ENPM673: Perception for Autonomous Robots

Project 1: AR Tag Detection

Pruthvi Sanghavi UID: 116951005
Naman Gupta UID: 116949330
Amoghavarsha Prasanna UID: 116952910

Submitted **Date:**26 February 2020

Contents

1. AR Tag Detection:

hyperref The first task of the project is to detect the custom AR tag that has been provided. This AR tag encodes both the orientation of the tag. The encoding scheme is as follows:

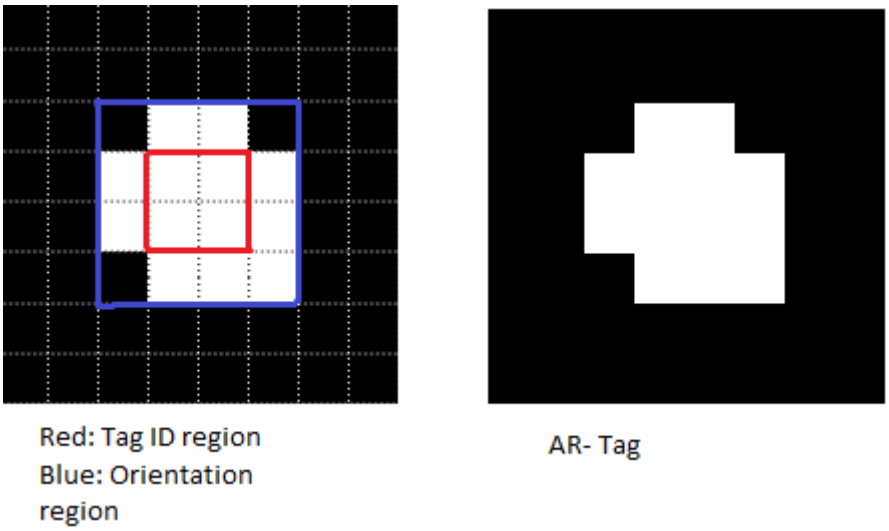

Figure 1: AR tag

The tag can be decomposed into an 8×8 grid of squares, which includes a padding of 2 squares width (outer black cells in the image) along the borders. This allows easy detection of the tag when placed on any contrasting background.
The inner 4×4 grid as seen in blue in figure 1(i.e. after removing the padding) has the orientation depicted by a white square in the lower-right corner. This represents the upright position of the tag.
Finally, the inner-most 2×2 grid, the red region in figure 1 (i.e. after removing the padding and the orientation grids) encodes the binary representation of the tag's ID, which is ordered in the clockwise direction from least significant bit to most significant. So, the top-left square is the least significant bit, and the bottom-left square is the most significant bit.

1.1 Implementation of Detection:

The detection of the tag needs to be done from the given videos, the opencv function cv2.videoCapture() is used to read the video frame by frame. The next step is to detect the edges, the edge detection algorithm cv2.canny is used, however, before the canny algorithm is implemented the frames are smoothened using a Gaussian blurring filter, since the canny function is sensitive to Gaussian noise.

Next, to find the contours of the Tag in the video, the opencv function cv2.findcontours(cnny_img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE) is used, this returns the values of all the contours that are detected in the video. The cv2.RETR_TREE basically arranges the points in a particular hierarchy of parent and child contours. The contours are streamlined and to make sure that internal rectangle is detected. The child contours with more than 4 points in the are considered and this further streamlined by using the cv2,approxpolyDP function to remove contours with more than 4 points in the contours that don't form a rectangle.[1]
Finally, all the smaller rectangle contours are stored in a list, that is used to draw the contours using cv2.drawcontours function. Please refer to the code attached for the complete implementation.

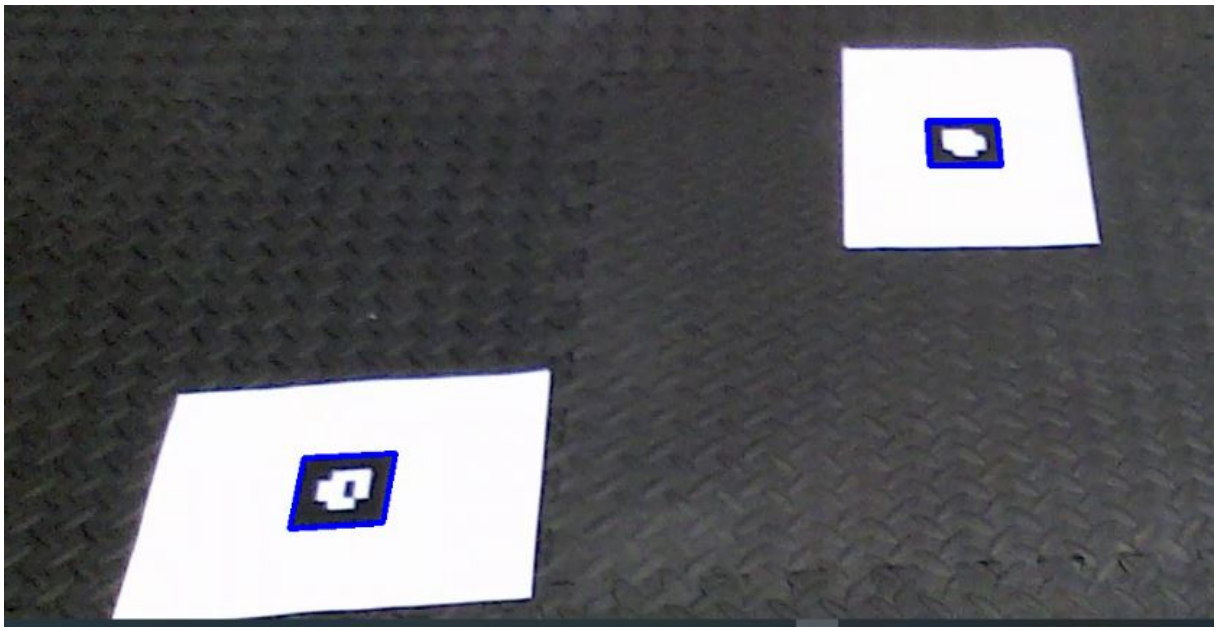

Figure 2: Contours detected from the video

1.2 Homography:

Homography is basically the transformation between two frames. The homography matrix plays a major role in establishing relationship between the reference AR Tag and the the image formed on the image plane.
consider a system of equations of the form $Ah = 0$, the general representation can be shown as follows:

$$\begin{bmatrix} {x_1}^{(w)} & {y_1}^{(w)} & 1 & 0 & 0 & 0 & -{x_1}^{(c)}{x_1}^{(w)} & -{x_1}^{(c)}{y_1}^{(w)} & -{x_1}^{(c)} \\ 0 & 0 & 0 & {x_1}^{(w)} & {y_1}^{(w)} & 1 & -{y_1}^{(c)}{x_1}^{(w)} & -{y_1}^{(c)}{y_1}^{(w)} & -{y_1}^{(c)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ {x_4}^{(w)} & {y_4}^{(w)} & 1 & 0 & 0 & 0 & -{x_4}^{(c)}{x_4}^{(w)} & -{x_4}^{(c)}{y_4}^{(w)} & -{x_4}^{(c)} \\ 0 & 0 & 0 & {x_4}^{(w)} & {y_4}^{(w)} & 1 & -{y_4}^{(c)}{x_4}^{(w)} & -{y_4}^{(c)}{y_4}^{(w)} & -{y_4}^{(c)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 3: general form

here, ${x_i}^{(w)}$,${y_i}^{(w)}$ and ${x_i}^{(c)}$,${y_i}^{(c)}$ are the points in the world co-ordinate system and the image co-ordinate system respectively, between which homography needs to be applied.

Since the A matrix is not symmetric, the solution needs to be found using least squares estimation. In this implementation, the builtin SVD function has been used. The homography matrix has been obtained as the last column of the Eigen vector matrix V obtained from SVD.[1]

1.3 Tag ID:

Now, that the Homography has been found, the Tag-ID needs to determined for further implementation.

The entire AR-tag has dimensions of 200×200 pixels. The blue region as seen on figure 1, there are certain regions where the pixel value is block(0) and the rest of the pixels are white(255), by mapping with respect to the these pixels we can determine the orientation. Please refer the attached code

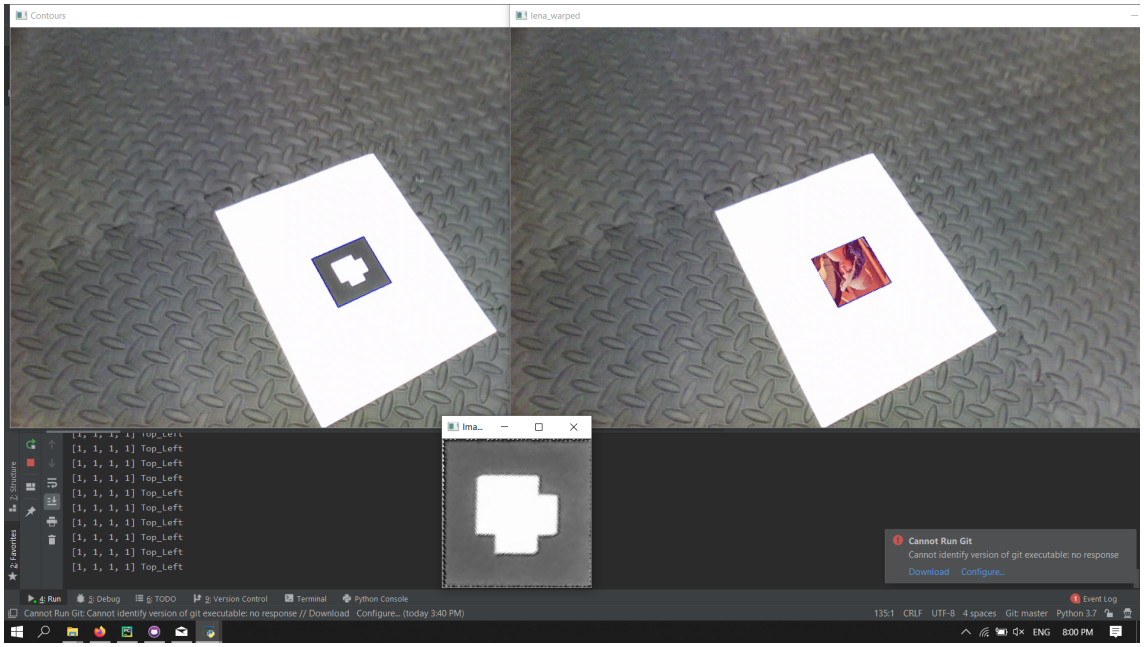

Figure 4: Tag-ID Orientation and Image Superimposition

2. Image Superimposition:

Once the tag ID and the contours are detected the given Lena image needs to be superimposed onto the tag in the video.

Essentially this procedure is overlapping a 2D image on to the Tag that has been detected on the video. This operation needs to be performed without built-in functions. First, the Lena image is resized to 200×200 pixels.

The Tag-ID function is used to determine the orientations of the AR tags on which superimposition should occur.Then,the superimposition in essence is carried out by mapping each pixel value from the 2-D Lena image onto the AR-tag, by multiplying the inverse homography matrix with each pixel value.
In addition, this implementation warps the Lena image on to the destination quite effectively. As seen in figure 4.

3. 3D Cube Superimposition:

3D Virtual objects can also be superimposed on the AR Tag using computer vision techniques. The information necessary to superimpose the cube on the AR Tag can be obtained using linear algebraic calculation. Mathematically, the coordinates of the cube are determined in the world frame which are in fact vectors. These vectors are multiplied by the transformation matrix to get the coordinates of the point on the tag to plot it on the tag. The transformation matrix can be obtained by the multiplication of the homography matrix and camera matrix. The camera matrix consists of the intrinsic parameters of the camera.

The first and the second columns of the matrix which is obtained after the multiplication represents the first two columns of the rotation matrix. We calculate R_3 column of the rotation matrix through the cross product of R_1 and R_2. The last column calculated from the afore-mentioned product is the required translation matrix. A scaling factor λ is calculated which is used for scaling the transformation matrix. The obtained data is used calculate the projected point coordinates which are then fed into a function which is then used to draw the cube on the tag. The projected point coordinates are calculated through the cv2.projectPoints() function and cv2.drawContours() and cv2.line() functions are used to draw the separate faces and then the pillars between the two faces respectively.

Drive Link for results: `https://drive.google.com/drive/u/2/folders/1bBNtIiOcpX6tTNRHlQWdCpyqdHI`

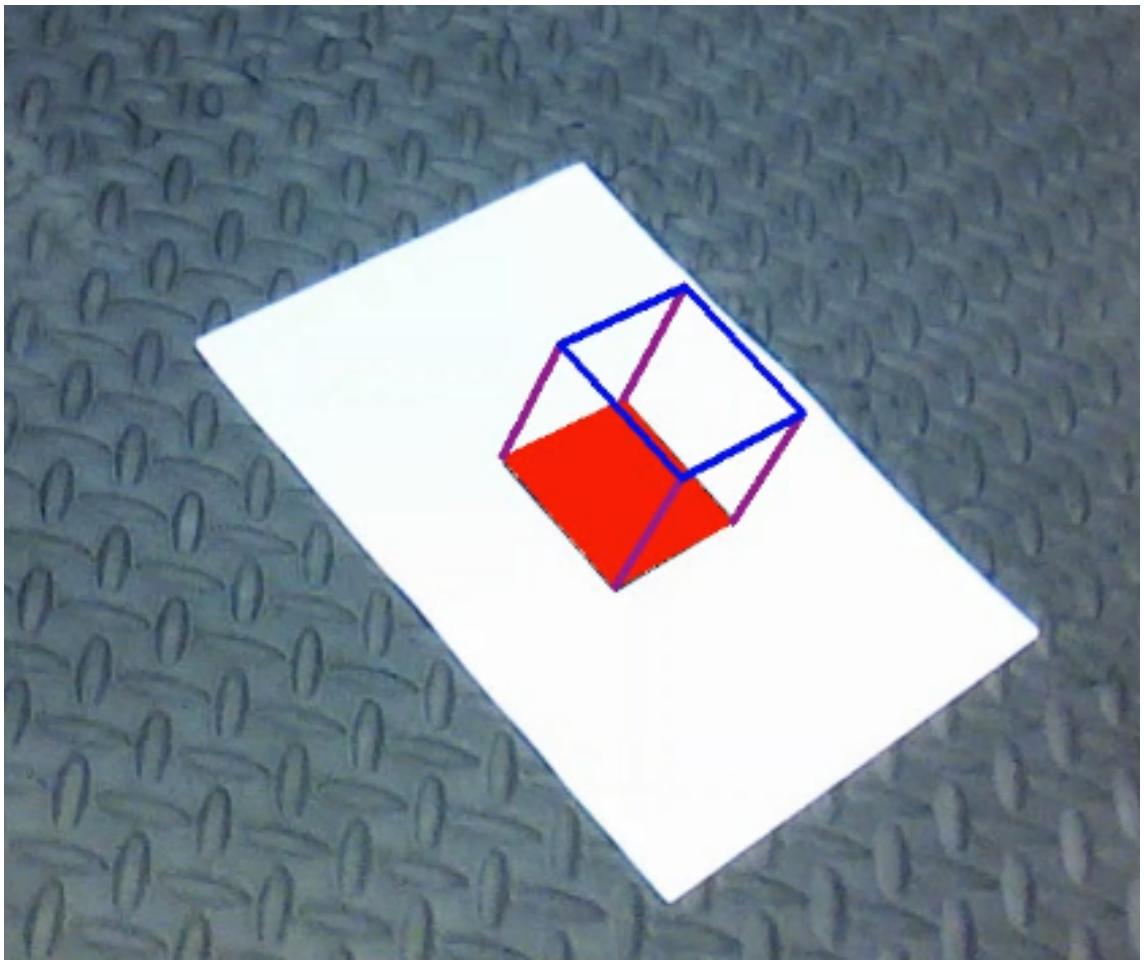

Figure 5: 3D Cube on AR Tag.

4. Bibliography

References

[1] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.