PERCEPTION FOR AUTONOMOUS ROBOTS

MIDTERM EXAM

Problem 1.A:

The list of all perception sensors required to design a wall painting robot is given below:

1. RGBD Camera –
   A Red Green Blue-Depth camera will be required by the robot to capture images and identify which wall needs to be painted. A map of the wall can be generated by processing the images which can then be used by the robot to plan the path of the painting brush/spray. Also, the depth obtained can be used to detect and avoid obstacles while approaching the wall or while carrying out the painting plan. This sensor can also help to ensure that the robot is at a correct distance from the wall for optimal painting.

2. Colour sensor –
   This sensor will be required to get the precise matching of colours and to check if the paint is being applied properly or not. If the paint brush/spray is not efficient, there will be uneven patches of coating, and this will help the robot to identify if certain areas require multiple coats of paint.

3. LIDAR sensor –
   Light Detection and Ranging sensor can be used to provide a detailed point cloud map in 3D space. This can then be combined with the RGBD camera sensor to overcome any obstacle in the painting path and also to find the best optimal plan for painting and reaching the wall.

4. Force sensor –
   This type of sensor can help the robot perceive the amount of force required to apply paint properly against the wall. Also, it allows the robot to apply less or more paint wherever required.

Problem 1.B:

The list of all major perception sensors required to design the underwater robot is given below:

1. Stereo Cameras:
   Even in harsh underwater conditions, where visibility is less, these cameras can provide accurate depth information to the robot. This will in turn help it to identify the best angle of approach for the broken pipe and help to avoid obstacles in its path.

2. Sonar sensor:
   These sensors are typically used to provide a 3D map of the environment which in turn can be combined with the stereo cameras to help facilitate proper detection of the broken pipe.

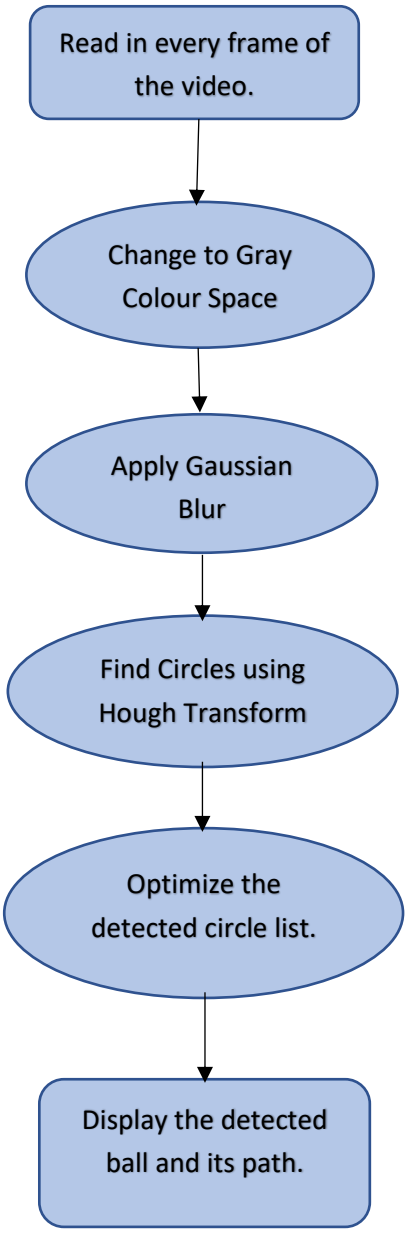
3. Magnetic sensor:
   This sensor can help to detect the pipe. Furthermore, by detecting the changes in the magnetic field around the broken pipe, the leak can then be located accurately.

4. Temperature/Thermal sensor:
   As the name suggests, this sensor can help to detect the location of the broken pipe if the leak causes sudden increase or decrease in temperature.

5. Pressure sensor:
   By detecting an increase in pressure from the broken pipe, the location of the leak can then be detected.

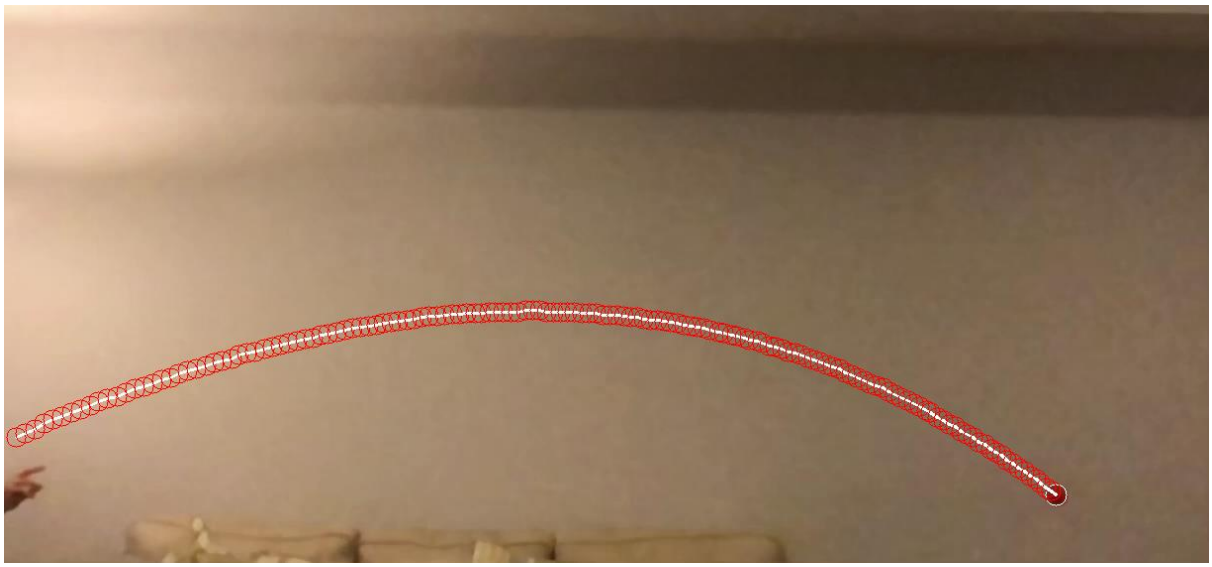
Problem 2:

To detect the ball effectively in the video for each frame, the following image processing pipeline was followed:



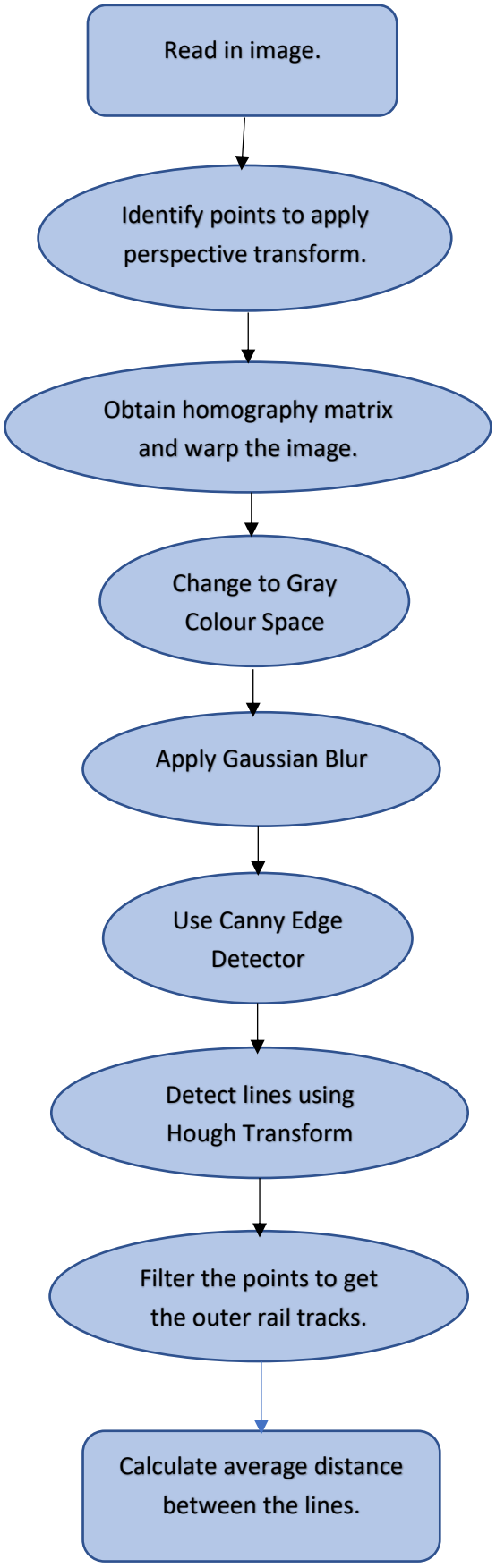
After reading in every frame of the video, changing the color space to gray and applying gaussian blur helps to remove outliers and enhances the edges for the Hough Transform to work effectively. To detect circles accurately, "cv.HoughCircles()" is used and tweaking the function's parameters helped to find the circles in each frame. With the list of center points of circles detected stored, this can be further optimized by calculating the distance between the current circle and the previous detected circle and combining them to give only a single circle if their centre points are close to each other. Finally, the path of the ball can then be drawn over each frame using these centre points.

RESULTS FOR 2:



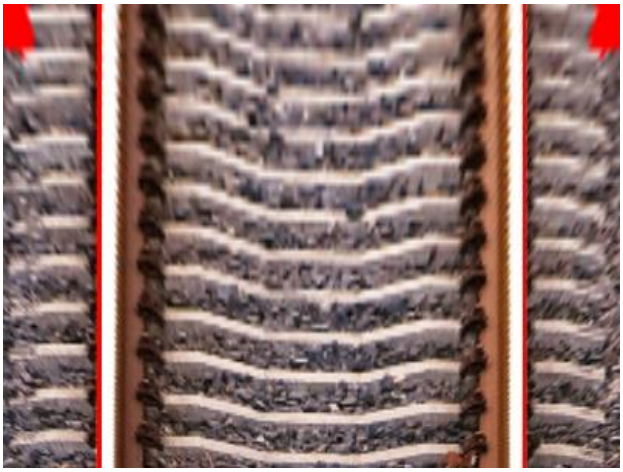
Problem 3:

To find the distance between track rails in the given image, the following image processing pipeline was followed:



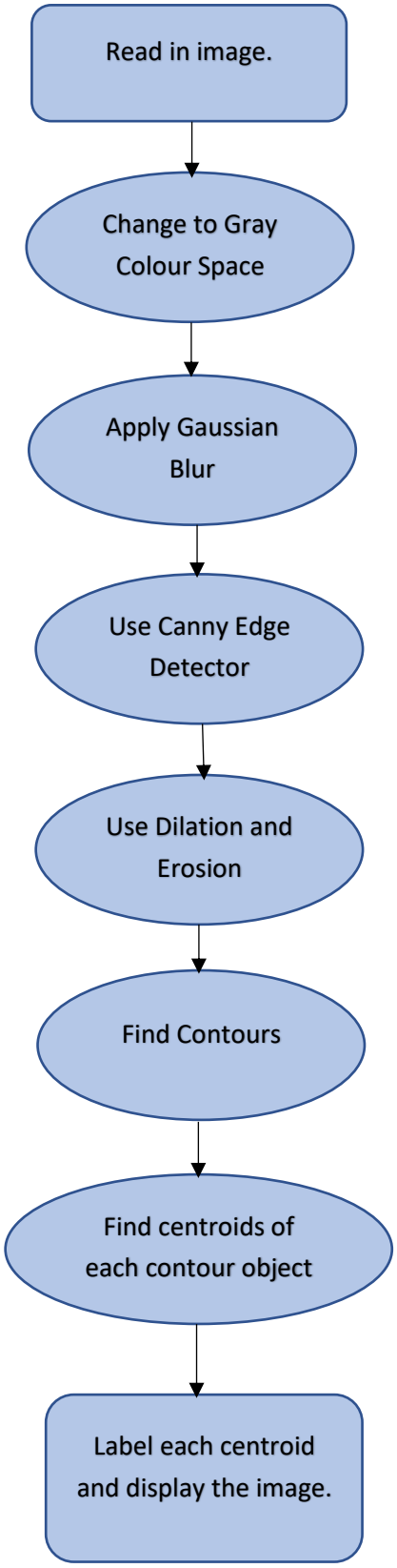
Read in the image and then detect the points along which the image needs to warp. Warp the image and change the image to grey colour space. Apply gaussian blur to remove any outliers. Use canny edge detectors and tweak its parameters to only detect the outer edge points of the train track. Using “cv.HoughLinesP()” helps to classify edge points as lines by tweaking its parameters and get a list of the points that classify it as a line. Filter out the points between the left and right rail track. Finally calculate the average of x-coordinates of left and right tracks and calculate the absolute difference between them to give the average distance between the outer rail tracks.

RESULTS FOR 3:



Problem 4:

To detect each hot air balloon in the given image, the following image processing pipeline was followed:



Read in the image, change to grey colour space, and apply gaussian blur to remove any outliers. Use canny edge detectors and tweak its parameters to only detect the outer edge points of balloons. Apply dilation and erosion functions to further optimize edge points. Find contours to classify and segregate each balloon object. Finally, find the centroid coordinates of each contour/balloon object and label them with different colours.

RESULTS FOR 4:

Problem 5:

[25 13 2 11 4 6 15 22]

To apply k-means clustering, we begin by selecting three random numbers and set them as initial centroids. Let's take 2, 11, 22 as our initial centroids.

Step 1:

Centroids = [2, 11, 22]

Step 2:

We proceed by calculating the distance between each data point and each centroid. Assign the data points to the nearest centroid.

For 25; 25-2 = 23; 25-11 = 14; 25-22 = 3

Since the smallest distance is to 22, we assign 25 to cluster of 22.

For 13; 13-2 = 11; 13-11 = 2; 22-13 = 9 → Assign to 11

For 4; Assign to 2

For 6; Assign to 2

For 15; Assign to 11

The three clusters we get are: [2, 4, 6]; [11, 13, 15]; [22, 25]

Step 3:

We proceed by taking means of the clusters:

Cluster 1: (2+4+6)/3 = 4

Cluster 2: (11+13+15)/3 = 13

Cluster 3: (22+25)/2 = 23.5

Repeat Step 2 and 3, till a convergence is reached considering the new centroids.

Centroids = [4, 13, 23.5]

For 25; Assign to 23.5

For 2; Assign to 4

For 11; Assign to 13

For 6; Assign to 4

For 15; Assign to 13

For 22; Assign to 23.5

The new clusters are: [4, 2, 6]; [13, 11,15]; [25, 22]

Cluster 1: (2+4+6)/3 = 4

Cluster 2: (11+13+15)/3 = 13

Cluster 3: (22+25)/2 = 23.5

Centroids = [4, 13, 23.5]

As we can see a convergence of data points and centroids is reached, therefore the clusters are:

[2,4,6]; [11,13,15]; [22,25]