ENPM673: Perception for Autonomous Robots

Project 1

Umang Rastogi UID: 116773232

Sayani Roy UID: 116818766

Prateek Bhargava UID: 116947992

Date: 25 February 2020

1. Introduction

This project focuses on the applications of Augmented Reality where single or multiple custom built tags are detected in a frame of video. The project is divided into two phases. The first phase involves the detection and tracking of the tag using contour and corner detection. The second phase involves the image processing where the detected tag is first replaced or superimposed by a provided template image (Lena.png) and then a virtual 3D cube is placed over the tag.

2. Phase 1

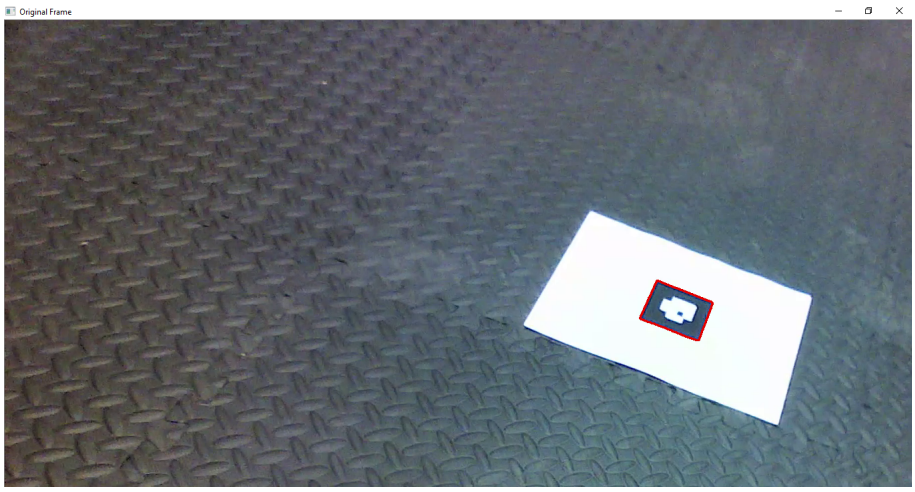
We started our implementation by importing our video and detecting it frame by frame in our main.py file. Detection of the AR tag is performed by converting the frames of the video from BGR to GRAY color model and defining a threshold which helps in the identification of the edges or other important features.

We identify the tags with the help of the OpenCV function - findcontours(), which stores the (x,y) co-ordinates of the boundaries of the AR Tag. The outermost contours from the required contour is filtered out by using a for loop and the function contourArea(). These functions are performed to acquire the desired contour by checking if the contour has 4 edges and that the contour has an area between 2000 and 22500.

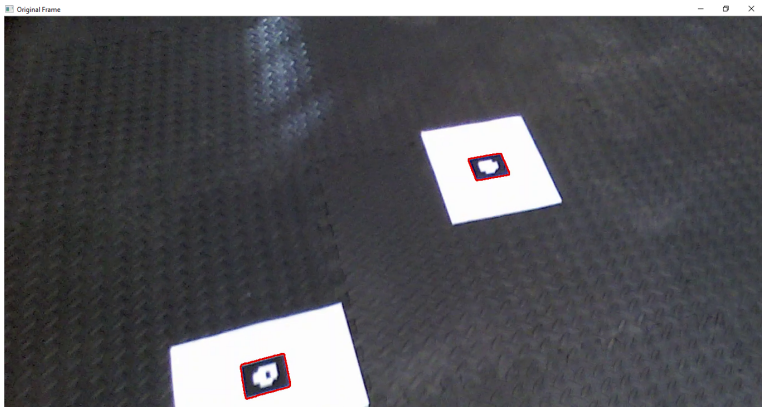
contours, _ = cv.findContours (vf_threshold, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)

The above find contours function has three arguments respectively:
1. Source image
2. Contour Retrieval mode
3. Contour approximation method - cv.CHAIN_APPROX_SIMPLE function removes all the redundant points while finding the contours thus saving memory while only removing the end points of the lines.

After detecting the desired contour using the cv.drawContours function we draw the contour displaying it using a red line. Then we find id of the tags to find the correct orientation of the AR tag. This is done by taking the world frame and checking where is the placement of the white box. We set it such that the ID is detecting the upright orientation with the white block in the bottom right corner.



Single Contour Detection



Multiple Contour Detection

3. Phase 2

3.1 Part a

After finding the four corners of the tag and their co-ordinates we find the homography between the corners of the template and the corners of the tags. The co-ordinates in the world frame are chosen to be as (0,0), (200,0), (200,200), (0,200).

To find Homography we first define the A matrix :

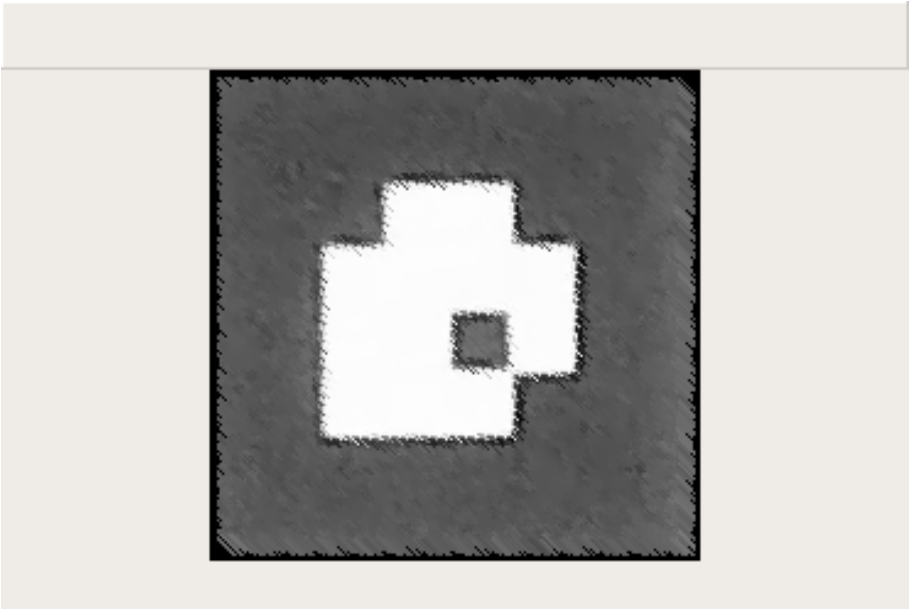
$$A = \begin{bmatrix} x0 & y0 & 1 & 0 & 0 & 0 & -xc0 * x0 & -xc0 * y0 & -xc0 \\ 0 & 0 & 0 & x0 & y0 & 1 & -yc0 * x0 & -yc0 * y0 & -yc0 \\ x1 & y1 & 1 & 0 & 0 & 0 & -xc1 * x1 & -xc1 * y1 & -xc1 \\ 0 & 0 & 0 & x1 & y1 & 1 & -yc1 * x1 & -yc1 * y1 & -yc1 \\ x2 & y2 & 1 & 0 & 0 & 0 & -xc2 * x2 & -xc2 * y2 & -xc2 \\ 0 & 0 & 0 & x2 & y2 & 1 & -yc2 * x2 & -yc2 * y2 & -yc2 \\ x3 & y3 & 1 & 0 & 0 & 0 & -xc3 * x3 & -xc3 * y3 & -xc3 \\ 0 & 0 & 0 & x3 & y3 & 1 & -yc3 * x3 & -yc3 * y3 & -yc3 \end{bmatrix}$$

Now, using Singular Value Decomposition we find U, S and V Matrices. SVD is expressed as:

$$A = USV^T \tag{1}$$

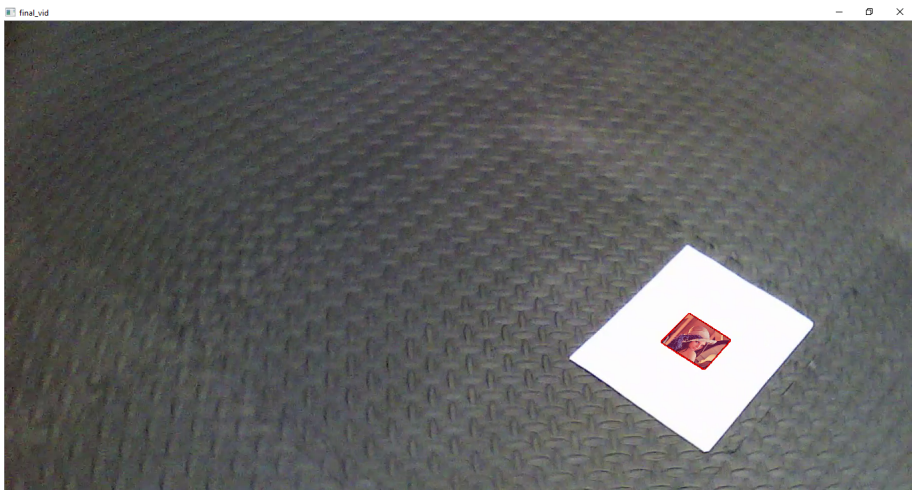
U is $m \times m$ orthogonal matrix, where each column forms the Eigen Vectors of the matrix AA^T
V^T is a $n \times n$ orthogonal matrix, where each column forms the transpose of Eigen vectors of the matrix A^TA.

Now to find the homography matrix we take the last column of the V matrix and then reshaping it to form a 3x3 matrix. Once the homography matrix is found we then warp the contours from the image plane to the world frame. This can be seen in the figure below.



Tag 2 warped on image frame

Now, using the detected tags found in the first phase of the project we superimpose Lena's image onto the AR tag with the correct orientation. We once again find the homography between the corner points of Lena and the corners of the tags produced. Lena is then projected on the world frame with the order of its points reversed. This then gives the image of Lena in the world frame.(seen in figure)



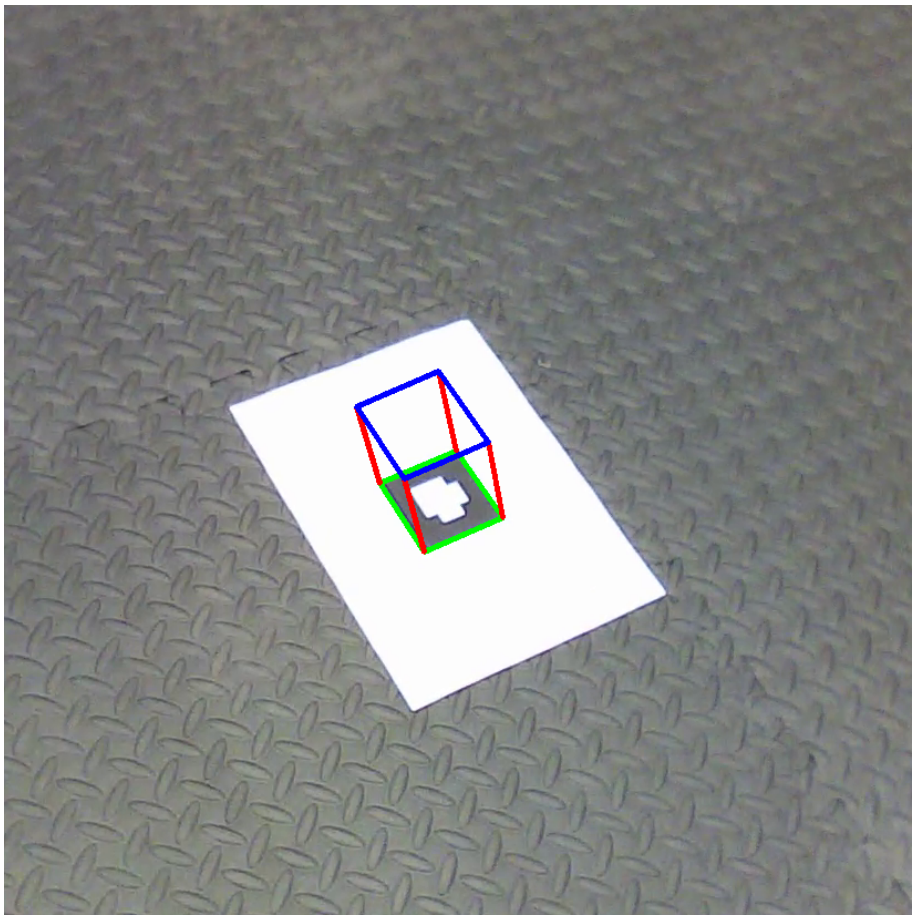
Lena Superimposed on a single Tag

3.2 Part b

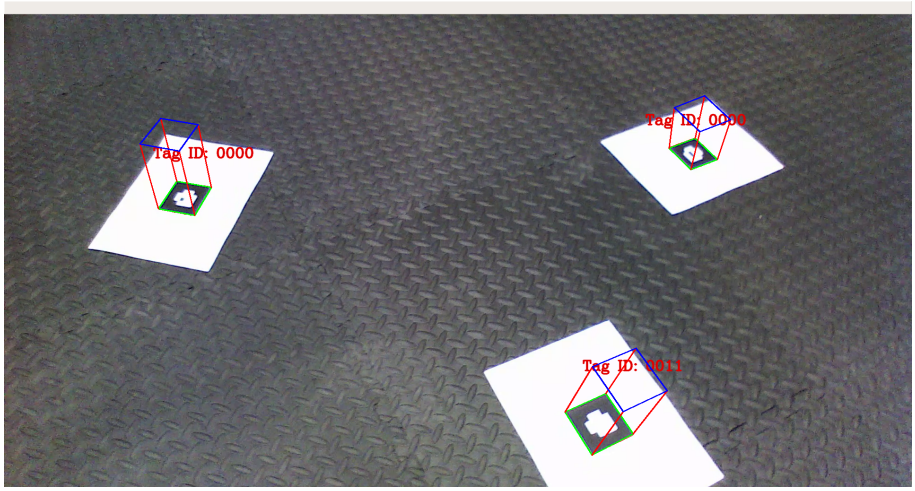
In this part of phase 2, we have to project a 3D cube on the tracked AR Tag from the video. We use the homography matrix derived from previous phase and the given camera calibration matrix to calculate the rotational and translational components of the projection matrix. The transposed calibration matrix was defined as:

$$K = \begin{bmatrix} 1406.08415449821 & 0 & 0 \\ 2.20679787308599 & 1417.99930662800 & 0 \\ 1014.13643417416 & 566.347754321696 & 1 \end{bmatrix} \quad (2)$$

The values of these rotational and translational components along with a 3D axis are then used to detect the coordinates of the tag in world frame by using the function projectPoints(). The cube is now to be placed at the generated coordinates. Since, we are projecting a 3D shape on a 2D image, we assumed the z axis to be negative in the upwards direction and calculated the other four corner coordinates of the cube using projection matrix. The cube is finally made by drawing simple lines using the functions line() and drawContours().



Cube placement on single tag



Cube placement on Multiple Tags

4. Problems encountered

The main problem with the code was faced with the 'multipleTags' video. Due to blurry images, detection of the tags could not be done continuously. As the frames in the video came to better focus, the tags were detected easily. Problems were also faced while superimposing and projecting the cube over the detected tag. The cubes were not being positioned properly on the tag. Different values for the 3D axis matrix were tried and tested on all the videos. The value used in he final code gave the most successful result.

5. Conclusion

Implementing Homography and Warp Perspective functions from scratch helped us gain a better understanding towards their functions. This can be specifically be seen in the application to transform each pixel and mapping it using forward and inverse warping.

Furthermore, the corner and edge detection techniques required us to perform several iterations to get an optimised result, helping us learn the concepts of contours and and detection algorithms.

6. Links

This section includes links to the google drive where the video outputs are stored and the GitHub repository. Please note that each video contains output for all three parts of the project, i.e., tag detection, image superimposition, and draw 3D cube on the tag.

- Link to folder containing all the output videos: `https://drive.google.com/drive/folders/1fPg8qZ5UhrjwJsX3OKZ2b5rX3MXvWC4G?usp=sharing`

- GitHub repository link : `https://github.com/urastogi885/ar-tag-detection`

7. References

- Learn how to use VideoWriter
- Understanding logic behind warp-perspective
- OpenCV Documentation
- Understanding logic behind find-homography