ENPM673 - Perception for Autonomous Robots

Project 1

Submitted by:

Raj Shinde (raj0407)	116852104
Shubham Sonawane (shubhams)	116808996
Prasheel Renkuntla (prasheel)	116925570

Contents

1 Problem 1 - Detection

Problem 1 involves the detection of the AR tag. It also involves detecting the tag ID and its orientation. Our work flow was as below:

1. Using the contour detection function, the contours of the AR tag were detected.
2. Using the hierarchy obtained from the above function, the unecessary contours were eliminated and the ar tag contours were accurately obtained.

3. The obtained image was divided into a square matrix of size 8, and each cell was checked for white and black pixels. It was assigned black or white based on the number of pixels. Thus, a matrix made up of 1s and 0s was obtained.

4. Based on the encoding scheme provided, the tagID for each tag was obtained, along with its orientation.

The videos for all outputs can be found on: `https://drive.google.com/open?id=1r55_L47CRc19NLebnaCbZAUxM8aot-SP`

The output obtained after these steps were as follows:

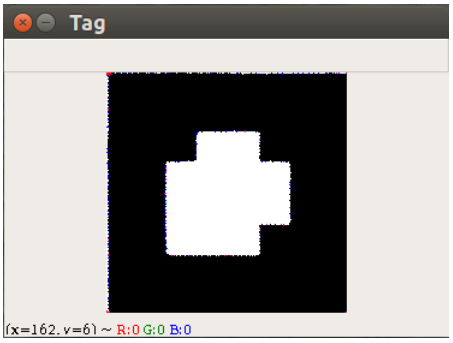

Figure 1: Tag detection

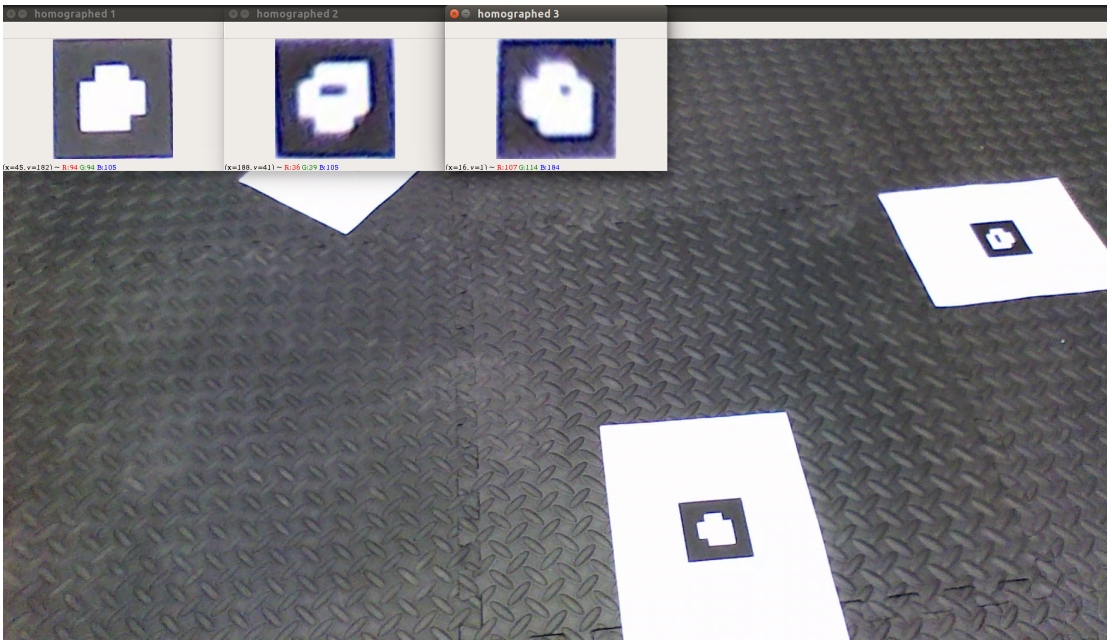

Figure 2: Multiple tags detection

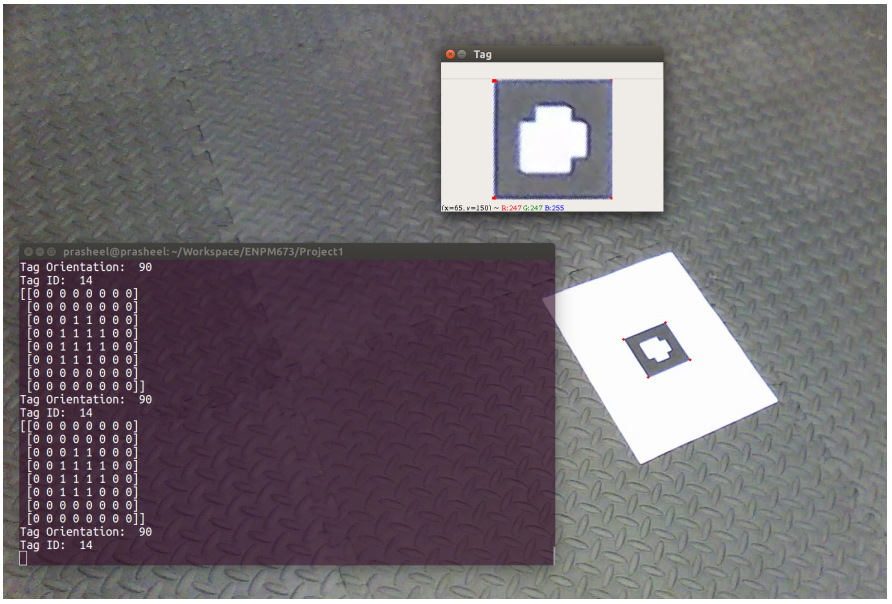

Figure 3: Tag orientation and id detection

2 Problem 2

2.1 Problem 2(a) - Superimposing an image on tag

We are provided with the template image, Lena.jpg. In order to superimpose the image on the tag, we need to perform 2 important actions: performing homography[1] estimation and performing warp[2] perspective. The orientation of the tag was obtained from the earlier section.

We developed our own functions for calculating the homography matrix and performing the warping action.

Warp function uses the homography matrix to map each and every pixel of the source image on to a different plane. First we obtained the contour points of the plane to which the image is to be warped, along with the original size of the image. Using the 2 sets of points, i.e. the size and the contour points the homography matrix was calculated. This homography matrix is then multiplied to the indices of every pixel of the image, thereby giving new pixel locations.

The results of this part are as follows:

Figure 4: Tag detection and image superimposition

Problems faced:

1. As the camera pans about in the video, the coordinates of the contour keep on changing. Initially we were trying to update the contours as and when they change. However, it was inefficient and would give an error in case a contour was not detected. Therefore, we used the orientation of the tag to rotate the image and superimpose it on the tag.

2. Using numpy arrays to perform element wise operations for the function warp(),

we were able to reduce the computation time. Traditional approach of using for loops greatly affected the frame rate and computation time.

3. Using the forward warping on the tag, produced holes on the warped tag image. Hence, we implemented inverse warping to get rid of the holes.

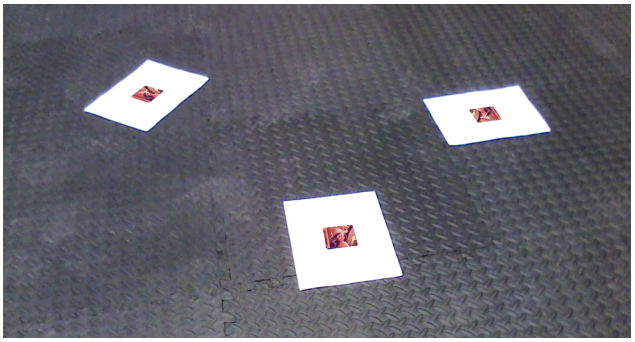

Figure 5: Tag detection and image superimposition on Multiple Tags

3 Problem 3

In this section we had to project a cube on the AR Tag. The projection matrix comes handy when dealing with this problem. Using the points obtained from contour detection, the points of the cube were calculated. The results of the same are as below:

Figure 6: Cube on Tag 0

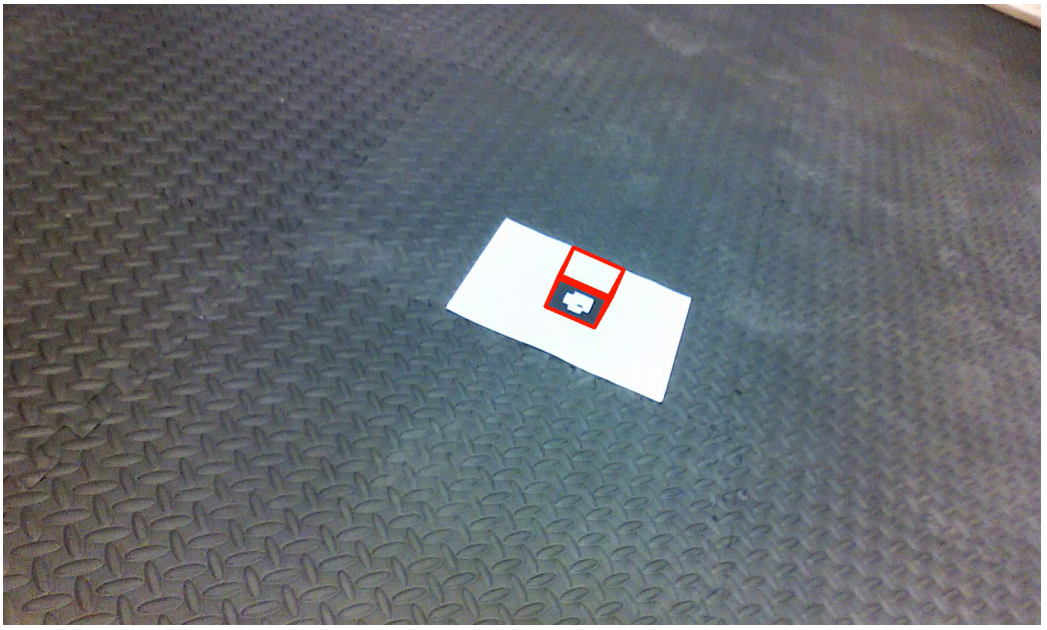

Figure 7: Cube on Tag 1

Figure 8: Cubes on multiple tag

Bibliography

[1] Jordan Hughes. Github Homography. https://github.com/hughesj919/HomographyEstimation/blob/master/Homography.py. Accessed: 2020 - 02 - 22.

[2] Rice University. Warping. https://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/warpsri.html. Accessed: 2019 - 02 -19.