ENPM662

Midterm Exam

SURIYA SURESH

TABLE OF CONTENTS

Table of Contents ii
List of Illustrations iii
Chapter I: Problem One 1
1.1 Sensors used for a Wall Painting Robot 1
1.2 Sensors Used for a Underwater Robot 2
Chapter II: Problem 2 4
2.1 Pipeline Followed 4
2.2 Loading Video 4
2.3 Masking Lower Part of Image 4
2.4 Blurring and HSV Thresholding 5
2.5 Hough Circles 5
2.6 Tracking the Ball 6
2.7 Problems Faced and Solutions Implemented 6
Chapter III: Problem 3 7
3.1 Pipeline Followed 7
3.2 Resizing the Image 7
3.3 Defining Region of Interest 7
3.4 Getting Perspective Matrix 8
3.5 Warping Image 9
3.6 Thresholding and Hough Lines 9
3.7 Finding the Distance of The Tracks 10
3.8 Problems Faced and Solutions Implemented 11
Chapter IV: Problem 4 12
4.1 Pipeline Followed 12
4.2 Resizing the Image 12
4.3 Converting to Gray-scale and Blurring Image 12
4.4 Thresholding 12
4.5 Find Contours and Bounding Box 13
4.6 Counting Number of Balloons 13
4.7 Problems Faced and Solutions Implemented 14
Chapter V: Problem 5 15
5.1 Steps Involved 15
5.2 Mathematical Steps Done for Given Problem 15

LIST OF ILLUSTRATIONS

Number *Page*

2.1 Masking Lower Part of circle 5
2.2 Masked Image 5
2.3 Tracked Circle Output 6
3.1 Getting ROI using plt.show 8
3.2 Perspective Matrix Obtained 8
3.3 ROI of image 8
3.4 Bird's Eye View of Image after Warping 9
3.5 Masking of Lines 10
3.6 Lines Detected 11
4.1 Threshold Output 13
4.2 All detected Balloons 14

Chapter 1

PROBLEM ONE

The kind of perception sensors and the reasons for choosing them was explained for two applications.

1.1 Sensors used for a Wall Painting Robot

For a robot to perform indoor wall painting tasks, we would consider using CMOS cameras, a LIDAR, ultrasonic sensors and an IMU.The camera would be the primary visual aid , used to guide the robot to paint the walls , identify the colors it may need to pick up or use to pick up tools like a paint brush. The camera can be also used to customize the painting job as per needs of the user.

The LIDAR can be used for mapping a room, for localization purposes and obstacle navigation. The LIDAR scans can be used to construct a map of a place the robot is in and do planning according to the map. The LIDAR used can be a 2D or 3D LIDAR. This data can be also used to determine the heading of the robot in a unknown room and navigate if a map was not given. The main concerns would be the possibility of paint spilling on the LIDAR itself, affecting its performance and the cost.

The LIDAR in conjunction with the ultrasonic sensors can be used for obstacle detection and avoidance. The ultrasonic sensors will be able to give the distance of obstacles, which then the data from it be used by the camera or the LIDAR to plan a path around the obstacle or take required action. This also can be used to mitigate the blind spots of the camera and the LIDAR to detect unseen challenges in the environment, such as short objects near to the robot.

An alternative option for using the LIDAR can be using a inertial measurement unit in addition with a camera to do visual odometry for localization and determining the heading of the robot and would not need the expense of having to buy a LIDAR for it.

The bare essentials for such a robot would be a camera, a sensor to detect obstacles and a solution to track and determine the route for the robot to follow provided it is not supervised for most of the time and has no teleoperation done by a user for navigation in a area.

1.2 Sensors Used for a Underwater Robot

For an underwater robot, a variety of sensors can be used, primary ones being IR cameras, night vision cameras, CMOS cameras, sonar or radar sensors, IMU and ultrasonic sensors. The main drawback of using a regular camera would be low light and uncertain conditions of the water which result in dark environments which the sensor would struggle to cope with while offering decent performance.

A night vision camera would be able to see in pitch black conditions and provide good details in such conditions. However, it maybe not be used in the daytime and may not give accurate colors in low light. Alternatively, IR cameras can be used which are cheaper and use infrared rays to see in the dark, which can provide good details, but the image will be without color. Both types of cameras can get affected by murky and turbulent waters, resulting in degraded image quality.

A LIDAR can be certainly used to map the surrounding, providing a high-definition image of the surroundings which can be used to detected smaller details such as cracks in a pipe. However, LIDAR is expensive and can be made more expensive as it needs to be water resistant when used underwater. LIDAR also can struggle when there is a lot of impurities in the water, affecting the range and quality of the data it provides to the robot.

Ultrasonic sensors can be used to for obstacle sensing , leak detection and navigation. A ultrasonic sensor can be used for mapping a pipe the robot maybe traveling in and detect leaks by the data it receives from the reflected sound waves. Similarly, it can be used to map the area around the robot for a certain distance. These can get affected by murky water as impurities can reflect waves and cause false readings. A sonar sensor can be used to map the area around the robot for a longer distance compared to a ultrasonic sensor. These are the main sensors used to map ocean beds and areas underwater. These provide high resolution images of the environment. Radar can be used for similar purposes but sonar uses sound-waves while radar uses radio-waves for mapping. These sensors are expensive but provide accurate and reliable information in a variety of underwater environments.

CMOS cameras would not be of much use in low light environments, but they can be used for collecting details, pictures and information needed apart from navigation using illumination sources in underwater low light conditions. They also can be used in bright light unlike night vision cameras tend to suffer and be used when the lighting is better by the robot.

A IMU with sensor fusion with any vision sensor can be used to track the distance, orientation and the path a underwater robot has taken without needing to use other

aids reliably which is crucial underwater as it may not be possible to for the robot to have constant communication with a master while carrying out tasks.

Chapter 2

PROBLEM 2

Problem 2 involved tracking a ball thrown from a corner till it dropped out of frame using Hough Transform.

2.1 Pipeline Followed

The following steps were followed to solve problem two.

- 1.Loading Video
- 2.Masking Lower Part of Image
- 3.Blurring and HSV Thresholding
- 4.Hough Circles
- 5.Tracking the Ball

2.2 Loading Video

The video was loaded from a file and was passed into a loop frame by frame.

2.3 Masking Lower Part of Image

The lower part of the image was masked as to reduce the noise caused by objects on the lower part of the video.The mask also covers the hand which gave many false positives during hough transform.

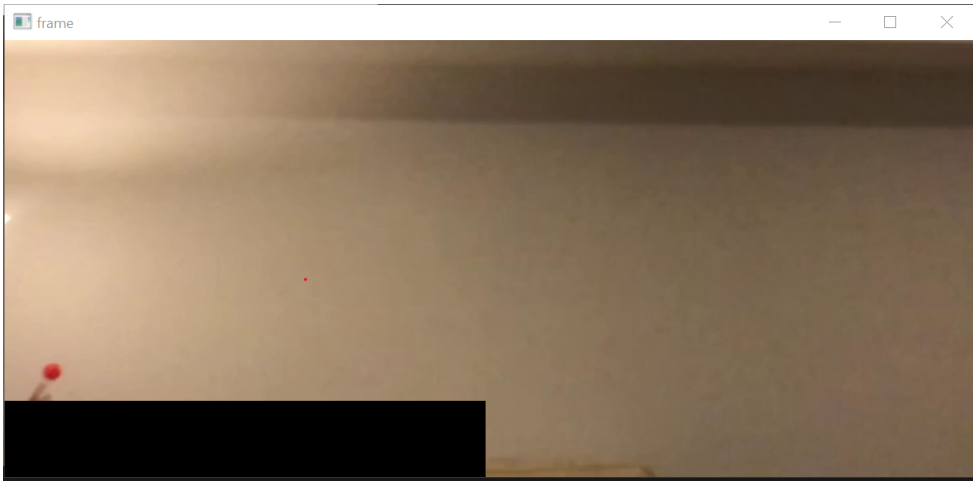

Figure 2.1: Masking Lower Part of circle

2.4 Blurring and HSV Thresholding

To reduce the effects of noise on the hough transform, Gaussian blue was done.Afterwards, using cv2.inrange, a binary mask was obtained of the frame for which erosion and dilation was done to further reduce the effects of noise on the output.Afterwards, the frame was processed for detecting the ball using hough transform.

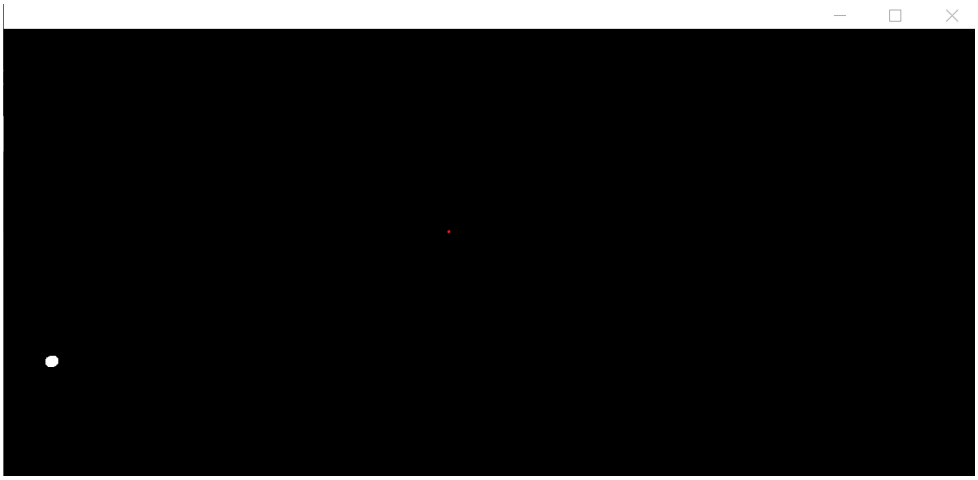

Figure 2.2: Masked Image

2.5 Hough Circles

The opencv function, cv2.houghcircles [1] with parameter of min distance between circles as 1.5, min and max radius of detected circle to be between 3 and 10 , as the given diameter of the ball was 11 pixels was used.The output of houghcircles is drawn using cv2.circle which is able to track the ball for most frames. Due to

noise and slightly off HSV mask values, the ball was not able to be tracked for a few frames at the end of the video.

2.6 Tracking the Ball

After obtaining the output of the houghcircles, the circles were drawn on the original frame before image processing was done.The tracking of the ball is relatively accurate w.r.t to the position of the ball in the frame.

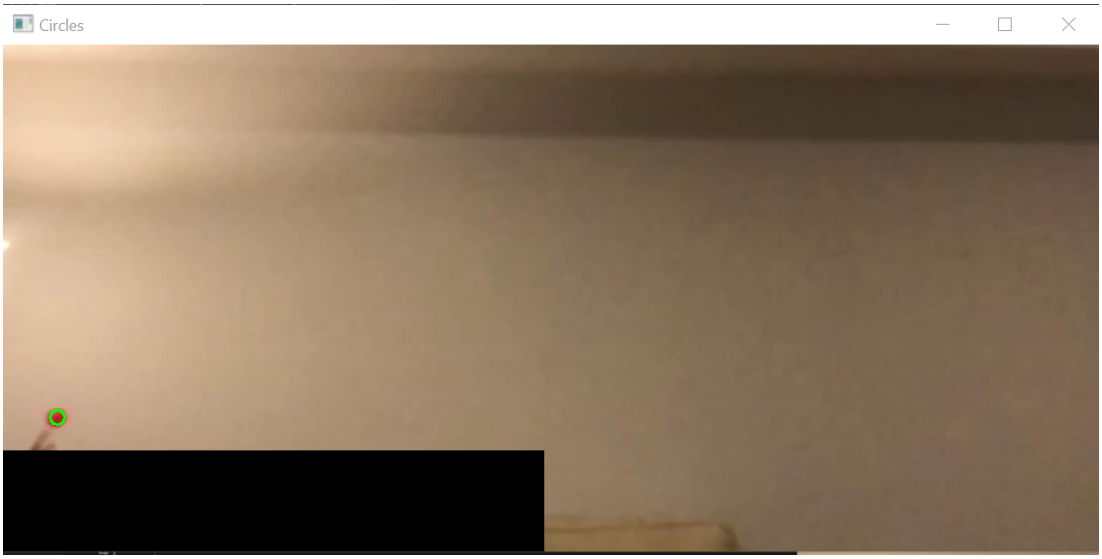

Figure 2.3: Tracked Circle Output

2.7 Problems Faced and Solutions Implemented

This section describes the problems faced and solutions :

- Giving the gray-scale image to the Hough Transform function,a lot of the frames were not detected for any possible circles.Hence, HSV filtering and masking had to be done to ensure better accurate images.
- The presence of noise in the source video caused a lot of false positives, so the frames had undergo image processing to reduce the presence of false positives.It was also due to presence of similar colors of the target ball being in other locations in the video that caused this also.

Chapter 3

PROBLEM 3

Problem 3 involves getting the top-down view of railway tracks and calculating the average distance between the tracks.

3.1 Pipeline Followed

The following steps were followed to solve problem four.

- 1.Resizing the image
- 2.Defining Region of Interest
- 3.Getting Perspective Matrix
- 4. Warping Image
- 5. Thresholding and Hough Lines
- 6. Finding the distance of the tracks

3.2 Resizing the Image

The source image was large so it was reduced to 640*480 for making calculations faster and to make the output viewable on a display.

3.3 Defining Region of Interest

The region of interest for this image was the vanishing points of the railway with respect to the position of the railways tracks at the bottom of the image.The final points were considered to be a rectangle whose position was relative to the ROI which was in the form of a trapezoid. The points were taken roughly by using plt.imshow.

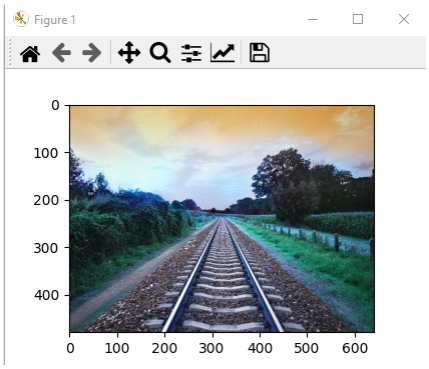

Figure 3.1: Getting ROI using plt.show

3.4 Getting Perspective Matrix

After getting the ROI and the final points for the required perspective , we use the function, cv2.getperspective [2] which returns a 3x3 matrix which then is used to warp the perspective of the image to a bird's eye view.

The region of interest that was defined for this image is shown below.

```
[[-3.51421189e-01 -8.68217054e-01  3.10614987e+02]
 [-3.29894842e-15 -2.46253230e+00  6.54883721e+02]
 [-6.17801170e-18 -4.37123170e-03  1.00000000e+00]]
```

Figure 3.2: Perspective Matrix Obtained

Figure 3.3: ROI of image

3.5 Warping Image

We warp the image using cv2.warp perspective [2],that gives us the top view of the image. We can assume that since railway tracks are parallel to each other, the view we get will show the distance between the tracks for the whole of the tracks in the source image that go till the vanishing point. After warping , we get a almost 90 degree view of the tracks that can be used to determine the distance between the tracks.

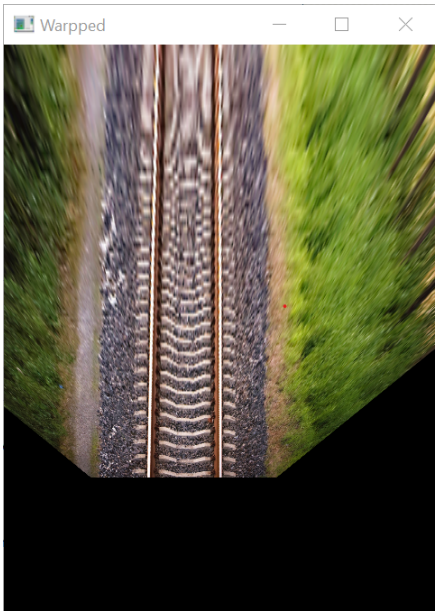

Figure 3.4: Bird's Eye View of Image after Warping

3.6 Thresholding and Hough Lines

Thresholding is done to isolate a specific color of the tracks which allows for easier calculation of the distance between the tracks. The output of this operation gives a set of lines which is passed to the Hough lines Probabilistic function [3] of opencv which finds the lines. A resolution of one pixel and one radian is used for ρ and θ with a threshold of 50. The output obtained has many parallel lines that are close to each other and are far away on the image

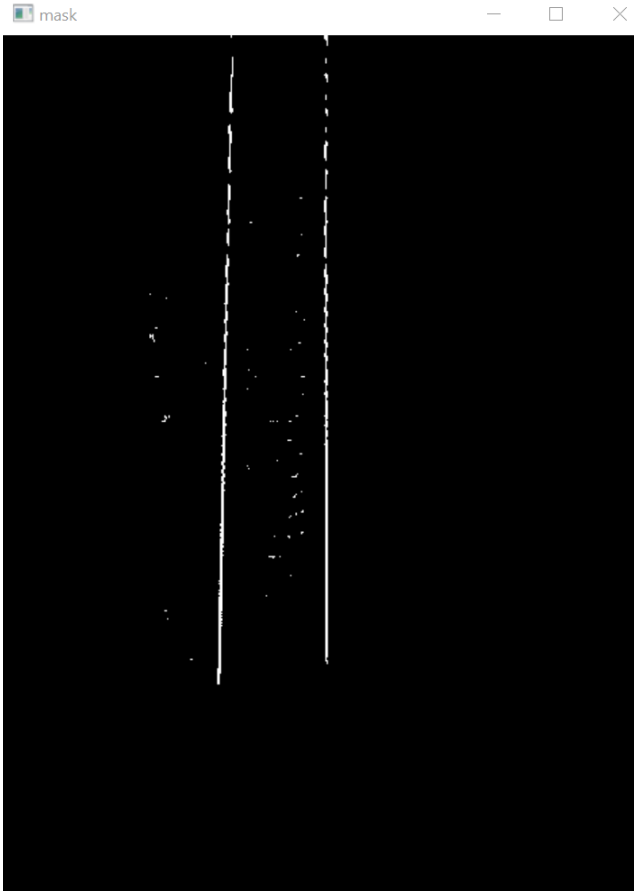

Figure 3.5: Masking of Lines

3.7 Finding the Distance of The Tracks

The output of Hough Lines is filtered based on the distance between lines to be greater than a certain threshold so that they are not close to each other and the average distance is finally found for the tracks , which is 75.25 pixel units.

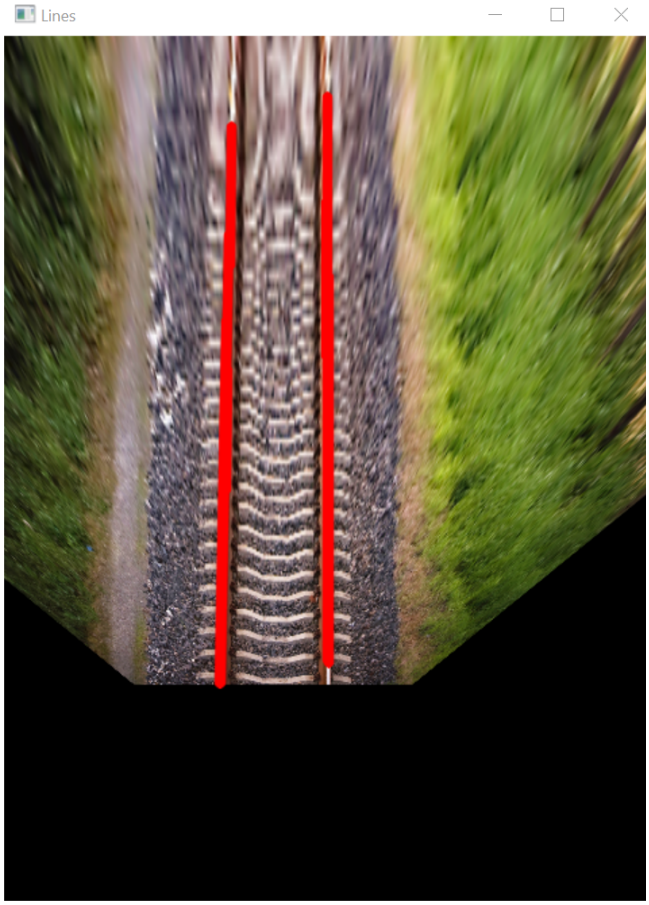

Figure 3.6: Lines Detected

3.8 Problems Faced and Solutions Implemented

This section describes the problems faced and solutions :

- Determining the ideal points to get a decent top view was a major hurdle.Multiple trials of points were found till the output of the warped image was satisfactory.
- The thresholding based on color was very narrow as the colors of the tracks were very similar.Focusing on the middle part of the tracks gave the desired threshold output which allowed for hough lines to be found faster and easily.
- Filtering the parallel lines had to be done to prevent them from affecting the final mean of the distances due to lines being close to each other having lower distances and affecting the mean.

Chapter 4

PROBLEM 4

Problem 4 involves counting the number of hot-air balloons in a given image.

4.1 Pipeline Followed

The following steps were followed to solve problem four.

- 1.Resizing the image
- 2.Converting to Gray-scale and Blurring Image
- 3.Find Contours and Bounding Box
- 4. Counting the number of Balloons

4.2 Resizing the Image

The source image was large so it was reduced by a factor of 4 for viewing the output on a display easier.

4.3 Converting to Gray-scale and Blurring Image

The image was converted to gray-scale using a build-in opencv function as the cv2.findcontours[4] function prefers a gray-scale or binary image for optimal results.The gray-scale image is represented in various shades of white and black instead of having 3 channels of BGR.This also reduces processing time of the image.

The image was blurred using Gaussian and Median Blur to reduce the sharp features and to smooth out the image so that while finding contours the chances of false positives are reduced to a extent. The kernel size for Gaussian blur was (21,21) and for median , it was 9.

4.4 Thresholding

Thresholding was done to filter out the objects of interest from rest of the image, that is the background.The values for thresholding were determined using a histogram of the intensities of the image.After thresholding , the majority of the part remaining are the balloons , a few parts of the background can be also seen in the output.

Figure 4.1: Threshold Output

4.5 Find Contours and Bounding Box

After the thresholding , the cv2.findcontours function with parameters cv2.RETR-TREE and cv2.CHAIN-APPROX-SIMPLE,which ensure that the hierarchy of the features and all detected curve points are returned.

After the output of this function is fed to cv2.bounding which determines a rectangle that bounds the objects of interest and can be drawn using the normal cv2.rectangles function.A condition for ensuring false positives being filtered out was implemented , which used the absolute difference of the height and width of the rectangles to be below a certain threshold for the rectangle not to be a false positive.

4.6 Counting Number of Balloons

After all possible bounding rectangles are detected and drawn using random colors for each box , the number of balloons detected is 16 and counts balloons that are over each other as a single balloon.The output is displayed below.

Figure 4.2: All detected Balloons

4.7 Problems Faced and Solutions Implemented

This section describes the problems faced and solutions :

- The output of find contours detected parts of the background and the entire image as rectangles.After filtering out rectangles with conditions, the false positives are zero now.
- Using a gray-scale image for find contours gave lots of false positives that needed tuning to be reduced.Thresholding reduced the number of false positives by a large extent.
- Filtering false positives was initially done by finding the area of the drawn contours which resulted in false negatives.Switching to filtering based on values of the width and height of the bounding rectangles reduced the false negatives to zero and gave no false positives.

Chapter 5

PROBLEM 5

Problem 5 involves finding the clusters of given points using K-Means.

5.1 Steps Involved

- 1.Three random points are taken and the Manhattan distance is calculated for the other reminding points.Once the distances are found,then each point is assigned to a cluster based on the nearest distance of it from the taken sample points.
- 2.After the points are reassigned into clusters , the mean of the points are taken and the resulting means are the new sample points for which again the given points are clustered based on the closest distance to them.
- 3. Step 2 is repeated until the cluster arrangements do not change and the mean values of each cluster converges for ever iteration.

The final clusters are the clusters formed using K-means and the final means are the final sample points taken to determine the cluster arrangement.

5.2 Mathematical Steps Done for Given Problem

Here the points,25,13,2,11,4,6,15 and 22 are given, a total of 8 points, we take 6, 15 and 22 as the three sample points, calculating the Manhattan distances , we get,

$$Distance\,1 \quad (6) = [19 \quad 7 \quad 4 \quad 5 \quad 2 \quad 9 \quad 16] \tag{5.1}$$

$$Distance\,2 \quad (15) = [10 \quad 12 \quad 13 \quad 4 \quad 11 \quad 9 \quad 7] \tag{5.2}$$

$$Distance\,3 \quad (22) = [3 \quad 12 \quad 20 \quad 11 \quad 18 \quad 16 \quad 7] \tag{5.3}$$

Calculating the New Clusters ,we get

$$Cluster\,1 \quad (6) = [4 \quad 6 \quad 9] \tag{5.4}$$

$$Cluster\,2 \quad (15) = [15 \quad 11 \quad 13] \tag{5.5}$$

$$Cluster\,3 \quad (22) = [25 \quad 22] \tag{5.6}$$

Calculating the means of the cluster and get it as 4,13 and 23.5. Rearranging clusters , we get:

$$Cluster\,1 \quad (4) = [4 \quad 6 \quad 9] \tag{5.7}$$

$$Cluster\,2 \quad (13) = [15 \quad 11 \quad 13] \tag{5.8}$$

$$Cluster\,3 \quad (23.5) = [25 \quad 22] \tag{5.9}$$

If we continue this for a few more iterations, we can see that the cluster values remain the same and the means are also the same.Hence we found the clusters, it's respective means using K-means clustering.

REFERENCES

[1] Opencv Docs. *Hough Circles*. https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html. [Online]. 2022.

[2] OPEN-CV docs. *Perspective Transform*. https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html. [Online]. 2022.

[3] Opencv Docs. *Hough Line*. https://docs.opencv.org/3.4/d3/de6/tutorial_js_houghlines.html. [Online]. 2022.

[4] Opencv Docs. *Contours*. https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html. [Online]. 2022.