ENPM673: Project 1 Report
AR Tag Detection and Tracking

Smriti Gupta
sgupta23@umd.edu

Varun Asthana
vasthana@umd.edu

Saumil Shah
sshah293@terpmail.umd.edu

***Abstract*— This project aims at detecting and decoding a custom AR-Tag in multiple videos. Once a tag is detected, its ID is to be reported as per the encoding scheme provided in the problem statement, then detect the orientation of the tag in order to replace the tag with another image with correct homography. Last phase of the project is to replace the tag with a virtual cube in the image coordinate system.**

I. Pase 1: AR-tag id detection

In the this phase to detect the tag id, the first step is to identify and segregate the tag from the remainder of the frame image of the video. To do this we have utilized *cv2.findContours* function. Along with the actual tag contour (boundary around the black box), the function returned many other contours based on the objects visible in the frame image.

The next step to filter the correct contour from a set of contours we used the hierarchy return value of the *cv2.findContours* in commutation with a condition on minimum and maximum area of the contour using *cv2.contourArea* function. With this we were able to filter out the data but it still had some noise. To further filter out the noise we used the *cv2.approxPolyDP* function to approximate which contours form a quadrilateral.

The above methodology helped to reduce most of the noise, but the a new error was faced. The approximation of the quadrilateral by the function *cv2.approxPolyDP* was not accurate and hence the tag inside the black square was also classified as a quadrilateral and it even satisfied our filtering condition of the contour area. This is shown in Fig 1. We cannot reduce the minimum area limit further as the video plays out, the distance of AR-Tag from the camera changes and hence its size reduces.

To further rectify our data set of contours to have only black bounding box contour we explored the fact that the tag is printed on a *white-paper*. Hence if we shift slightly in all the 4 directions around the vertices of the contour of the black box, then it will have white pixels in 3 directions and a black pixel in the remaining direction. The result of this step is shown in Fig 2. Using this filtering we were also able to categorize which vertex is at Top Left, Top Right, Bottom Right and Bottom Left (relative to each other).

Now the next step was to detect the tag id from the inner grid of the 8×8 tag (as shown in the reference image of the tag Fig 3). In order to form a grid around the detected AR-Tag from the frame image, we first need to bring the image tag into the perspective of the world coordinate system (reference tag).

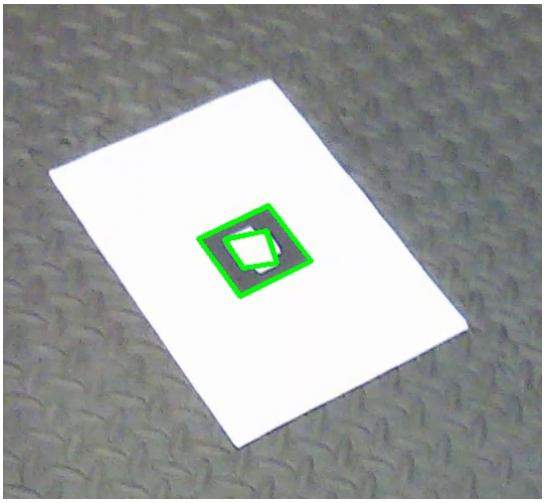

Fig. 1: Trial 1 for filtering of contours to get only black bounding box of the AR-Tag

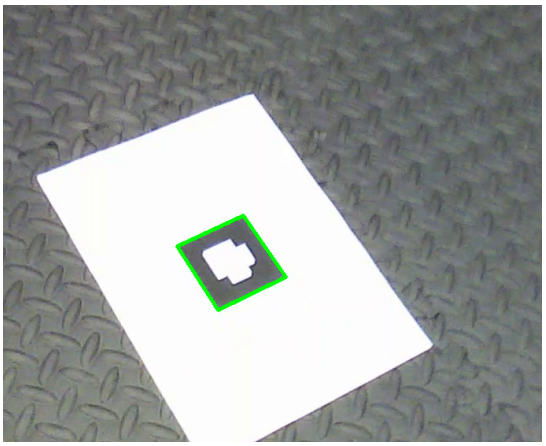

Fig. 2: Correct filtering of contours to get only black bounding box of the AR-Tag

Since now we already have the correct contour data with its vertices with duly sorted in clock-wise rotation from top left to bottom left, we can find the homography from the image coordinate system to world coordinate system by solving the below equation with he use of *numpy.linalg.svd*

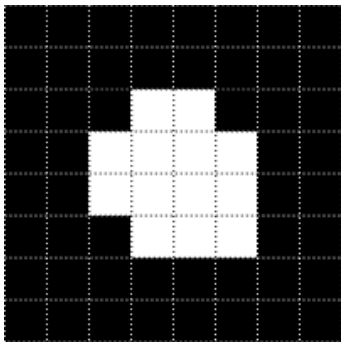

Fig. 3: Reference AR-Tag with grid

Fig. 4: Image of detected AR-Tag warped into the world coordinate system

function.

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

The homography matrix calculated above (H_w^i) relates the image coordinate system to the world coordinate system. But in order to find the projection matrix *P*, we need the homography from world coordinate system to the image coordinate system (H_i^w). This can be found simply by taking the inverse of the H_w^i. With this we then calculated the *P* matrix following the procedure provided in the supplementary document.

The next step was to warp the detected tag into the world coordinate system. Since we only have the coordinates of the vertices of the detected contour, in order to warp the image the requirement was to have the coordinates of all the pixels inside the contour. To get this data, we applied the H_i^w on the reference marker to obtain the corresponding coordinates, which are actually the coordinates inside the detected contour. Once this data was available, we were able to warp the AR-Tag from the frame image into the world coordinate system. The result is shown in Fig 4.

From the Fig 4 it is visible that a lot of magnification has distorted the image quality. This data will not be very useful for decoding the tag id and its orientation. To correct this we generated a new tag image by thresholding the pixel values to 0 (pure black) or 255 (pure white). The result is shown in in Fig 5a. It can also be observed that the inner tag occupies a grid larger than 4×4. This is due to the fact that the detected contour for the black bounding box was not perfectly correct and hence their is random scaling in each frame. To correct this, we summed the number of white pixels along all the rows and took the maximum value of it. Since it is known that the maximum width of the tag (maximum sum of the white pixels) should be within the 4×4 grid, we re-scaled the image accordingly. The output image is shown in Fig 5b.

(a) AR-Tag after thresholding (b) AR-Tag after grid alignment

Fig. 5: Grid setting of warped AR-Tag

Now by traversing each grid block we can generate a grid matrix for the tag. For the shown tag in Fig 5b, the grid matrix would be-

$$Tag - Grid = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

By using the tag-grid, we can now obtain the angular rotation of the detected tag and its id by comparing it with the reference grid for upright position. For orientation detection only the corners of the grids are important. Once the orientation is detected, the tag id is extracted from the most significant bit to least significant bit (shown in Tag-Ref-Grid by 1,2,3 and 4 with 1 being the most significant bit and 4 being the least significant bit). Output result of detected tag id is shown in Fig 6.

$$Tag - Ref - Grid = \begin{bmatrix} 0 & * & * & 0 \\ * & 4 & 3 & * \\ * & 1 & 2 & * \\ 0 & * & * & 1 \end{bmatrix}$$

Fig. 6: Detecting AR-Tag ID

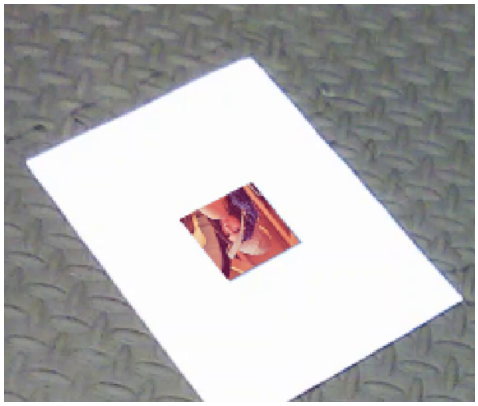

Fig. 7: Image of Lena which will replace AR-Tag

II. Phase 2 : Replace the AR-Tag with image of Lena

In order to replace our tag with the image of Lena (shown in Fig **??**) in correct orientation, we need to detect the location of the tag and its orientation. Tag vertices and orientation detection has already been completed in the first phase of the project. Using the orientation of the tag, we initially rotate the Lena image. We also have the projection matrix *P* computed. With the Lena image as the world reference at *Z=0*, we can get the corresponding coordinates for it in the image coordinate system. Once all the corresponding coordinates are found, we can now warp the Lena image from world coordinate system to the image coordinate system. The output results are shown in the Fig 7.

III. Phase 3 : Placing a virtual cube on the AR-Tag

In continuation to the approach used to replace the Ar-Tag with the image of Lena, using the projection matrix *P*, we can superimpose a cube over the AR-Tag. We have assumed the cube base to be of the same dimension as of the reference AR-Tag image (which in this case is 200 pixels). In the world coordinate system, the base of the cube will still be at *z=0* while the opposite top surface will be at *z=200*. Using the projection matrix with *z=0* and *z=200*, we can project the vertices of the cube into the image coordinate system and then connect all the vertices (in correct order) to draw a cube. The result of this methodology is shown in Fig 8.

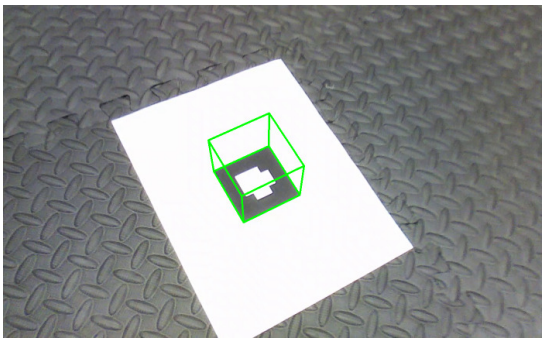

Fig. 8: Placing a virtual cube on the AR-Tag

Fig. 9: Blurred (distorted) AR-Tag detected with camera in motion

IV. Conclusion

We were able to successfully detect the AR-Tag id, replace the AR-Tag with the image of Lena in correct orientation, and finally superimpose a virtual cube on the detected AR-Tag.

It was at times noticed that the detected id and orientation was not correct. This was due to the fact that when the camera position is changing, we are getting a blurred or distorted image of the tag in the frame image, which affects the tag-grid formation (from where we are extracting the tag id and orientation data). Fig 9 shows the detected tag quality when the camera is in motion. With this effect at times we are also not able to detect the tag properly, which leads to flickering of the output.

A. Output Videos

Videos can be found here.

Videos : `https://drive.google.com/open?id=1QkOwxFsOhuP44F8y5A4iU7RAB1itW2J2`