Project1

Derek Thompson, Vishnu Sashank Dorbala, Surabhi Verma

February 25, 2020

Problem 1

We need to detect an AR tag from a video stream, and find its orientation and ID.

Using OpenCV, we first extract images from the video stream, frame by frame. Once this is done, we convert each of these RGB frames into grayscale, perform binary thresholding, erosion and detect the edges using a *Canny* edge detector. We use the OpenCV functions *cv2.theshold*, *cv2.erode* and *cv2.Canny* to do this. The outcome of this is presented in figure 1.

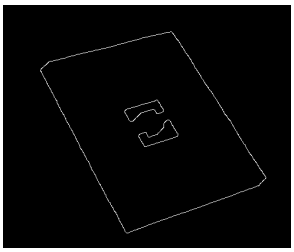


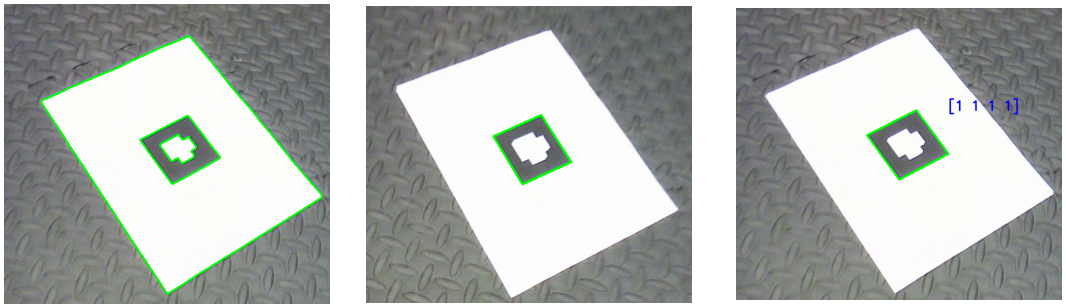
Figure 1: Output of Canny Filter

We use *cv2.findContours* to detect contours on this edge image (Figure 3). The detected contours are refined based on the following criteria,

- The desired AR tag is a quadrilateral and hence has only 4 edges. Thus, only contours with 4 edges are detected.
- The area of the contour is conditioned to be less than a certain amount of pixels.
- The cv2.RETR_TREE option is used to obtain the hierarchy of the contours. We eliminate the contours that do no have a parent, as every AR tag will have a parent.

The outcome is shown in figure 2. To ensure the homography was correct the array of contour points was checked to ensure the definition of the points was clockwise as the other homograpy points were defined. Using the order of the upper most two detected points the array was flipped if the order was found to be negative.

With the generated contour points we ran the points through a homography function to put the tag image onto a flat 80x80 grid. This homography function calculated the defined A matrix from the contour points given and found the singular value decomposition to find the homography matrix H. The indexes for each pixel in the Lena image and each pixel index in a 80x80 grid were generated at the start of the program into nx3 arrays that could be multiplied by the H matrix to get a final position of the tranformed image. Each pixel of the pulled tag



(a) Before conditioning (b) After conditioning (c) After tagging

Figure 2: The contours before and after refinement and tagging

Figure 3: Tag with ID 1111 after being pulled from image and put through the threshold

was taken from the original image as defined by the product of the homography matrix H and the generated index array. This result is shown in Figure 3.

After we segment out the AR tag from the image frame, we then need to identify the AR tag. Initially the tag is put through a threshold to find a binary representation of each location on the tag. The tags orientation is found by looking for the odd white square in the inner 4x4 grid that represents the bottom right corner. When the tag is pulled it often contains motion blur and noise. In order to combat this noise when looking for the orientation grid space the average value of that entire grid space is taken and the value with the highest average pixel value is taken and the corner. Additionally when calculating the orientation, the orientation error is saved to rotate the initial found contour lines for when Lena needed to be superimposed on the image.

The encoding scheme is given by the innermost 2x2 matrix and needs to be read clockwise from the top left. In this case, as all of the inner 2x2 squares are white, the encoding for this particular image is (1111) (Figure 2c). The encoding is done by taking the the value of the pixels in each of the grid spaces corresponding to the encoding squares and if any pixel was with the threshold the grid space was assigned as a 1. This encoding was then put on the image at the tag position for clarity.

Problem 2

For the second problem, we need to superimpose an image and a virtual cube onto the AR tag.

Problem 2(a) - Superimposing Lena onto the AR tag

For this task, we need to superimpose a picture of Lena onto the AR tag. This is done using the contour points that have been flipped to be clockwise definition and rotated to have the same orientation as the defined corners as Lena, the homography function was agian used to superimpose Lena. The coordinates on the final image were computed using the precomputed Lena index array to find the final destination of each pixel in the Lena image. This was carried out for each Lena pixel. The result is shown below in Figure 4.

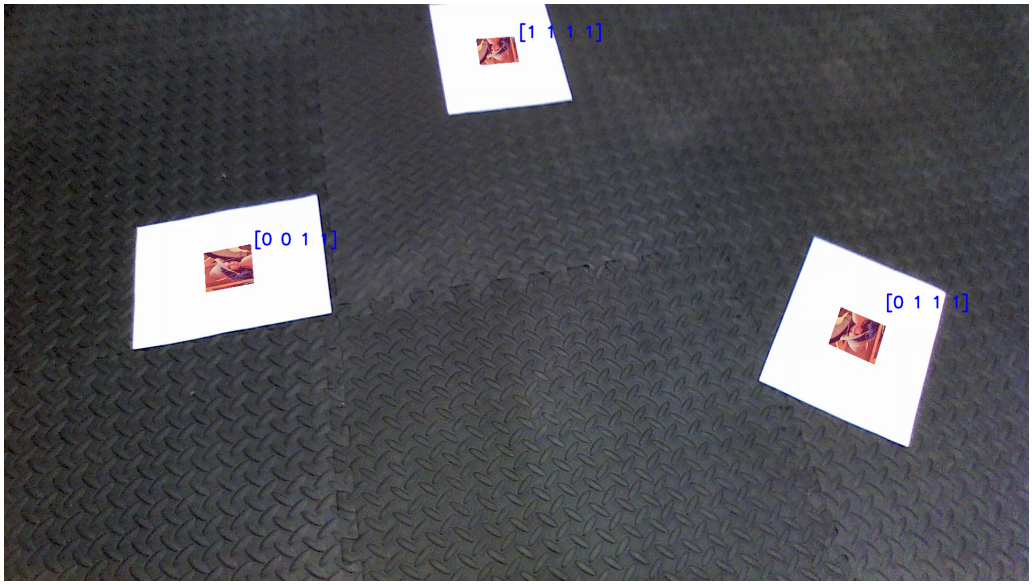


Figure 4: Figure of superimposed images of Lena

Problem 2(b) - Superimposing Cube onto the AR tag

For placing the cube on the AR tag, we first define the cube coordinates and compute the homography between the world coordinates and the image frame.
We then have to compute the projection matrix, which is a product of the homography matrix and the intrinsic camera calibration matrix. The resultant matrix has its first two columns representing the first two columns **r1** and **r2** of the rotation matrix. The **r3** column is given by $r3 = r1$ x $r2$.
In order to compute the translation matrix, we make use of a scaling factor λ. This is computed by,

$$\lambda = \left(\frac{\left\| \boldsymbol{K}^{-1}\boldsymbol{h}_1 \right\| + \left\| \boldsymbol{K}^{-1}\boldsymbol{h}_2 \right\|}{2} \right)^{-1}$$

This λ is then used to compute the projection matrix which then gives us the coordinates of the cube on the tag. Finally, once we have all the desired points projected onto the AR tag, we use *cv2.line* to draw lines between them to form a cube.

Figure 5 shows the outcome of this on Tag 0.

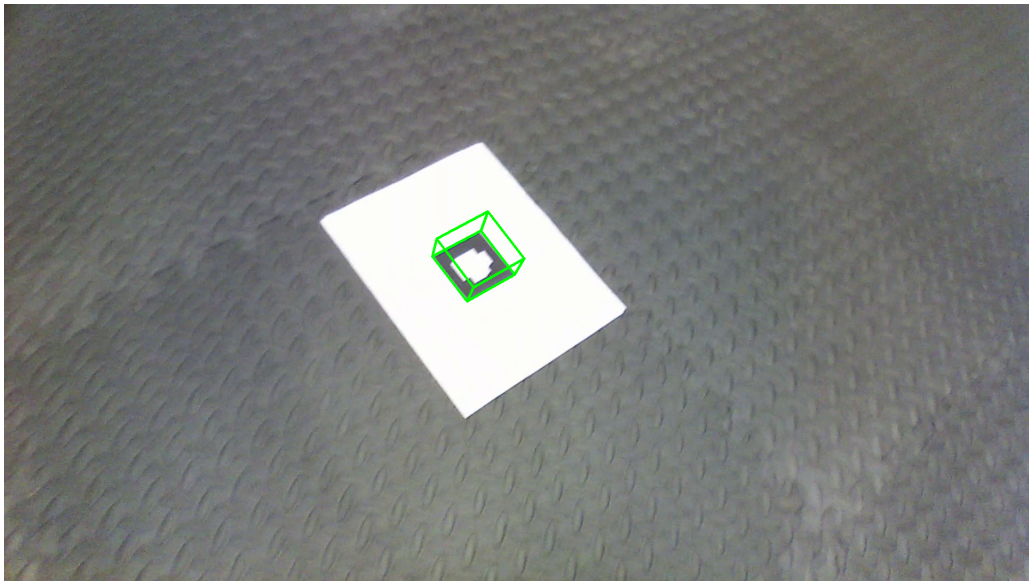


Figure 5: Figure of superimposed cube onto AR tag

Output Videos

Links to video output with encoding and Lena superposition:

- Tag0
- Tag1
- Tag2
- Tag Multiple

Links to video output with cube superposition:

- Tag0
- Tag1
- Tag2
- Tag Multiple

Conclusion

In this project, we implemented homography and warping of an image onto an AR tag. Later, we also superimposed an AR cube onto the tag. In the process, we learnt a great deal about fundamental multi view geometry concepts in Computer Vision.