# “[ENPM 673] HOMEWORK - 1”

ENPM 673 – PERCEPTION FOR AUTONOMOUS ROBOTS

PROJECT 1 REPORT


**ADITYA VAISHAMPAYAN -116077354**

**ISHAN PATEL – 116169103**

**NAKUL PATEL – 116334877**

Instructor: Dr. Cornelia Fermuller

# Abstract:

The purpose of this project is to understand the working of the AR tags and learn the concepts of the projective geometry. The project is divided into three parts. The first part is about detecting the AR tag's corners and its ID. This covers the detection part of the AR tag. The next task in the project was to superimpose an image over the detected AR tag and make it rotation invariant which encompasses the tracking part of the AR tag. Then the third part of the project encompasses placing a virtual cube over the detected AR tag. We use the projection matrix to do this.

# 1) Preprocess filtering:

Every frame in the video needs to be preprocessed in order to apply various image processing operations on it. The first basic step is to covert the image into a gray image and then removing various noises from it. Further we do binary thresholding to convert it into a black and white image.

1. Converting to gray
2. Remove salt and pepper noise using median filter
3. Do binary thresholding
4. Find contours – attributes: cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE. This function gives us the 4 corners of the AR tag

The above preprocess filtering steps are common to all the processes that we do. Now we show the process of detecting the TAG ID and orientation.

# 2) Tag ID and orientation

In order to obtain the TAG id, we need to first obtain the 4 points of the AR tag which are obtained through corner detection or by obtaining the contours and then extracting the four corners of the AR tag. We use the following functions to obtain the contours cv2.arcLength, cv2.approxPolyDP, cv2.contourArea and cv2.isContourConvex.

Our goal is to obtain a bird's eye view of the AR tag. To do that we need to rotate and translate the AR tag to obtain its perspective view. So, upon obtaining the four points from the contour, we find the homography matrix of the points of the AR tag with respect to the new windows on which we need to project the birds eye view. To obtain the homography matrix we have created our own function that takes in the source and destination points from the frame and the image to project on and gives us back a 3x3 matrix.

In order to obtain the Tag Id, we resize the birds eye view image of the AR tag to a 8x8 size image. This makes it easy to access the individual pixels and thus obtain the Tag ID. According to the given instructions, we obtain the MSB and the LSB of the AR tag. And use the pixels in a cyclic manner. The maximum no of tag IDs that once can generate is 16 because only 4 data pixels of the image are to be taken into consideration.

TAG ID in binary form 1001

| | MSB | | | LSB |
|---|---|---|---|---|
| Pixel | 1 | 0 | 0 | 1 |
| Weight | 8 | 4 | 2 | 1 |
| Result | 8*1 = 8 | 4*0 = 0 | 0*2=0 | 1*1=1 |

Now we sum the results to obtain the TAGID which is 9 in this case.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | **1** | **0** | 1 | 0 | 0 |
| 0 | 0 | 1 | **1** | **0** | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | **1** | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The bits shown in red color are used to represent the Tag id of the AR tag and then green bit is used for its orientation

Next in order to obtain the orientation of the AR tag, we need to consider pixel no's [2,2], [5,2], [5,5], [2,5]. Thus, we obtain the LSB by determining the position of the orientation bit. The position of the orientation bit is then useful to superimpose the Lena image and make it rotation invariant.

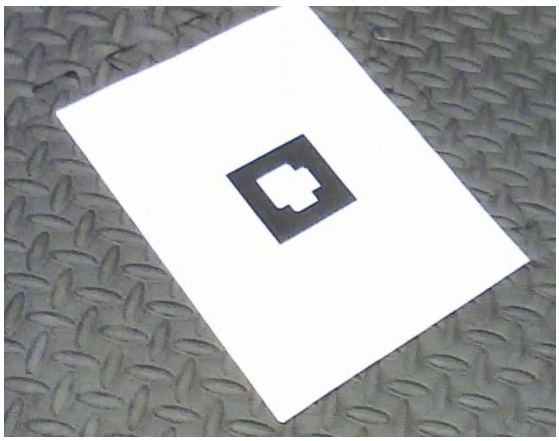

Fig: AR tag

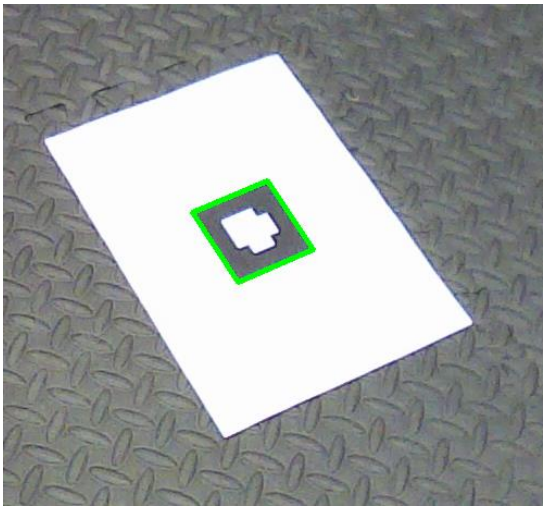

Fig: Contours around the AR tag

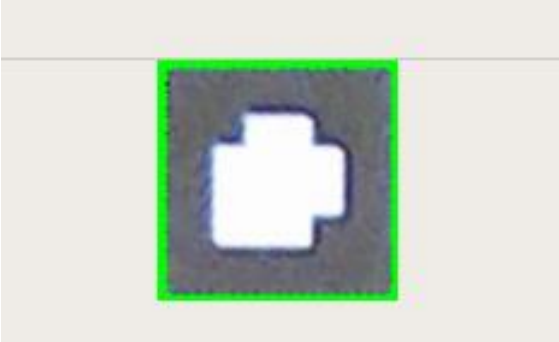

Fig: Bird eye view of the AR tag

# 3) Superimposing an image onto the tag

In order to superimpose the Lena image over the tag, we need to do the same procedure as above where obtain the bird's eye view of the AR tag. Then we used the homography matrix obtained above in order to superimpose Lena over the AR tag. In order to implement this, we have used our own function for calculating the homography matrix.

Also, in order to match the image orientation of the transformed template with the tag we have used the orientation bit from the AR tag. The coding has been done in such a way that as and when the frame rotates, the rotated orientation bit is mapped to a particular corner in the Lena image and its mapping is kept constant during the entire video i.e. throughout all the video frames. We have used switch cases in python in order to implement this.

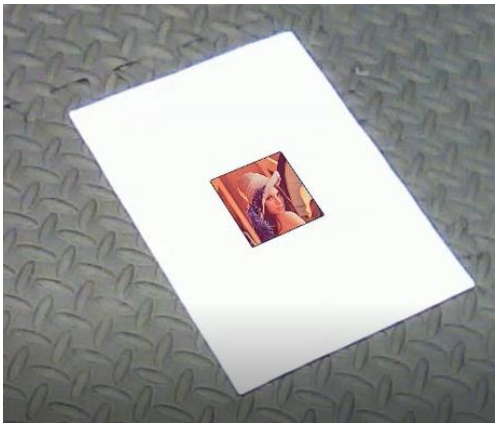

Fig: Lena Image orientation 1

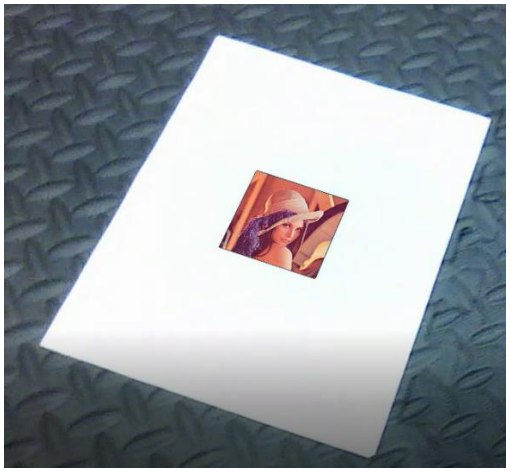

Fig: Lena Image orientation 2

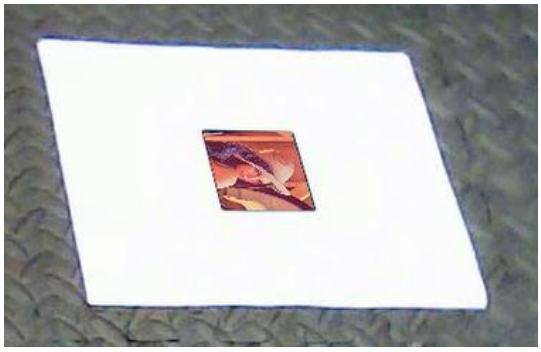

Fig: Lena image orientation 3

From the above three images we can see that no matter what the orientation of the AR tag is, which we can see from the orientation of the paper, the super imposed Lena image also rotates along with the paper and thus is rotation invariant

## 4) Placing a virtual cube on the tag

Placing a virtual cube on the AR tag is similar to superimposing Lena over the tag. We begin by computing the homography between the world coordinates and the image plane. Then we use the function that is defined by us for obtaining the projection matrix. We use the calibration matrix and the homography matrix as the input parameters and the function returns the projection matrix. Later this matrix is multiplied with the world coordinates of the cube to result in 8x3 matrix. We need to divide the first two columns of this matrix with the third column and thus we obtain the coordinates of the cube in the video frame.

In order to draw the cube, we implement a function called draw cube that basically draws lines between the points from the 8x3 matrix. And hence we are able to render a virtual cube this way.

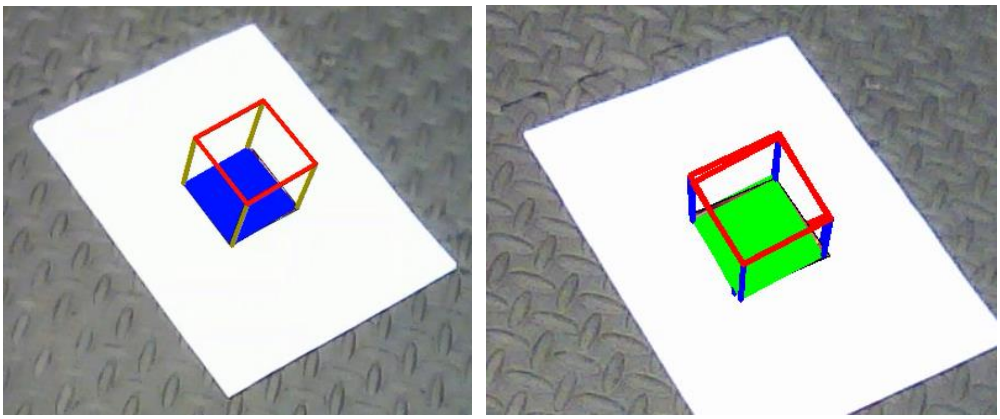

Fig: Placing a virtual AR cube on the AR tag

## 5) Conclusion

By implementing this project, we learnt about projection geometry, finding homography and also learnt about projecting virtual 3D points in a 2D image plane. We were able to successfully implement detecting the AR tag in a given image and also determining the coordinates of its contours. We were also able to superimpose an image over the AR tag and similarly we extended the process by doing it for rendering a 3D cube over the AR tag. Hence, we were able to successfully implement Augmented reality.