Lab report for the course of 'Robot Planning and it's Application'

Ayushi Shah Vallabh Ansingkar

25th October 2020

1 Introduction

This report summarizes the work performed by Ayushi Shah and Vallabh Ansingkar on the lab assignments for the course of 'Robot Planning and it's Application'

2 Generic Image listener

This function takes an input image from the image listener node which is the image of the arena as seen by the camera. The image can be saved in a local folder by pressing the button 's'.

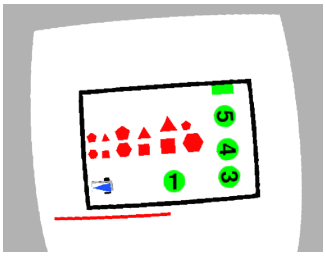

Figure 1: Raw Image

3 Intrinsic Calibration

In order to calculate the intrinsic camera parameters we provide a particular set of images to the file 'cameracalibration.cpp'. The images are specifically of a chessboard oriented in multiple ways along each axis and also at different positions in the frame.
After the calibration is performed we get the following data:

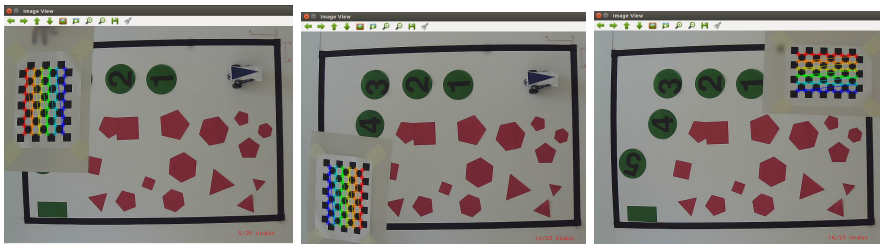

Figure 2: Intrinsic calibration using chessboard images

Camera focal lengths:
fx=1.3214280543679899e+03
fy=1.3214280543679899e+03

Image center :
cx=5.9881594712190906e+02
cy=3.3456704569178942e+02

Distortion coefficients:
k1=-3.8003885491418415e-01
k2=1.6491935276593886e-01
k3=0
p1=-7.2969684041836462e-04
p2=-8.3850194278408752e-04

4 Extrinsic calibration

This function is responsible to create a 2D Arena pose from the input 3D objecct points, also called as Extrinsic calibration. The Extrinsic calibration is performed in the following steps:
1. Initially a new config folder containing a file called 'extrinsicCalib.csv' is created in order to store the extracted 2D image points.
2. Using the function 'pickNPoints', the user specifies the 4 corner points of the arena on the input arena image.

3. From these image points the 'x' and 'y' values are extracted and stored inside 'extrinsicCalib.csv'.
4. In case the file already exists, the above steps are skipped and the 'x' and 'y' values of the image points are read from the file.
5. Finally using the 'cv::solvePnP()' method this function finds the arena pose from 3D-2D point correspondences and Extrinsic calibration is finally complete.

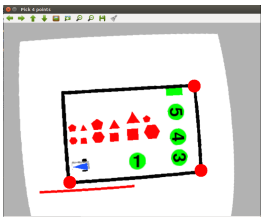

Figure 3: Picking up 4 corner points

5 Image Undistortion

This image provided by the camera is a distorted one due to the shape of the lens. This function is reponsible to undistort that image using the following inputs:
1. Distorted image
2. Camera matrix
3. Distortion coefficients
The method 'cv::initUndistortRectifyMap()' is used for the undistortion and rectification and 'cv::remap()' is used to put the undistorted pixels in the right place in the new image.

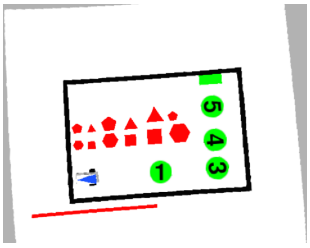

Figure 4: Undistorted Image

6 Image Unwarp

This function takes the input image and the perspective plane transform matrix in order to generate the unwarped image.

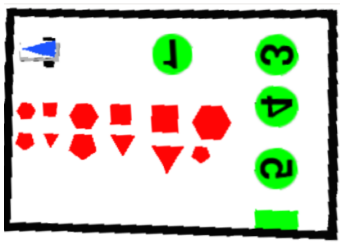

Figure 5: Unwarped Image

7 Process map

This function is used to process the whole arena. It first converts the RGB image to HSV as shown in Figure6 .Then it calls functions to detect Obstacles, victims and gate.

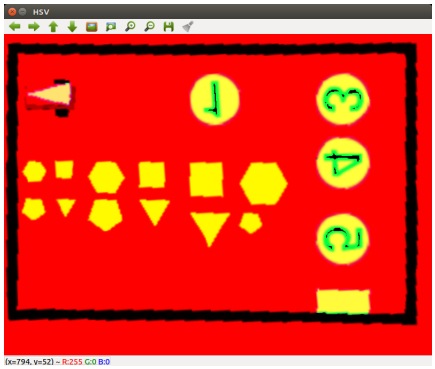

Figure 6: HSV Image

- processObstacles- This function is used to filter the obstacles by creating the mask and then drawing a contour around the obstacles in the arena. We see that in the RGB image the obstacles are of Red color. Thus we create a red mask and find contours. The filtered image is shown in Figure 7

  Next task is to draw these contours on the HSV image as shown in Figure 8

- processGate We see that in the RGB image the Gate and victims are of green color. So we create a green mask and find contours. The filtered image is shown in Figure9

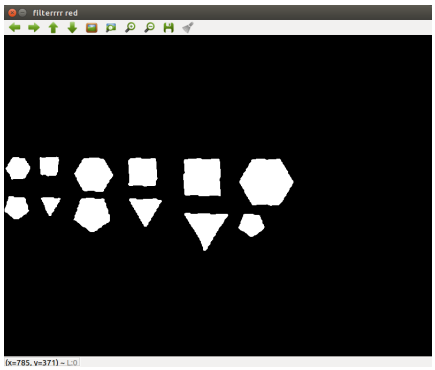

Figure 7: Obstacles Filtered Image

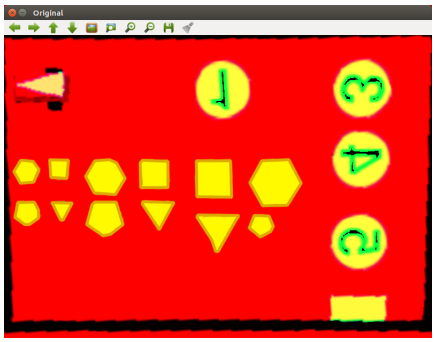

Figure 8: Obstacles Contour Image

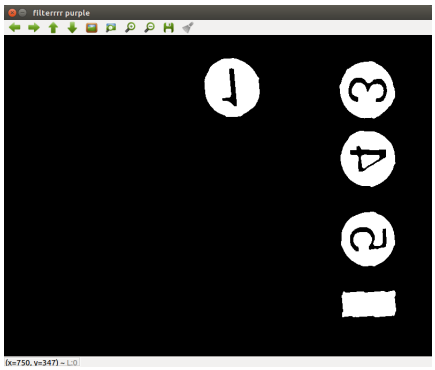

Figure 9: Victim and Gate Filter Image

The main aim of the function is to detect gate. So we draw contours around gate as shown in Figure10

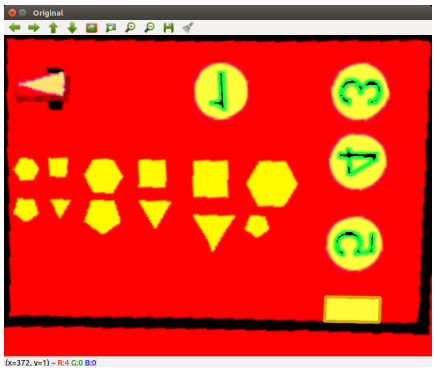

Figure 10: Gate Contour Image

- processVictims The main aim of the function is to detect victims. In the previous function we have already masked the victims. The next step is to draw the contour around victims. This is shown in Figure11

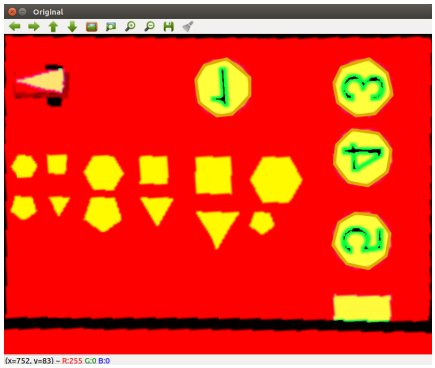

Figure 11: Victim Contour Image

8 Find Robot

The aim of this function is to detect the robot. We can see that in the RGB image the robot is of blue coloured and its shape is triangle, thus size of the contour should be equal to 3. We keep these things in mind and build a blue mask. The filtered image of the robot is shown in Figure12

Next step again will be to draw contour around the robot. This is shown in Figure13

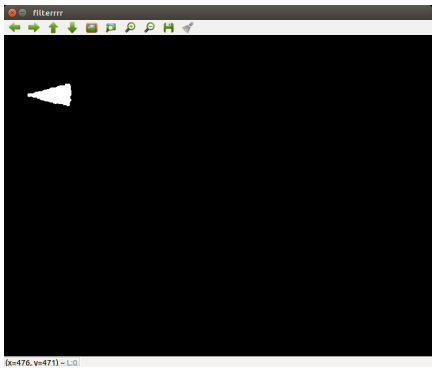

Figure 12: Robot Filter Image

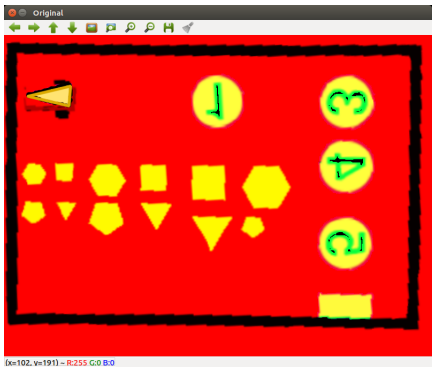

Figure 13: Robot Contour Image