Rohan
Abhinav
Kamakshi

Project 1

ENPM 673
February 28, 2019

Problem 1 Detecting AR Tag

Blob detection algorithm:

A Blob is a group of connected pixels in an image that share some common property. This algorithm is controlled by parameters such as Threshold, color, area etc.

We have used blob detection to detect the black area of the tag. The blob is detected by using two parameters only: the area of the blob and the color i.e., black. We set a threshold for the minimum and the maximum area of a black blob and find the center of this desired region. This is being used to crop the video frame to show a black square(the AR tag) on a white background.

Corner detection

Contours are simply curves joining all the continuous points (along the boundary), having same color or intensity. Here, in our code we have used the parameter CV_CHAIN_APPROX_SIMPLE in the **cv2.findContours** that compresses horizontal, vertical, and diagonal segments and leaves only their end points. However, we get a set of points along the boundary. To get 4 corners of the tag we apply **approxPolyDP** which selects the corners out of the multiple points generated. This gives us the corners of black region of our AR tag.

Homography

So far we have found p_2, which is source point and p_1 has been chosen randomly based on an arbitrary scale value. Now, to find Homography, we know that:

$$A * h = 0_{8x1}$$

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

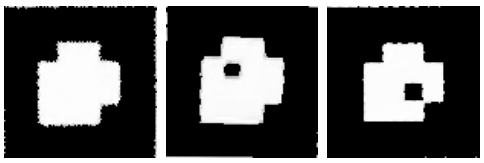

Figure 1: Detected AR Tags

Here, A is not a square matrix with size 8×9. We use singular value decomposition (SVD) to find the smallest eigen vector of A. Accordingly, A=UDV^T

The elements of homography, H, is the last column of the V matrix in the above formula. Thus,

$$h = \frac{[v_{19}, ..., v_{99}]^T}{v_{99}}$$

The H found now is reshaped into a 3x3 matrix. The inverse of this square matrix is then computed and normalised.

Warping

After finding homography from source coordinates to warped coordinates, we perform warping by multiplying the source coordinates to the obtained homography matrix. We observed that sometimes due to scaling of the 2 image coordinates, our warped image develops some blank pixels. This happened because after multiplication the obtained coordinates were rounded to the nearest integer and if the warped image size was larger than source image not all pixels were covered, leaving blank pixels.

We solved this problem by multiplying the warped image coordinates by the inverse of the homography matrix instead. This was done to find which coordinate of the warped image matches with which coordinate of the source image, and then using the pixel value from the source image and pasting it back on the warped image. This way we ensure that all the pixels of the warped image are covered, Thus pixel is left empty no empty.

Extracting information
After warping, we have finally detected our AR tag.Now, to extract information from our tag we first find the mean for the 8x8 area of the tag. Then we shift all the value below to the mean to 0 and above the mean to 1.

After removing the padding, inside the 4x4 grid the 1 value in lower right corner gives the orientation. Further inside the 2x2 grid the tag ID in binary representation is presented in the clockwise direction from least significant bit to most significant.

Problem 2 Superimposing image to the tag

The image given is that of lena as shown in the figure(2). Corners of image lena are in the world frame. We already have the tag corners from the above problem. Using the function created above we find the homography for these points. These points are then warped. Finally, a mask is applied to the tag to make each pixel 0 and then the mask is added with the chosen image.

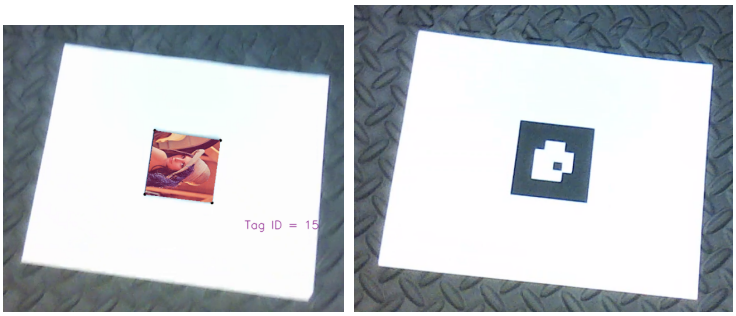

Figure 2: AR tag detected vs original AR tag for video Tag2

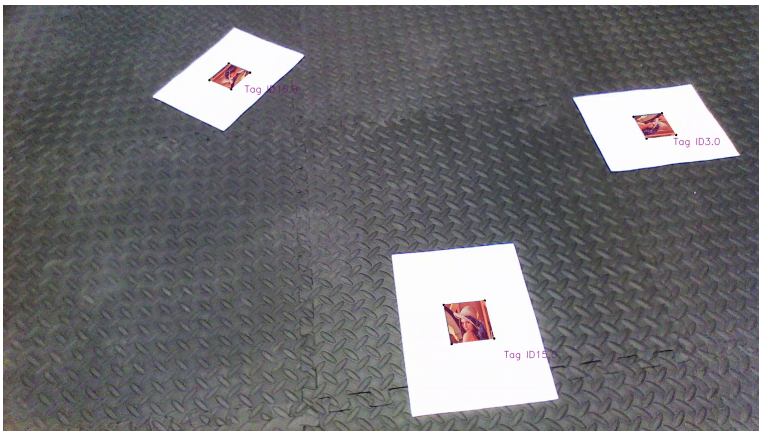

Figure 3: Lena superimposed on the AR tag for video multiTags

Problem 3 Creating Cube

In augmented reality,we wish to determine how the point $x^{(w)}$ projects into the point $x^{(c)} = [x^{(c)}, y^{(c)}, 1]^T$ lying in the cameras image plane. For this we use the Projection matrix.

$$x_c = Px^w$$

where P=K[R | T], and R is the rotation matrix and T is the translation vector. This Projection matrix is calculated using equations provided in the supplementary material[link]. Also, this projection matrix is an extended version of the homography, where H=K[r_1, r_2,t] whereas P =K[r_1, r_2,r_3,t] and we can estimate it from the correspondences between the known model of our planar marker and the markers detection in the image.

As, we already have the corners for the AR tag, we now use the projection matrix P to find the 4 corners of the top face in the camera coordinate frame. Finally, we use the **draw_cube()** function to render a 3D cube in our 2D image plane as can be see in the figure(3-5).
We have been able to successfully detect all the three tags in the multiple tag video and add lena to all three at the same times and also create cube for all three of them.

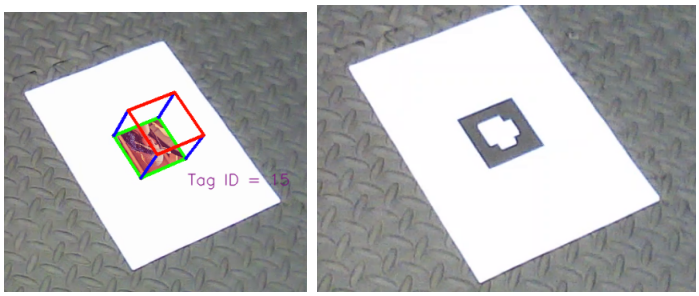

Figure 4: AR tag detected and cube projectedvs original AR tag for video Tag0

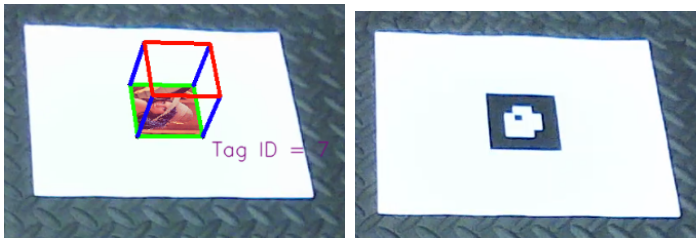

Figure 5: AR tag detected and cube projected vs original AR tag for video Tag1

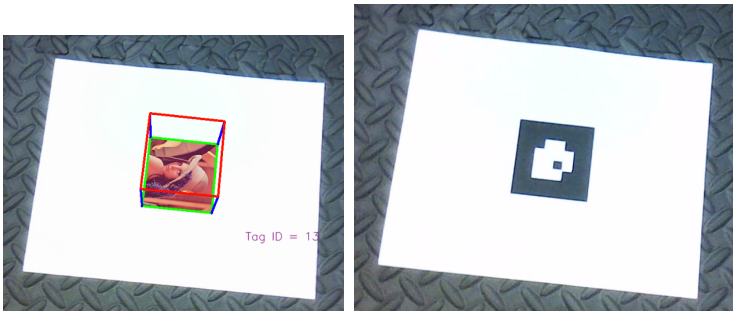

Figure 6: AR tag detected and cube projected vs original AR tag for video Tag2

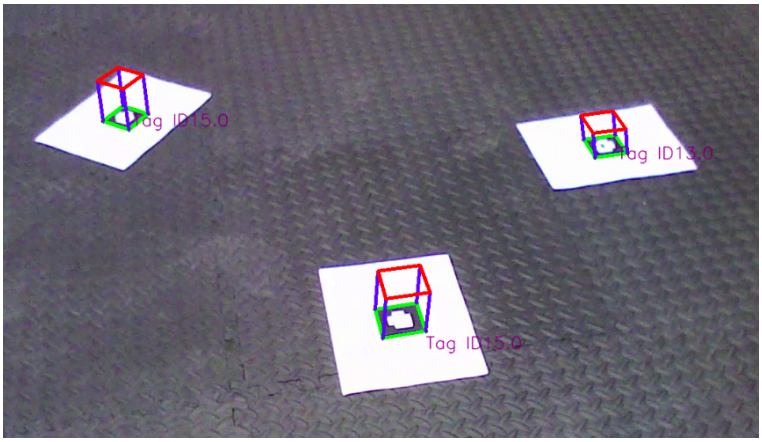

Figure 7: AR tags detected and cube projected for video multiTags

Problems Encountered

1. Due to jitters in the video and blurry tags we faced a lot of problems in extracting information from the tags. In videos where there isn't enough focus on the tags (like the "multitags" video) in a lot of frames all that was detected was just a blob. We couldn't figure out a way to solve issue, as detecting the info on the tag is sometimes not even possible with the naked eye. We tried smoothing the video by finding the trajectory between frames of the video using homography. In this we faced an issue with saving the corrected video and due to time constraints couldn't apply it.

2. An another interesting problem that we had no idea how we managed to solve it was a multiplication problem. "np.dot" or "*" didn't work but "np.matmul" worked while solving the projection matrix. It's not like "np.dot" didn't work at all, its just that our projection matrix calculation gave us very weird results.

3. We also didn't get time to implement correction techniques (like RANSAC) on our corner detection to reduce noise in that(this small indistinguishable noise results in

large jitters in our cube plotting). We just made the best of the situation and made a video of dancing cubes with the Uptown Funk soundtrack (Link - fun.mp4) (don't judge us, work gets frustrating sometimes).

NOTE

To run the code follow the README file. All the videos are available on our drive. Follow this link