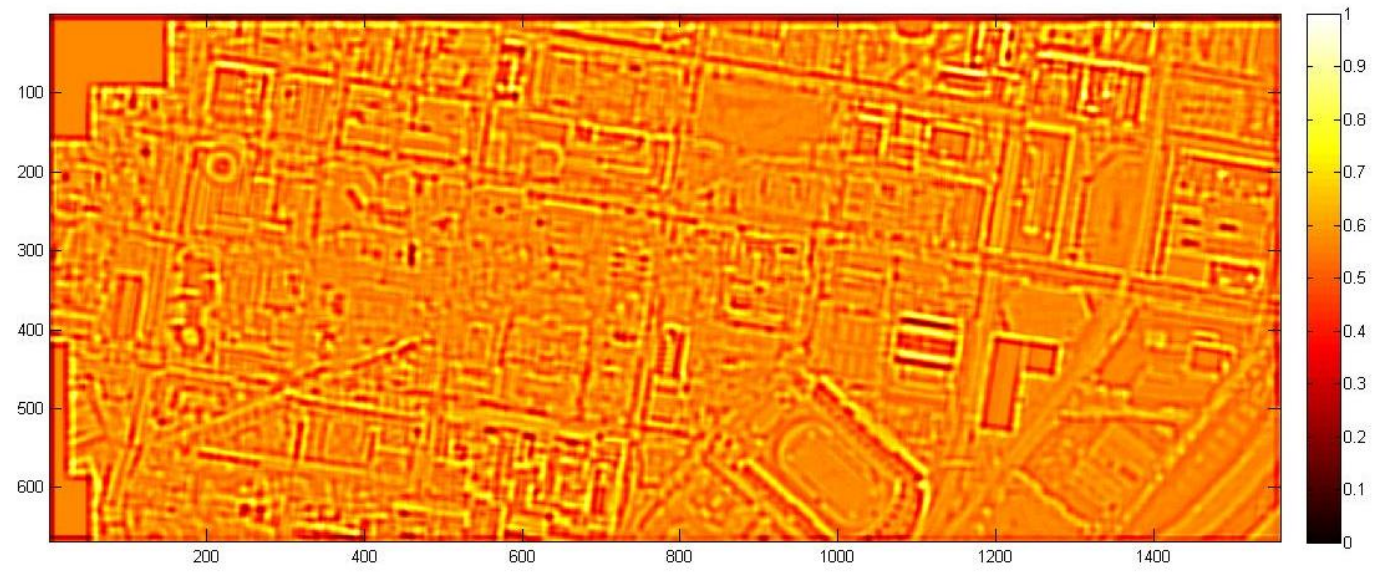
Vertical Edge Response



Corner Response



Laplacian of Gaussian Response



Training and Testing

a) I tried to estimate a good approximation of the weights for each of the above features by visual inspection, this was just to obtain a good seed for the Learch algorithm to learn a cost map faster (a random input weight and then training on one or two paths to a good seed of the weights also worked but manual initial seeding converged faster to a good optimal solution).

b) I hand drew 20 examples for car. I picked the top 10 examples for training. I tried the different example paths and trained only on the ones it got wrong. For the pedestrian I started with a good guess of the weights. But still the output was not satisfactory, so I made examples where the path generated was not close to the desired and trained the model on it.

c) Now I performed the Learch algorithm. In each iteration of the algorithm, I computed the cost map using the recently updated weights (using the seed for the first iteration). I augment this cost map by using a loss field function. The loss field function is a cost map with the same size of the original cost map. I made it 0 every but the desired path which was set to 10% of the maximum value of the current cost map. Now I smoothed the loss field map using a Gaussian filter of window size 3 and sigma value 1.4. I also made the center of the Gaussian kernel 1. This loss map was subtracted from the original cost map to produce the Loss-Augmented cost map.

d) Now I ran dijkstra search over this new cost map and found the optimal path. I took all the points in the optimal path and the ones in the desired path of all the MDP's perturb them randomly upto 3 pixels along x and y. I ran a binary classification between the features of the points on the perturbed optimal path and the desired path with a L2 regularized Linear Support Vector Machine using the Liblinear toolbox developed by NTU.

e) The learning rate η is given by the following equation. The exponential decay helps in convergence.

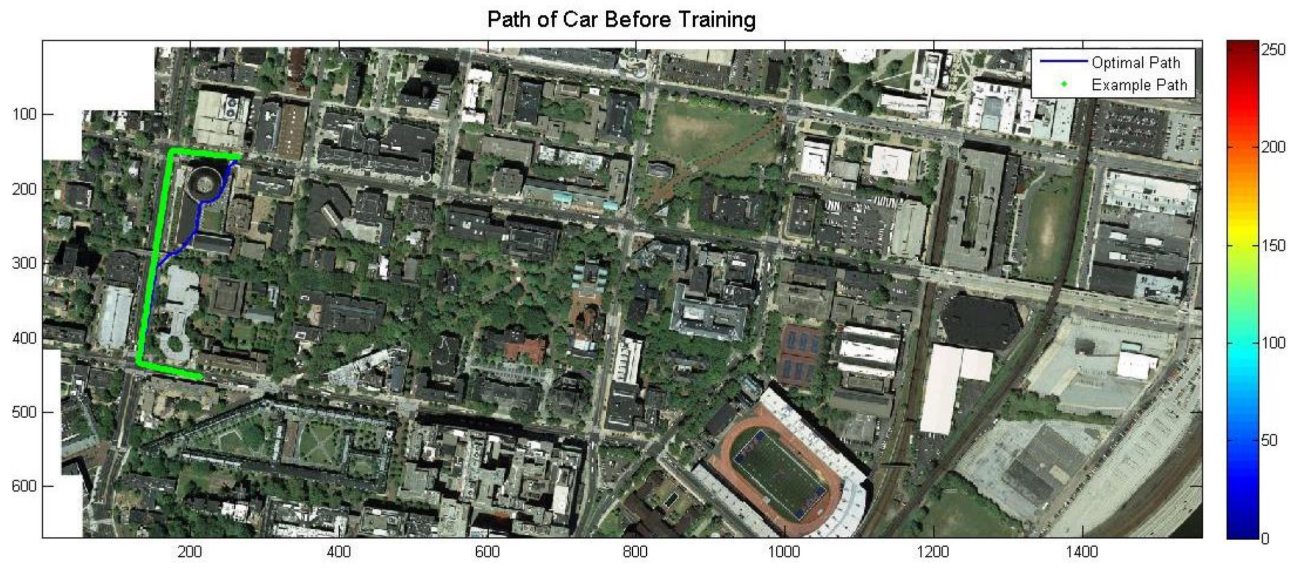
$$\eta = \frac{.5 \times 10^{-4}}{\sqrt[5]{Iteration\ Number}}$$

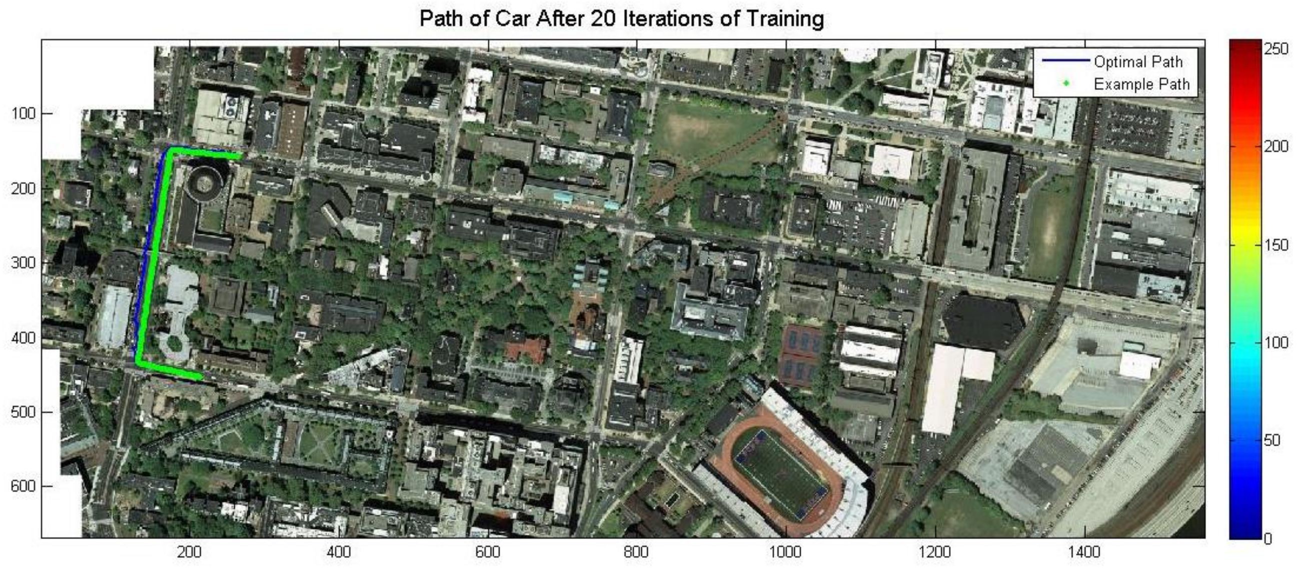
f) The algorithm is terminated when the change in weights is within a threshold or the algorithm hasn't converged after 2000 iterations.

g) The above learning method is repeated for the pedestrian and the car.

h) The testing part was the simple part where I just loaded the cost maps and then I ran dijkstra on it and found the optimal path.

Training Example





Results

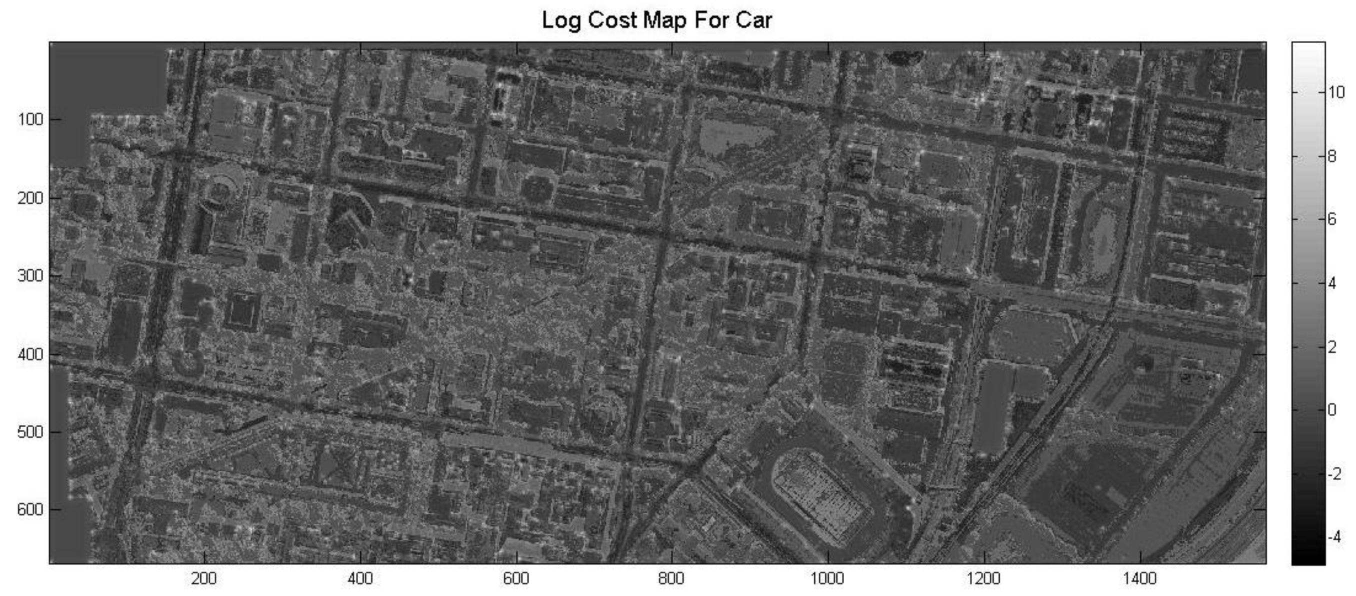
The path planning for the car work very well in most places. But it suffers non optimal behavior in the regions with the presence of shadow as it is hard to differentiate between a road and footpaths. Similar issue arises in the highway where the road has striking similarity in color to that of a footpath in other regions of the map.

But overall the path planning was very accurate and followed good behavior.

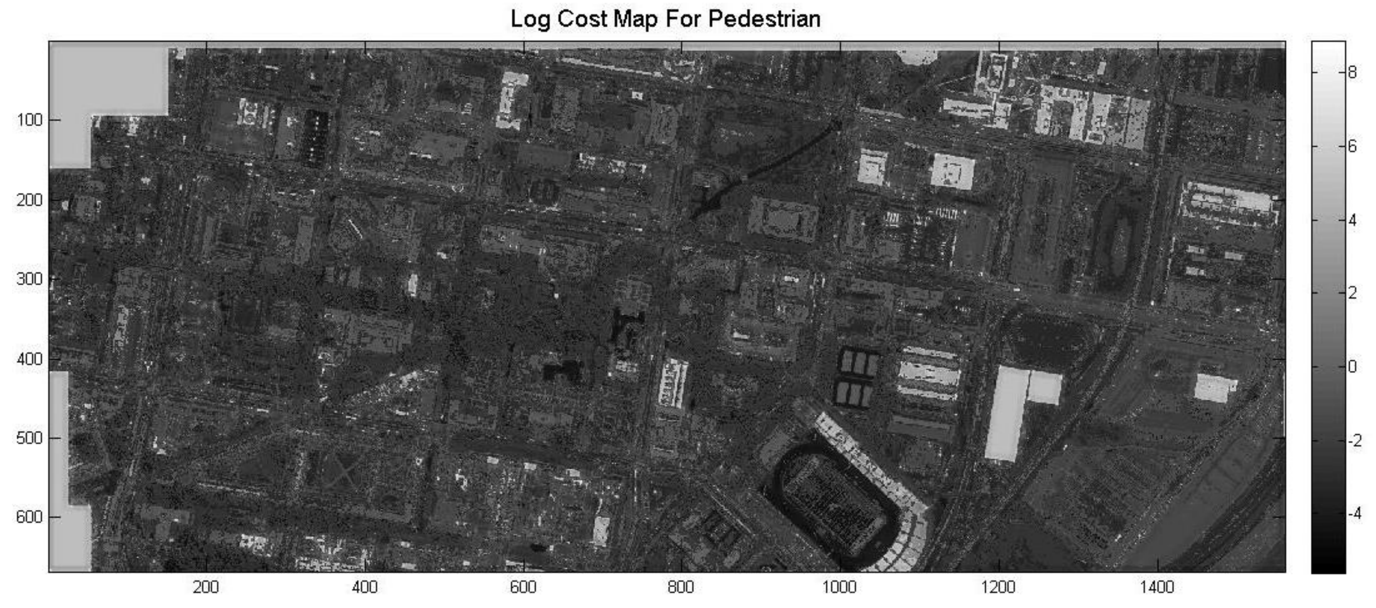
For the pedestrian it was a bit more difficult to accurately determine a model. The path planning for the pedestrian works in most situations and it takes the shortest possible routes while also sticking to the sidewalks, pathways and routes with trees as much as possible but occasionally it does move over buildings.

For future work I would like to include non-holonomic dijkstra for the car instead of assuming it is holonomic.

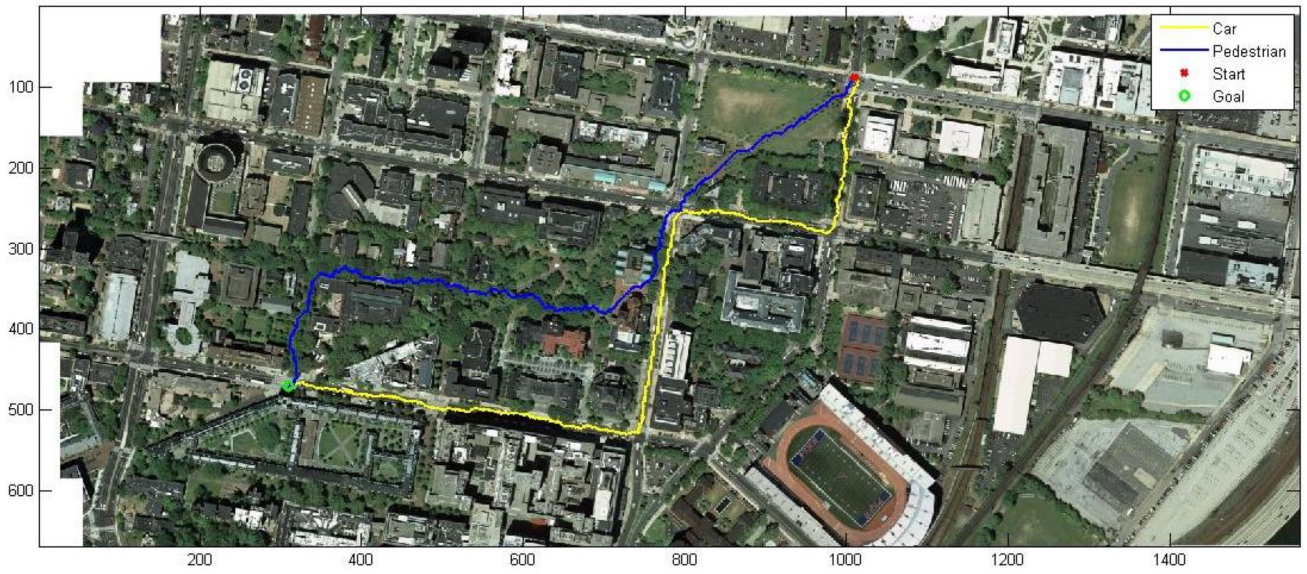
CAR LOG COST MAP

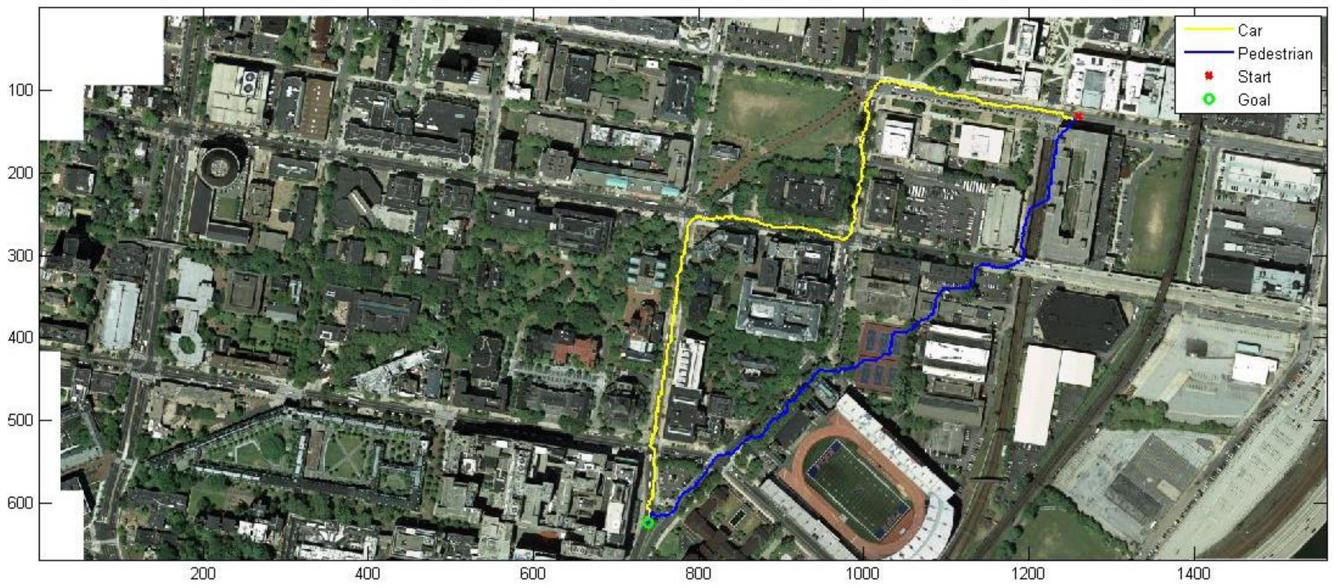


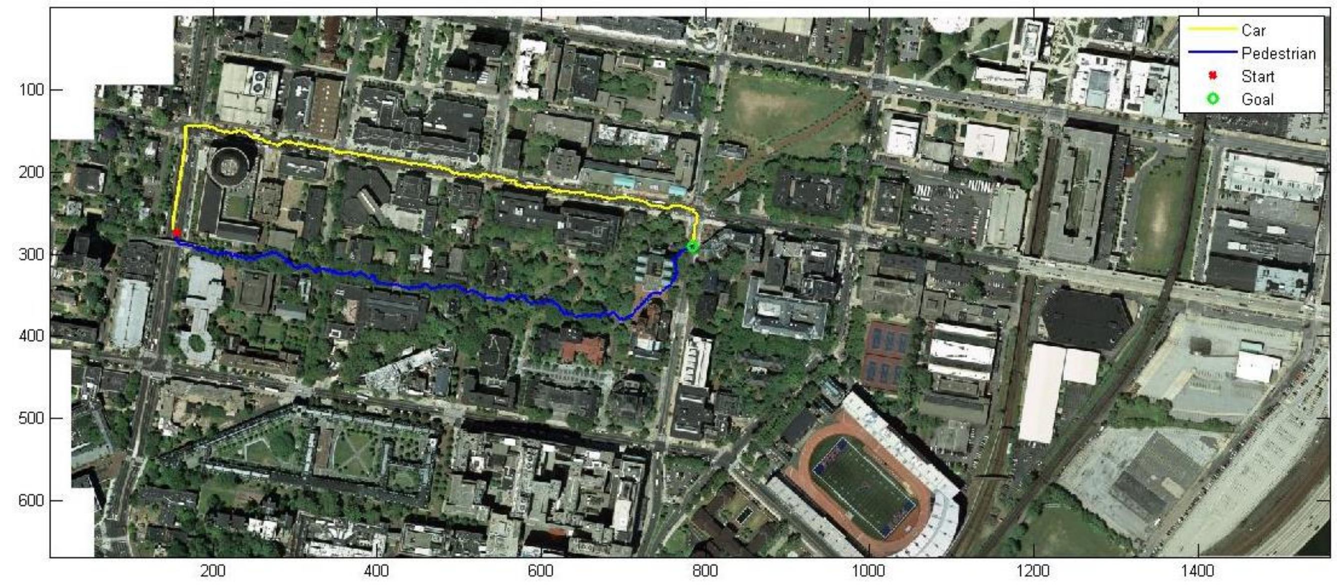
PEDESTRIAN LOG COST MAP



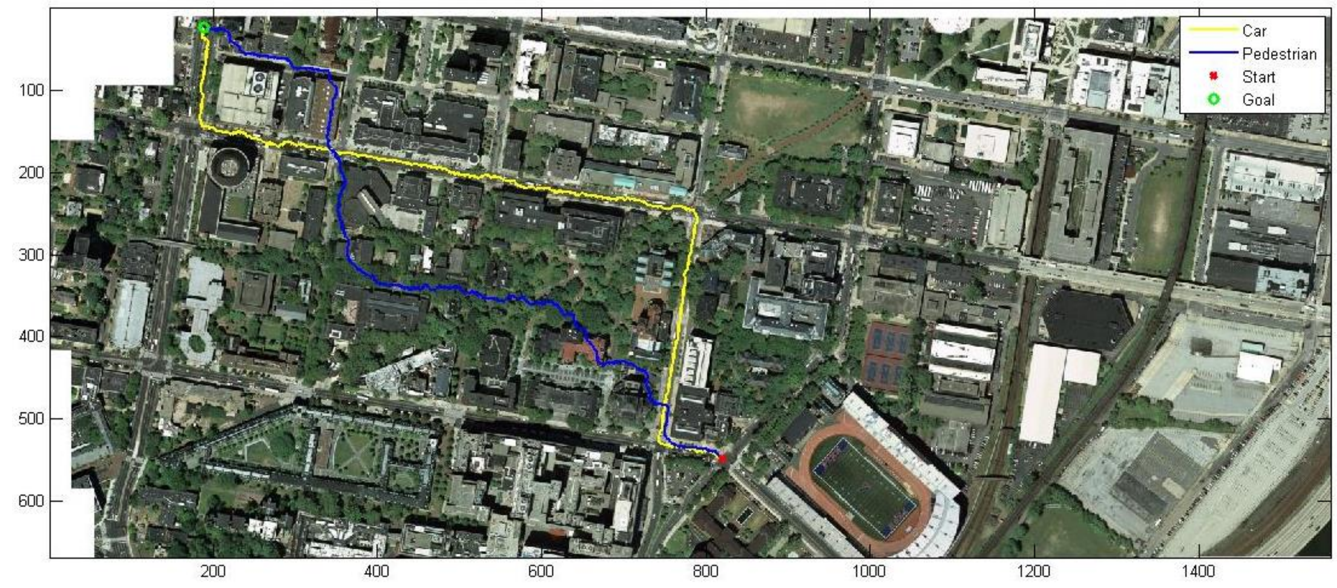
RESULTS


Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400

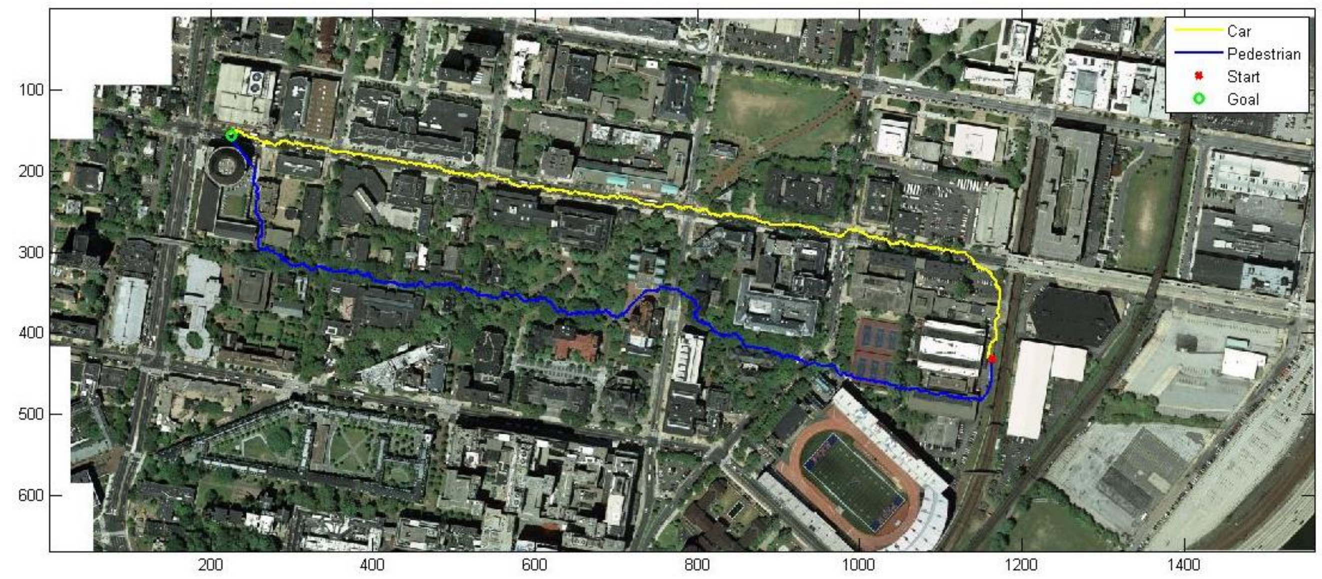

Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400

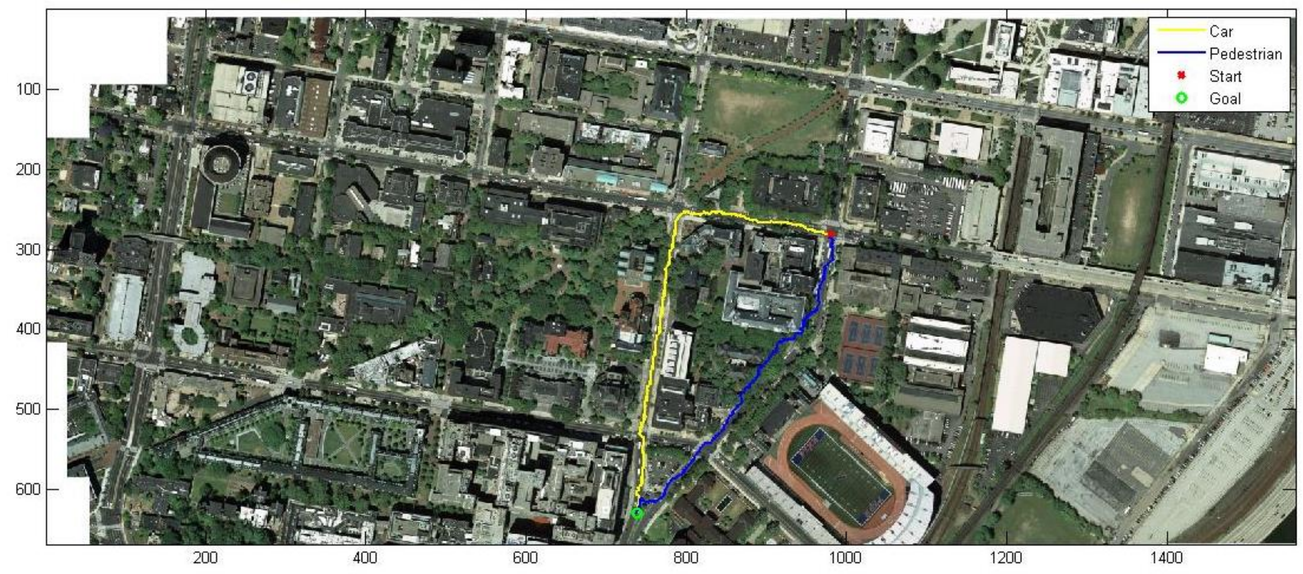


Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400



Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400


Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400


Car
Pedestrian
Start
Goal
100
200
300
400
500
600
200
400
600
800
1000
1200
1400

References

[1] ND Ratliff, D Silver, JA Bagnell Learning to search: Functional gradient techniques for imitation learning - Autonomous Robots, 2009 - Springer

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.