ENPM673 - Perception for Autonomous Robots

PROJECT 1

AR Tag tracking, Detection, Superimposition, and 3D Projection

Authors:

Name:	Karan Sutradhar	Markose Jacob	Trevian Jenkins	Girish Ethirajan
UID:	117037272	117000269	116781381	116408493

Date: February 25th, 2020

Objective:

Our objective is to develop a detection algorithm that returns the encoded ID of an AR tag. After this task, we seek to superimpose an image over the tag, followed by placing a three-dimensional cube on the tag.

Introduction:

This project mainly focuses on detecting a custom AR Tag, which is widely used in augmented reality applications. AR tags are particularly useful for accomplishing two major tasks: detection and tracking. The detection stage involves finding the AR Tag from a given image sequence, while the tracking stage involves keeping the tag in view throughout the sequence and processing images based on the tag's orientation and position. In this project, we implement tracking and detection using a corner detection algorithm, followed by homography calculation and warping the image. Then we place an image (lena.png) on the AR tag, as well as a three-dimensional cube.

Problem 1:

Tracking:

Upon executing the code, the user is prompted for the video to be played. Once the video and image file are available, we convert the image to grayscale. Each pixel now has three key characteristics: X coordinate, Y coordinate, and intensity.

Corner Detection:

We first convert the video into frames for individual processing. For better detection of contours, we convert each frame into grayscale and apply a binary threshold.

Since several contours including the AR tag may be detected, we filter out invalid contours based off a few criteria. Simplifying each contour to its polygonal corners, we omit any detections of contours that do not have exactly four vertices.

We also expect tags to be of substantial quality for ID detection, so any contours with an area below 600 pixels are also omitted. This rule also helps filter out small artifacts, such as the light-colored objects in the back of the room.

Homography:

To compute the homography matrix, we need both the coordinates of the AR tag in the world and the coordinates of the reference AR tag image. The coordinates of the reference AR tag image are known from its size, so we iterate among the contours, calculating the homography of each based off the coordinates of its corners.

We can compute homography using the matrix A below.

$$A = \begin{bmatrix} xw1 & yw1 & 1 & 0 & 0 & 0 & -xc1xw1 & -xc1yw1 & -xc1 \\ 0 & 0 & 0 & xw1 & yw1 & 1 & -yc1xw1 & -yc1yw1 & -yc1 \\ xw2 & yw2 & 1 & 0 & 0 & 0 & -xc2xw2 & -xc2yw2 & -xc2 \\ 0 & 0 & 0 & xw2 & yw2 & 1 & -yc2xw2 & -yc2yw2 & -yc2 \\ xw3 & yw3 & 1 & 0 & 0 & 0 & -xc3xw3 & -xc3yw3 & -xc3 \\ 0 & 0 & 0 & xw3 & yw3 & 1 & -yc3xw3 & -yc3yw3 & -yc3 \\ xw4 & yw4 & 1 & 0 & 0 & 0 & -xc4xw4 & -xc4yw4 & -xc4 \\ 0 & 0 & 0 & xw4 & yw4 & 1 & -yc4xw4 & -yc4yw4 & -yc4 \end{bmatrix}$$

where (x1, y1), (x2, y2), (x3, y3), and (x4, y4) are the coordinates of each corner. All four corners are required for this type of transformation, which is known as a projective

transformation. In a projective transformation, neither the side lengths, nor angles, nor side ratios are preserved, but lines remain straight.

We compute the singular value decomposition (SVD) to find Vt. The elements of homography are the last row of the Vt matrix. We normalized the matrix by dividing the last value with each element of the matrix. After this step we reshape the matrix into a 3 x 3 form.

Fig 1: Detected AR tags. Left: tag ID 1248, reoriented 90 degrees clockwise. Middle: tag ID 1248, reoriented 90 degrees counter-clockwise. Left: tag ID 1248, reoriented 90 degrees counter-clockwise.

Unwarping:

To translate any pixel coordinate, we multiply the inverse of the homography matrix H with the input position vector [x y 1]^{T} to obtain a vector [u v w], where our new x-coordinate is 'u/w' and our new y-coordinate is 'v/w'. Running this calculation for every pixel in the image (or contour) gives us the new (un)warped image.

Finding Orientation and Tag ID:

To find the orientation, we use the unwarped AR tag image as an input. If the average intensity of the outside border of the unwarped 8 x 8 grid is near 100 %, we have a contour of the page on which the tag ID is printed, rather than the tag itself, so we discard this detection. Otherwise, we proceed to scan each corner one by one within the inner 4 x 4 grid. We calculate the average intensity of the four cells in each corner and then pick the cell with maximum intensity, where white denotes the maximum possible intensity value of 255. We reorient the AR tag so that the white cell is in the bottom-right corner of the image.

To get the tag ID we scan each cell inside the 2 x 2 grid. We calculate the average intensity in each of the four innermost cells. If the average intensity of a cell is greater than a certain threshold (specifically 216 / 255), then that cell is assigned a value of one, otherwise it is given a value of zero. The least significant bit is at the top left cell of the 2 x 2 grid, with

significance increasing in the clockwise direction, meaning the most significant bit is at the bottom left cell. We label the images with this ID.

Problem 2(a):

This section involves superimposing an image (lena.png) onto the AR tag. Here, we use the homography matrix between the image and the AR tag to warp Lena onto the tag. The upright orientation of the tag corresponds to the upright orientation of the image of Lena. Finally, a mask is applied to the AR tag to draw Lena in place, with multiple masks combined using bitwise-or in the presence of multiple tags (see Fig. 3).

Fig 2: Superimposed image of Lena (left) onto the original AR tag (right)

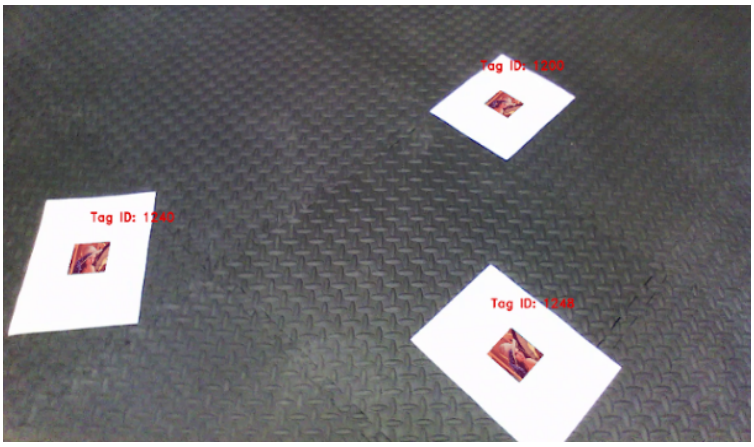

Fig 3: Lena superimposed on multiple AR tags

Problem 2(b):

This section describes displaying a three-dimensional cube onto the AR tag.

The next stage of this project is to superimpose a cube onto the tag. To accomplish this task, we define the cube coordinates in the world coordinate frame. Then we compute the transformation matrix which comprises the rotational and translational matrix. To calculate the transformation matrix, we first find the product of the homography matrix and camera matrix. The cross product of the first column and the second column of the aforementioned product is the required r3. We then compute the scaling factor 'lambda' which is used for scaling the transformation matrix. Then lambda is calculated as per the following equation:

$Eq : \lambda = ((||K^{-1}h_1|| + ||K^{-1}h_2||)/2)^{-1}$.

From this data, we feed the coordinates of the cube into a projection function to obtain the three-dimensional points. These coordinates are then used to draw a wireframe of the 3D figure (see Fig. 4 - 7).

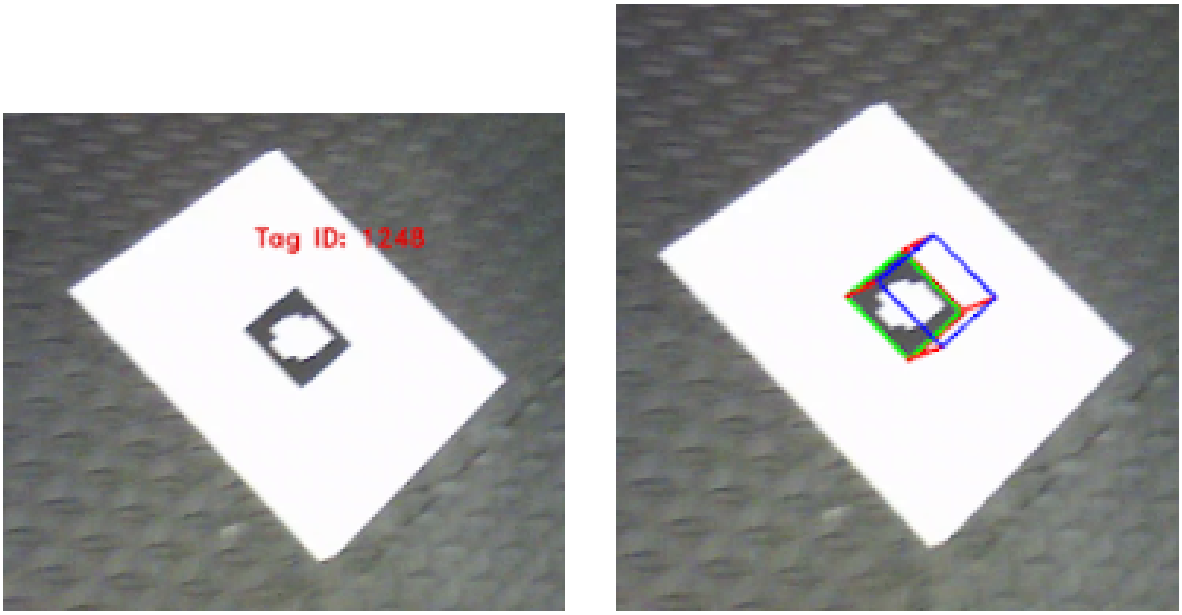

Fig 4: Original tag for video Tag 0 vs AR tag detected and cube projected

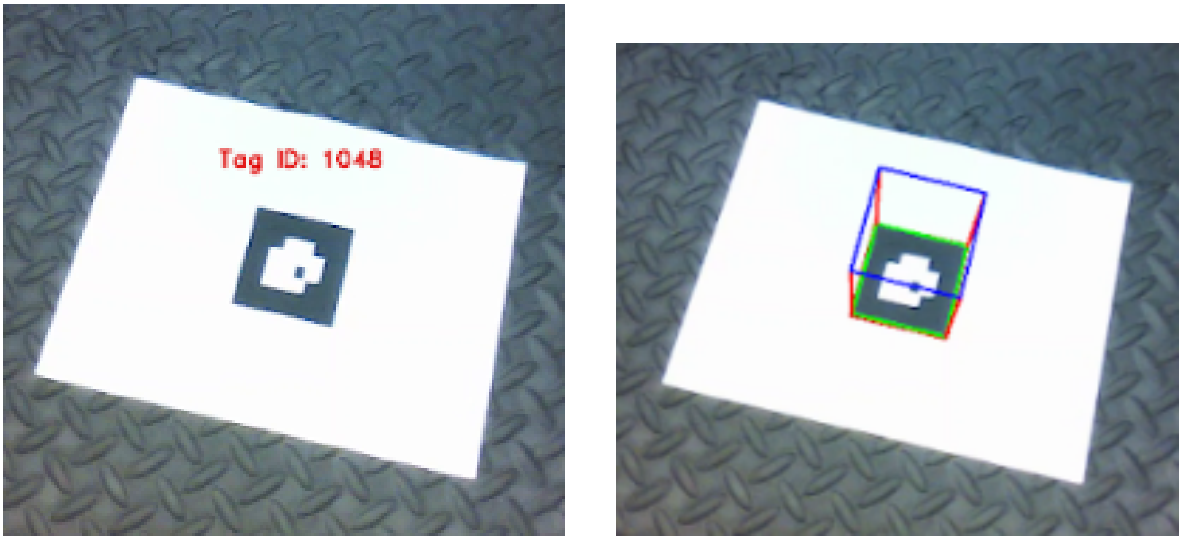

Fig 5: Original tag for video Tag 1 vs AR tag detected and cube projected

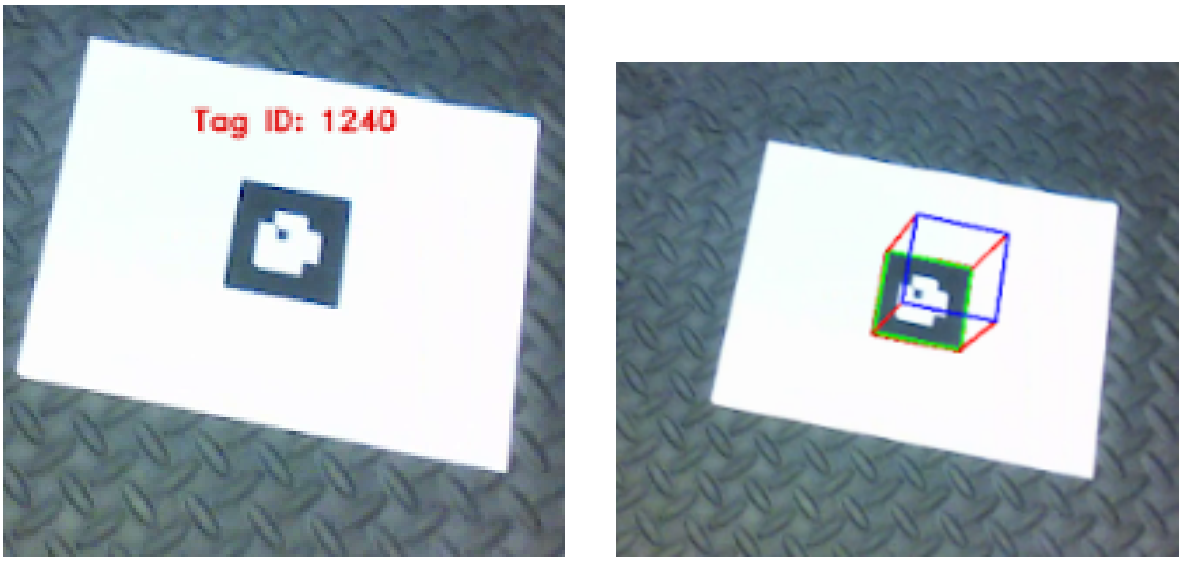

Fig 6: Original tag for video Tag 2 vs AR tag detected and cube projected

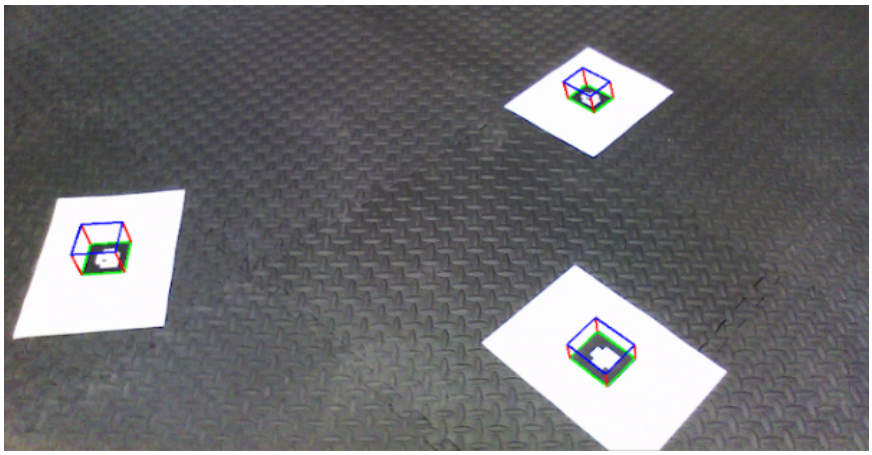

Fig 7: AR tag detected and cube projected for multiple tags

Problems Encountered:

1) Determining the orientation of the tag to get the ID was initially a tougher challenge. At first, our approach was to map the order of the significance of the inner cells with the orientation. This complicated the calculation of the ID because we mixed up a couple of the cells in our mappings. We eventually took a simpler approach of reorienting the unwarped tag so that we could use the same order for tag ID calculations every time.

2) Initially, our program did not work well for videos with multiple tags because Lena was only getting superimposed on one tag at a time. If a different tag was detected on the next iteration of the contour detection loop, then the previous superimposition image was overwritten. Combining the masks using bitwise-or solved this issue.

3) Sometimes Lena and the cubes would be drawn on the paper instead of the AR tag. At first, we attempted to mitigate this issue solely by checking that the detected contour was within a certain area. However, this prevented us from detecting some AR tags in some frames, so we decided on our approach of checking the border of the unwarped image. The tag images have a primarily black background, while the paper images have a white border. This distinction was helpful in keeping Lena and the cubes on the tag instead of superimposed on the whole paper.

4) Latency was an issue with our warp function. Each frame would take several seconds to render when we performed the warp using the whole input image. Each pixel from our unwarped tag image was multiplied by a homography matrix, so this operation was expensive for a large image. Eventually, we saved significant computational time by spacing out the pixels. Likewise in the other direction, the homography multiplication for a warped image only had to be performed on the contour, not the entire image of the world. In the future, we could decrease the resolution of the Lena image projected into the world to save even more processing time.

Video Output

The video output from our program can be found in the link below :
https://drive.google.com/drive/folders/19Blu8EAS0JE6sqs-EkyWMdnui6AJ861P

We recommend using VLC to play the videos.

NOTE:

To run the code, you need to follow the instructions provided in the README file.