ENPM673 - Perception for Autonomous Robots
Project 1

Cheng Chen, Haixiang Fang, Timothy Werder

Due February 25, 2020

Introduction

Augmented reality is a field in computer vision where live or pre-recorded videos or images are augmented by the user to provide information on the environment. For our project we were required to read and distinguish angular information about one to three QR codes and impose a picture or three dimensional cube on each code.

Problem 1 - Detection

Identifying the Tag

To start we first began by thresholding each frame as the video is read in. This threshold was first developed by gray-scaling the frame, then using a bitwise-or mask over the frame. We selected this mask as this would provide a binary 1 in the image for any values between the selected range and a zero for anything outside of it.

We then used a blob detection algorithm from opencv to find any distinct objects in the environment of the frame. This would spit back the center of the location and it's approximate radius. Iterating through each blob point and isolating the area around the frame into a modified frame, we could then use opencv's find the outermost contours. If the contours happened to not be detected, done by checking the area between the previous frame and the current frame, then it would use the previous frame's contours. An image of an isolated and contoured frame can be seen below.

Figure 1: Isolated and Contoured AR tag.

Once contours were established we used opencv's 'minrect' (minimum rectangle) to develop the smallest necessary rectangle to cover the QR code. These points were then sent through out homogeneous function to find the necessary matrix to dewarp the image.

Problem 2(a) - Superimposing the image

Homogeneous Transformation

As we have found the contours, the contours can be sorted according to their area. Then searching from largest contour, the contour of board can be found. From this contour (8 x 8 grid), the four corners can be detected. Those four corner is the four target points($p1$, $p2$, $p3$, $p4$) that Lena is going to be imposed. The four corner of flat Lena image is source points (1, 2, 3, 4). Now, to find Homography, we know that:

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 * x_{p1} & y_1 * x_{p1} & x_{p1} \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 * x_{p1} & y_1 * x_{p1} & x_{p1} \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 * x_{p2} & y_2 * x_{p2} & x_{p2} \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 * x_{p2} & y_2 * x_{p2} & x_{p2} \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 * x_{p3} & y_3 * x_{p3} & x_{p3} \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 * x_{p3} & y_3 * x_{p3} & x_{p3} \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 * x_{p4} & y_4 * x_{p4} & x_{p4} \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 * x_{p4} & y_4 * x_{p4} & x_{p4} \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix} = 0 \tag{1}$$

Use singular value decomposition (SVD), $A{=}U\,D\,V^T$, the smallest eigen vector of A are the element of homography.

$$h = \frac{[v_{19}, ..., v_{99}]^T}{v_{99}} \tag{2}$$

The H found now is reshaped into a 3x3 square matrix and normalised, so that the last column last row element of this matrix is one .

De-warping/ Warping the Image

Once the homogeneous matrix was acquired, image imposing was made possible. This was done by multiplying the homogeneous matrix across each pixel in the imposing image. Dividing out the third column we then check to make sure the new pixel is within the bounds of the image being imposed on. If the pixel is withing the image we then assign the imposing image pixel onto the image being imposed on.

Angle Finding

As warping done, we can start to detect a AR tag. First, to extract ID from a tag, the 8x8 tag image can be use to build a 8x8 grid. As each grid represent the corresponding area, if the pixel values mean is above 180, it's believed to be a white grid, otherwise to be a black grid. Once the padding removed, the inside the 4x4 grid gives the orientation, where upright figure has white in lower right corner. As direction is figured out, the inside the 2x2 grid tell the tag ID in binary representation, and ID computed by multiply 1, 2, 4, 8 accordingly in the clockwise direction from top-left.

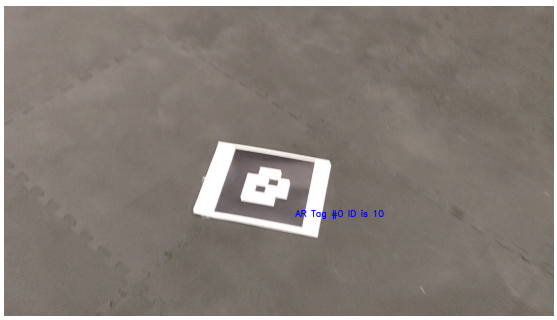

Figure 2: ID extraction on a single tags from data1.

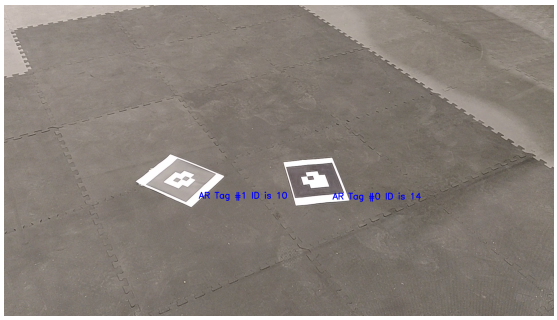

Figure 3: ID extraction over multi-tags from data2.

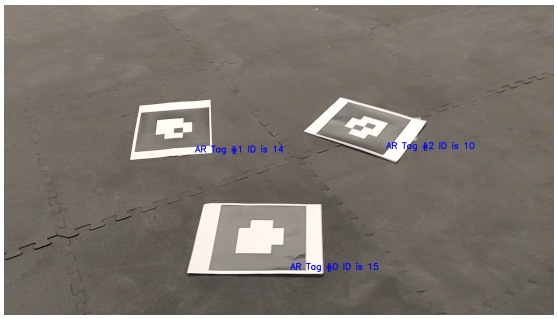

Figure 4: ID extraction over multi-tags from data3.

Superimposing Lena

Using the angle from the angle finder, we then send the size of Lena, (the image to be imposed) and the contour square of each QR code and find the homogeneous matrix between each one. The Lena image is then dewarped and imposed on the image. To correct for the QR's frame being a cutout of the main frame, the minimum and maximum are set to be the the center of blob in the main frame ad adjusted for the radius of said blob.

The main frame is then returned for the next blob to be imposed on. Once all blobs have been replaced, the main image is then displayed and the next frame is readied to repeat the process. Lena.png imposed on all three videos can be seen below.

Figure 5: Lena superimposed on the AR tag from data1 and data2.

Figure 6: Lena superimposed over multi-tags from data3.

Figure 7: Lena superimposed over a single tags from data1.

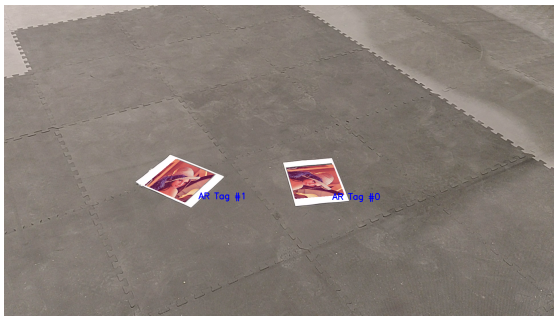

Figure 8: Lena superimposed over 2 tags from data2.

Figure 9: Lena superimposed over 3 tags from data2.

Problem 2(b) - Creating a virtual cube

The coordinate system of camera C is rotated and translated with respect to the world coordinate system W. In augmented reality, this relation is given by the equation from the supplementary material:

$$x_c = P\, x^w \tag{3}$$

Where the projection matrix P can be written as a product of two matrices, K and $[R \mid t]$.

$$P = K\,[R \mid t] \tag{4}$$

Where Matrix K is called intrinsic camera matrix which contains the parameters estimated during camera calibration. Matrix $[R \mid t]$ consists of the rotation matrix R and translation vector t. We can rewrite the rotation matrix using its columns, then $[R \mid t] = [r1,\ r2,\ r3,\ t]$.

Once we have computed the homography H, the first step is to calculate a matrix which is the dot product of the H inverse and K. The obtained matrix's first and second columns are proportional to the first two columns of the rotation matrix $r1$ and $r2$. While the third one is proportional to the translation vector t. We then calculate a scaling factor λ which is used for scaling the transformation matrix to be unit length. λ is calculated as the following equation:

$$\lambda = \left(\frac{\left\| K^{-1} h_1 \right\| + \left\| K^{-1} h_2 \right\|}{2} \right)^{-1} \tag{5}$$

Now the elements of the rotation matrix R and translation vector t can be computed by multiplying λṠince the rotation matrix is orthonormal that r3 column of the rotation matrix will be got through the cross product of $r1$ and $r2$.

For generating a cube on the tag we fist define the cube coordinates in the world coordinate system and then project them into the image plane. Using the draw contours function, we got the images shown below:

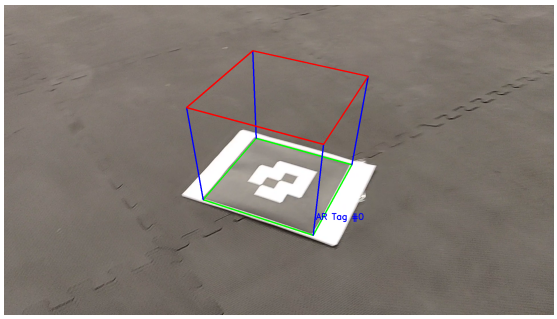

Figure 10: Placing cubes over a tag from data1.

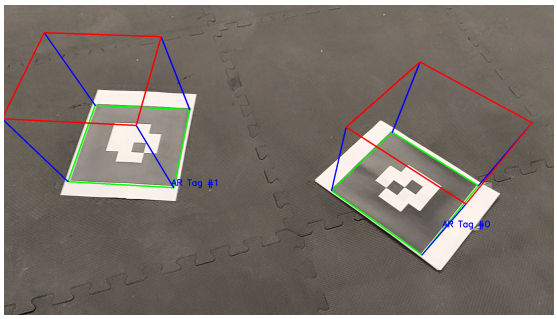

Figure 11: Placing cubes over multi-tags from data2.

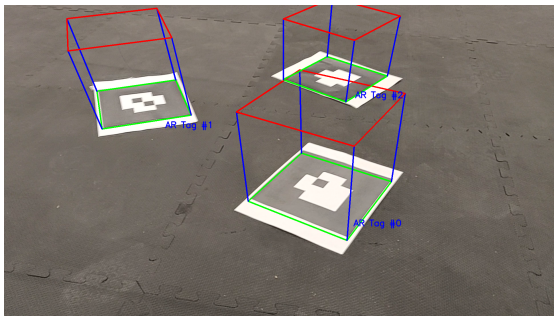

Figure 12: Placing cubes over multi-tags from data3.

Conclusion and Lessons Learned

We conclude that his project was ultimately a powerful introduction to computer vision techniques and team coding projects. We were able to get all parts running and working, though we believe with additional time we could have narrowed down on some of the bugs and inconsistencies such as frame flickering.

Some lessons learned from this project mainly focus on how to write code as a team. Establishing a set environment to code in can cut down on issues with integrating individual functions. We claim this as numerous issues arose from attempting from developing functions individually. Another lessons learned is to develop the main function using in built functions (even though they may be outlined and being needed to be developed from scratch). This is because not having a working main function can cause issues down the line when other modules or functions are in need to be developed. Additionally, completing the main file will allow the developers to determine what modules will be necessary to develop for everything to work smoothly.