ENPM 673 Project 1

Arjun Srinivasan Ambalam, Praveen Menaka Sekar, Arun Kumar Dhandayuthabani

February 27, 2020

1 Introduction

The project focuses on detecting a custom AR Tag (a form of fiducial marker), that is used for obtaining a point of reference in the real world, such as in augmented reality applications. The two aspects to using an AR Tag: detection and tracking, has been implemented in this project. Following are the 2 stages:

- Detection: Involves finding the AR Tag from a given frame in the video sequence
- Tracking: Involves keeping the tag in "view" throughout the sequence and performing image processing operations based on the tag's orientation and position (a.k.a. the pose).

After detection and tracking of AR tags, we perform two tasks, namely superimposing Lena image, and placing a virtual cube over the AR tag.

2 Problem 1

You are given a custom AR Tag image, as shown in Fig.1, to be used as reference. This tag encodes both the orientation as well as the ID of the tag. You are free to develop any detection pipeline, as long as the encoding scheme specified is followed. This part of your code should return the corners of the tag as well as its ID with respect to its original orientation (i.e. compensated for any rotation you might encounter). You may use any corner detector algorithm implemented in OpenCV (such as Harris or Shi-Tomasi).

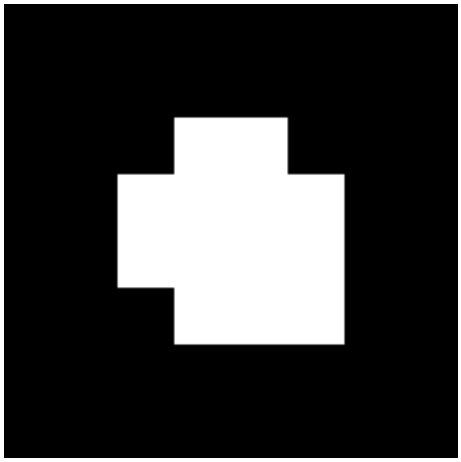

Figure 1: Reference Marker

2.1 Detection of AR tags

1. The given test videos Tag0.mp4, Tag1.mp4, Tag2.mp4 and multipleTags.mp4 are read into frames using cv2.VideoCapture.
2. Convert the frames into gray scale.
3. Apply a Gaussian or median or bilateral filter to smoothen the image and remove noise.
4. Threshold the resulting frame with a threshold value of 127 to obtain binary image.
5. Find contours of the binary image using cv2.findContours function.
6. Remove the unwanted contours using hierarchy.
7. Sort the remaining contours based on area bounded by it and choose the largest 3 or 4.
8. Approximate the obtained contours using cv2.convexHull, cv2.arcLength, and cv2.approxPolyDP.
9. If the approximated contours length is that of a 4 sided polygon, extract them as corners of the tag.
10. To arrange corners of an AR tag found, so as to use the corresponding world frame point we use a geometrical approach,
    (a) The x coordinates of the corners are sorted.
    (b) The first 2 coordinates correspond to the left most corners whereas the remaining 2 correspond to right most corners.
    (c) We again sort the left most coordinates to obtain the top left and bottom left corners.
    (d) We then calculate the Euclidean distance between the right most corners and top left corner.
    (e) The corner corresponding to largest distance is bottom right corner whereas the left out is top right corner.
11. Thus we get only one set of corners in a correct order for each AR-Tag present in the video, and hence we have successfully detected the AR-Tags from the video.

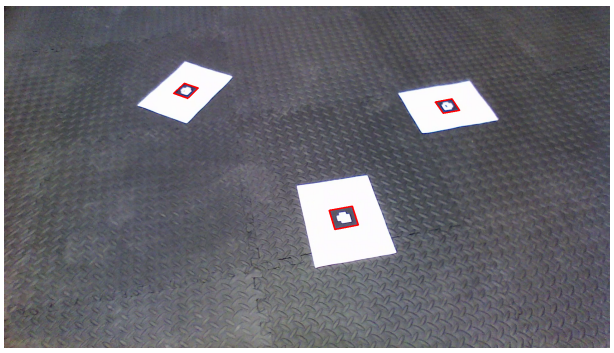

Figure 2: Tag Detection

2.2 AR Tag ID decoding

AR Tags facilitate the appearance of virtual objects, games, and animations within the real world. The analysis of these tags can be done as followed. The tag has been decomposed into an 8x8 grid of squares, which includes a padding of 2 squares width along the borders. This allows easy detection of the tag when placed on white background. The inner 4x4 grid (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. This

is different for each of the tags provided in the different image sequences. Lastly, the inner-most 2x2 grid (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.

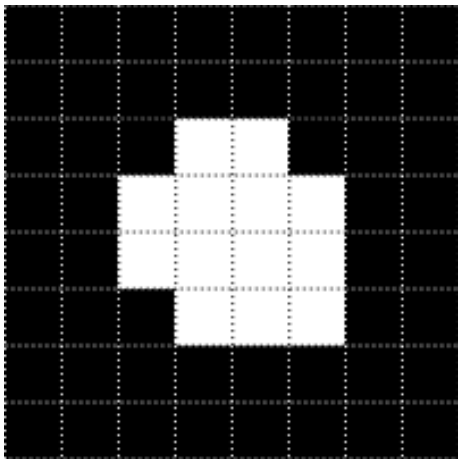

Figure 3: Reference Marker Grid

1. Estimate homography matrix (H) between the reference AR tag image (camera coordinate) and detected AR tag (World coordinates).
2. Warp the detected AR tag using estimated homography to obtain perspective image.
3. The warped image is then divided into 64 squares (downsized to a 8x8 image).
4. The value of each square is decided by finding its mean value;if mean ¿ 127 then the square corresponds to white pixel (1) else it corresponds to black pixel (0).
5. Once we have the 8x8 matrix, we try to find the position of the white box present in the corner of the 4x4 matrix, which will give us the orientation of the tag.
6. Once we have the orientation, find the value of tag ID through the 2x2 matrix in between.

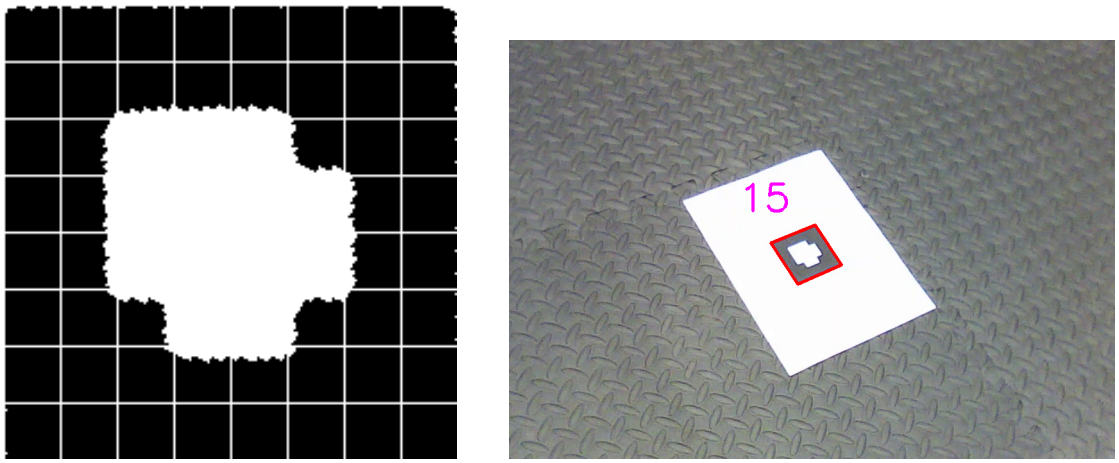

Figure 4: Perspective Image and Tag ID detection of Tag 0

3 Problem 2

Superimposing an image onto the tag

Once you have the four corners of the tag, you can perform homography estimation on this in order to perform some image processing operations, such as superimposing an image over the tag. The image you will use is the Lena.png file provided. Let us call this the template image.

1. The first step is to compute the homography between the corners of the template and the four corners of the tag.
2. You will then transform the template image onto the tag, such that the tag is "replaced" with the template.
3. It is implied that the orientation of the transferred template image must match that of the tag at any given frame.

Figure 5: Lena Image

Placing a virtual cube on the tag

Augmented reality applications generally place 3D objects onto the real world, which maintain the three dimensional properties such as rotation and scaling as you move around the "object" with your camera. In this part of the project you will implement a simple version of the same, by placing a 3D cube on the tag. This is the process of "projecting" a 3D shape onto a 2D image. The "cube" is a simple structure made up of 8 points and lines joining them. There is no need for a complex shape with planes and shading.

1. You will first compute the homography between the world coordinates(the reference AR tag) and the image plane (the tag in the image sequence).
2. You will then build the projection matrix from the camera calibration matrix provided and the homography matrix.
3. Assuming that the virtual cube is sitting on the top of the marker, and that the Z axis is negative in the upwards direction, you will be able to obtain the coordinates of the other four corners of the cube.
4. This allows you to now transform all the corners of the cube onto the image plane using the projection matrix.

3.1 Superimposing an image onto the tag

1. The corners of the AR tag are updated according to its orientation.
2. The image of Lena is to be superimposed onto the AR tag
3. We then estimate the homography matrix (H) between the given template, image of Lena and corners of the AR tags.
4. Warp the template Lena image onto the frame using the estimated homography matrix.
5. The orientation of the template Lena image matches to that of the AR tag which is clearly observed in the video since there is no flip or disorientation of the image with the changing orientation of the camera.

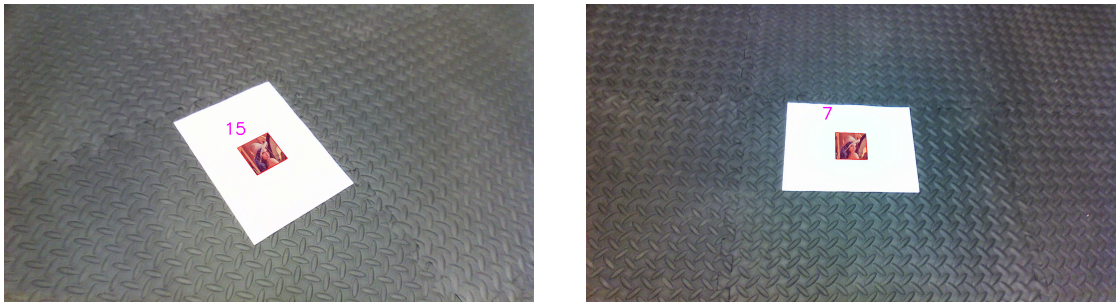

Figure 6: Superimposing Lena Image onto AR tags for Tag 0 and Tag 1

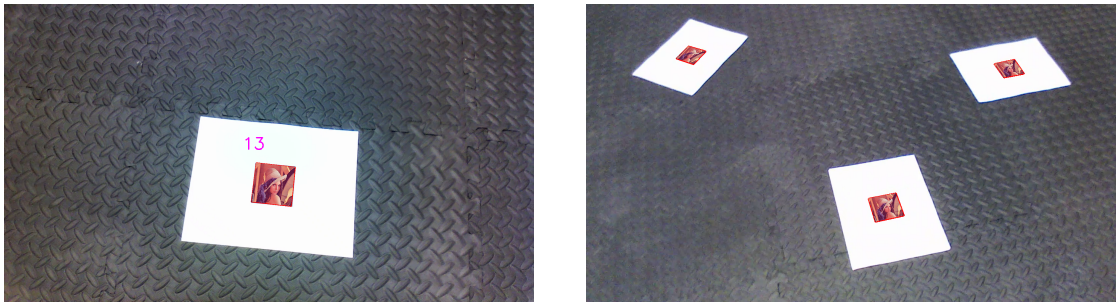

Figure 7: Superimposing Lena Image onto AR tags for Tag 2 and multiple tags

3.2 Placing a virtual cube on the tag

1. Estimate the homography matrix (H) between the corners of the reference AR tag and corners of the detected AR tag.
2. Estimate projection matrix (P) using the estimated homography matrix (H) and camera calibration matrix (K).

$$\lambda H = K\tilde{B} \tag{1}$$

$$\tilde{B} = \lambda K^{-1} H \tag{2}$$

$$\lambda = \left(\frac{\| K^{-1}h_1 \| + \| K^{-1}h_2 \|}{2}\right)^{-1} \tag{3}$$

$$r_1 = \lambda b_1,\ r_2 = \lambda b_2,\ r_3 = r_1 \times r_2,\ t = \lambda b_3 \tag{4}$$

$$P = K[R \mid T] \tag{5}$$

3. Assuming that the virtual cube is sitting on "top" of the marker, and that the Z axis is negative in the upwards direction, we have obtained the coordinates of the other four corners of the cube. This allows the transformation of all the corners of the cube onto the image plane using the projection matrix.

4. The cube is then drawn using cv2.line function.

Figure 8: Placing 3D virtual cube onto AR tags for test videos

4 Problems Encountered

1. For the corner detection of the AR tag first we were trying to use Harris corner and goodfeaturestotrack algorithm but the problem was that the corner points output which these were returning are randomly organized so it was difficult to separate and say to which contours they belong, as there can be contour of the white paper and other unwanted detections are possible. So we used cv2.findContours from which the unwanted contour can be discarded using the hierarchy of contours which this function returns

2. For implementing warping we initially used two for loops which were computationally intensive and time consuming, so we had to come up with the vectorized implementation of warping function