ENPM673 Project 1

Kumar Sambhav Mahipal, Raghav Agarwal and Vasista Ayyagari

February 2020

1 AR Tag Detection and Tracking

Please click here to redirect to the google drive with the output videos. This project focuses on detecting a custom AR Tag (a form of fiducial marker), that is used to obtain a point of reference in the real world, such as in Augmented Reality (AR) applications. The two aspects of AR Tags are detection and tracking, and been implemented in this project. **Detection:** Involves detecting the AR Tag from a given video data set.

Tracking: Involves keeping the tag in "view" throughout the sequence and performing image processing operations such as superimposing an image over the tag based on the tag's orientation and position.

What is an AR Tag and why it is used?
An AR Tag is a fiducial marker system to support AR. They can be used to facilitate the appearance of virtual objects, games, and animations within the real world. Once the camera's position is known, a virtual camera can be positioned at the same point, revealing the virtual object at the location of the AR Tag. Therefore, it addresses two of the key problems in Augmented Reality: viewpoint tracking and virtual object interaction.

2 Problem 1-Edge Detection of AR Tag

2.1 Tag Detection

Firstly, we apply the Gaussian blur to the frames, convert it into gray scale and then to a binary image with a threshold for better accuracy. We use the findContour function in opencv to find all the contours in the binary image. The contours outside a given range of area are filtered out. Then the approxPolyDP function is used to find the corners of the filtered contours. The contours with exactly 4 corners correspond to the corners of the tag. We further, find the homography with respect to the world frame. We warp the AR tag to the world

frame. We re-orient the tag in the world frame to the appropriate orientation by first resizing it to an 8×8 image, finding its orientation and then rotating the image. Then we check the innermost 4 squares to get the ID.
Thus, this implementation identifies and detects the AR tag in any given image sequence.

2.1.1 Homography Transformation

The coordinates of the tag in the world frame are defined as [(0,0), (200,0), (200,200), (0,200)]. Using the coordinates and the corners of the detected tag, we calculate the Homography matrix between the world frame and the camera frame.

Decomposing the Matrix A using the Singular Value Decomposition Method:

$$[U, S, V] = svd(A) \tag{1}$$

The last column of the eigen vector matrix V are the elements of the Homography Matrix H and because we require that $h_{33} = 1$, we normalize with the value of V_{99}

$$H = \frac{[V_{19}, ...V_{99}]}{V_{99}} \tag{2}$$

Then we reshape the elements of H into a 3×3 matrix using the function reshape, i.e., $H = H.reshape(3, 3)$

3 Problem 2 Tracking

3.1 Superimposing an image onto tag

Homographic transformation is the transformation of an image between two planes in projective space. In Augmented Reality (AR), it is the transformation that projects a square marker board from the world coordinate system into a quadrilateral in the image plane on the camera sensor (i.e., pixel coordinates). In this project, once the 4 corners of the tag are obtained with the edge and corner detection(as done above), homography estimation is performed. Thus, using homography we can superimpose Lena.png over the AR Tag.

3.1.1 Implementation Details

As in Problem 1, The homography matrix between the world frame and camera frame is computed. Initially, we project the AR tag in the world frame and find out the angle by which it needs to be rotated to get the upright orientation. Then we rotate Lena in the opposite direction by the same angle and use the Homography Matrix's inverse to project the rotated lena into the camera frame

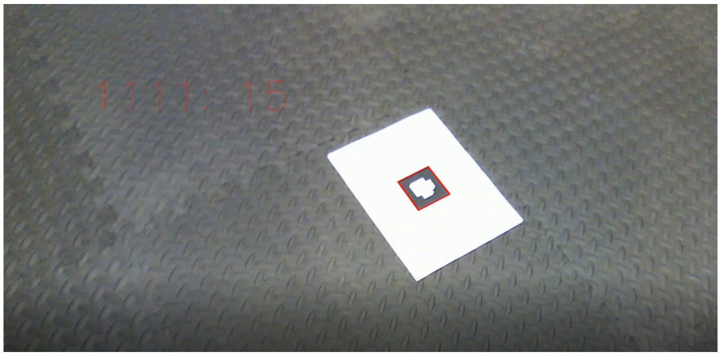

Figure 1: Screenshot of the Tag 0

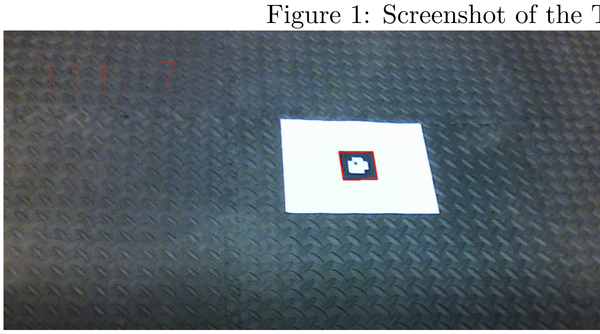

Figure 2: Screenshot of the Tag 1

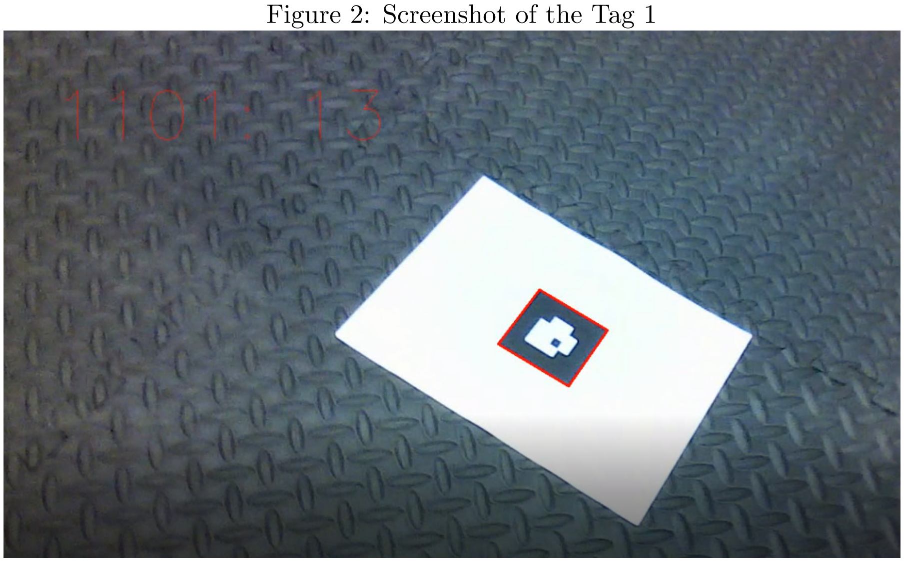

Figure 3: Screenshot of the Tag 2

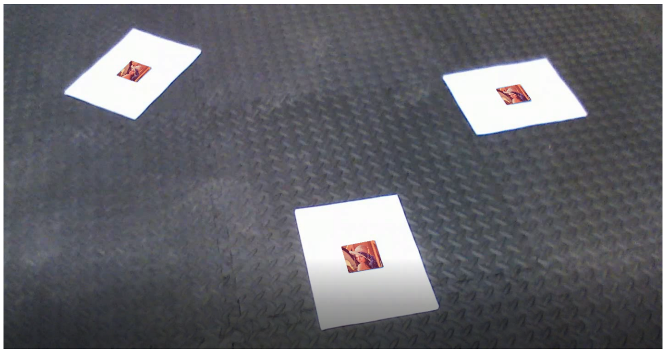

Figure 4: Screenshot of the Lena Image

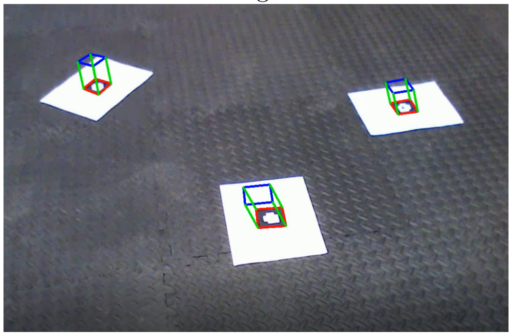

Figure 5: Screenshot of the Cube

to get the augmented image onto the AR Tag.
Figure 4 shows Lena augmented onto the AR tags. The images show that the superimposed image maintains its orientation, thus compensating for the rotation of the camera.

3.2 Placing a virtual cube on the tag

3.2.1 Implementation Details

From the supplementary material, we learn the procedure of computing the projection matrix P from camera intrinsic matrix K and the Homography Matrix H. We compute the projection matrix. We define a cube in the world frame using the following cordinates [(0,0,0),(200,0,0),(0,200,0),(200,200,0),(0,0,-200),(200,0,-200),(0,200,-200),(200,200,-200)] for the corners of the cube. We then convert these the corners to homogenous coordinates by concatenating $w = 1$ to each corner cordinate. Then using the projection matrix, we compute the cordinates of the corners of the cube in the camera frame.

$$X_c = PX_w \tag{3}$$

where $X_w = [x_w, y_w, z_w, 1]^T$, $X_c = [x_c, y_c, 1]^T$ Finally, to draw the cube, we used the functions in opencv to draw lines and contours. Figure 5 shows a cube augmented onto the AR tags.

3.3 Challenges faced during implementation

3.3.1 Prepossessing

We initially used Canny edge detection before computing the contours. This gave inaccurate results. When we switched to the threshold function in opencv, we obtained better results

3.3.2 Corner Detection

We used 2 different techniques to compute the corners of the AR tag. First was to filter the contours based on area and the number of corners. The second was computing the ratio of the area of each contour with the area of its convex hull and sorting the contours based on this ratio. The minimum ratio will correspond to the white outline in the AR Tag that gives us the orientation and ID. Its parent from the hierarchy tree will give the square corners. In practice, filtering by area gave more robust and accurate results.

3.3.3 Jittering

There was constant jittering in the video during the augmentation of Lena and the cube.