Salin147 Challenge Solution

## Problem 1: 2D Path Planning for Static and Dynamic obstacles

### 1.2) Discuss the performance of your solution and how you can further optimize it for faster computation or better path quality.

Ans)

My solution is based on a A\* path planner. The solution provides the smallest path from the start to the goal position while avoiding static and dynamic obstacles. I calculate the next closest node for A\* path planner at every frame in the solution.

With the current setup, I calculate a new path if any dynamic obstacle comes within 1 unit euclidean distance of the current optimal path. My solution works extremely well under computation time limits as it calculates a new path with an average time of 0.0075seconds. Although there are a few limitations where we can further optimize it

**1)**

**Limitation:**

The path calculates a trajectory that is optimal but probabilistically unsafe. The path is too close to the static obstacles.

**Solution:**

A solution can be to enlarge the size of the static obstacle polygon to ensure safety. That would be a sub-optimal result, but still safer. Another solution could be to only enlargen those parts of the obstacle that are close to the first iteration of the path created.

2)

**Limitation:**

The path uses a graph based search algorithm which depends on deterministic values of the environment parameters.

**Solution:**

In real case scenarios when working with probabilistic values, a combination of offline and online path planners can help solve this problem. For example, the graph based planners can be used to calculate an optimal path when the vehicle is stationary, and then during runtime a mixed integer programming solver with safety constraints can be used to calculate new trajectories as the costmap changes. This method can be considered as a combination of Sampling Based Path Planning and Model Predictive Path Integrator. I am conducting similar work in my research.

3)

**Limitation:**

The path does not consider the dynamics of the robot during planning.

**Solution:**

A Model Predictive Control approach can be used to predict the paths in the future for a receding horizon of N steps and then apply the control input for step 1. This method allows us to optimize a cost function, while satisfying the vehicle dynamic constraints. The computational time can be reduced by combining the same with offline maps/ path planners.

## Problem 2: Instance Segmentation for Images of Field Lines

### 1.2) Discuss the advantages and disadvantages of your method, explain the reasons behind any shortcomings, and suggest possible improvements to achieve better results.

### **Advantages:**

1. The method is based on well documented OpenCV functions that allow us to complete controllability over the output.
2. The method can applied to any shape, size or orientation of crop lines without requiring any pre-processing

### **Disadvantages:**

1. Currently, I am manually cropping the bounding boxes for the vertical, horizontal and curved crop lines
2. There are too many parameters for the OpenCV functions to play with in order to determine a close to optimal solution.
3. A single set of crop lines from (vertical,horizontal,curved) takes around 5 seconds to process completely.

**Reasons:**

1. A major reason for the shortcoming is the HoughLines function that calculates the line coordinates from the canny edge detector. I am currently unable to get the most optimal parameters to detect the lines the edges
2. In the vertical images, I have found the left and right edges of the green crop lines, but not the coordinates of lines passing through the center of the green crop.

**Solutions:**

1. A solution to reduce computation time is to perfect the mask for a determining the crop lines and then create a dataset that can be used to train a convolutional U-Net to mask all such images
2. If the lines are always close to vertical, horizontal and parallel, then they can be easily approximated within the bounding boxes based on the normal distance between each parallel line in pixels.
3. Another solution would be to automate the hyperparameter iteration for the OpenCV functions.