ENPM673 Project 1

Perception for Autonomous Robots

Prasanna Thirukudanthai Raghavan
UID: 118287546

Department of Robotics
University of Maryland
United States

Contents

Chapter 1

Problem Statement:

This project will focus on detecting a custom AR Tag (a form of fiducial marker), that is used for obtaining a point of reference in the real world, such as in augmented reality applications. There are two aspects to using an AR Tag, namely detection and tracking, both of which will be implemented in this project. The detection stage will involve finding the AR Tag from a given image sequence while the tracking stage will involve keeping the tag in “view” throughout the sequence and performing image processing operations based on the tag’s orientation and position (a.k.a. the pose). You are provided multiple video sequences on which you test your code. The problem is divided into 2 parts which are further divided into 2 parts to help us break down the series of steps required to achieve the output.

Chapter 2

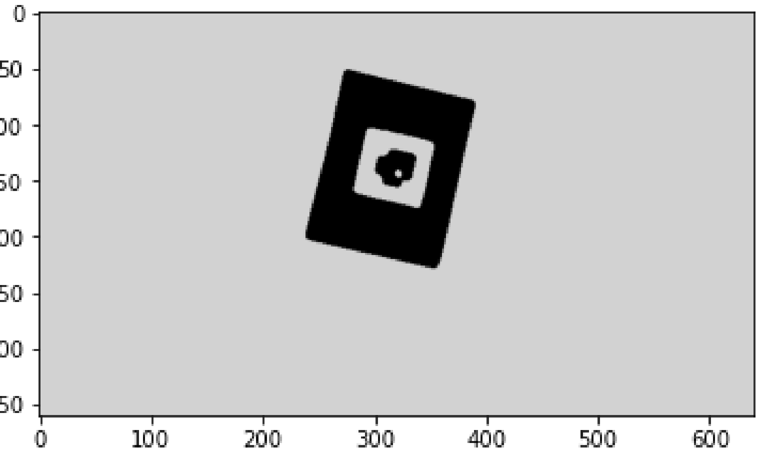
Problem 1 – Detection

2.1 1.a AR Code detection:

The task here is to detect the April Tag in any frame of Tag1 video (just one frame). Notice that the background in the image needs to removed so that you can detect the April tag. You are supposed to use Fast fourier transform (inbuilt scipy function fft is allowed) to detect the April tag.

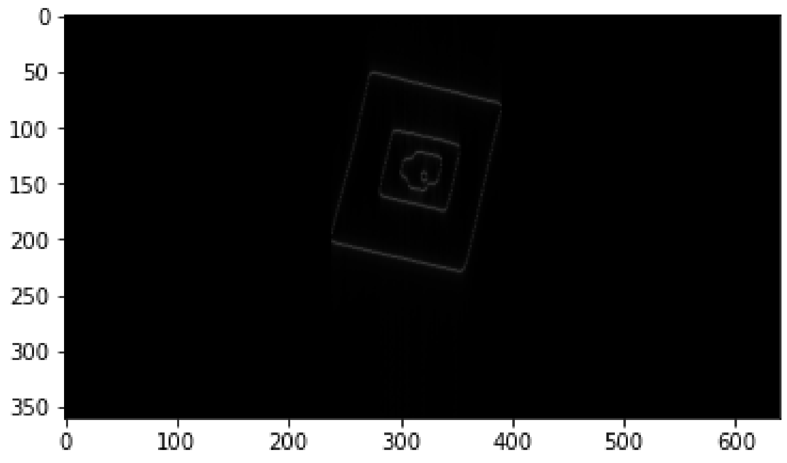
2.1.1 Filtering:

We get the frames from the video and proceed to pre-process the image so we can get a proper image on which we can perform filtering to get the edges and corners. For this purpose we initially convert the image into gray scale and proceed to blur before we threshold the image and convert into a binary image. Here the blur removes noise that are outside the required region. Here the thresholding and the blur kernel size is chosen by the output accuracy.



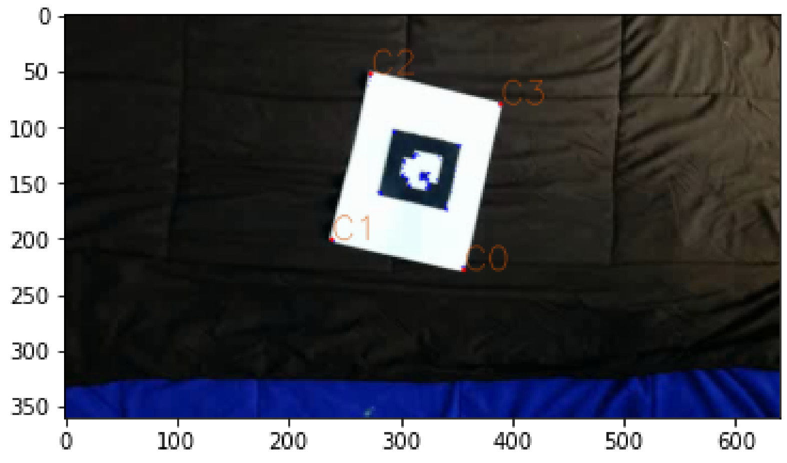
2.1.2 Fast fourier transform(fft):

The process of conversion of the image from spatial into frequency domain helps us greatly as we can filter out the image features as signals. Now, using the high-pass filter we keep the high frequency features such as corners or edges, These features will help in further processing. For this we use a circular mask on a grey thresholded image to filter the image and then proceed to inverse fft to get back the normal image.



2.1.3 Corner of Image:

To get the corners we use functions such as Harris corner, Goodfeatures to track, etc from which we sometimes get more number of corners than required. Then we proceed to take certain select corner points from the list of all points by filtering out the points that are extra by taking the combination of maximum and minimum points and then further eliminating the outer points to get the internal corners.

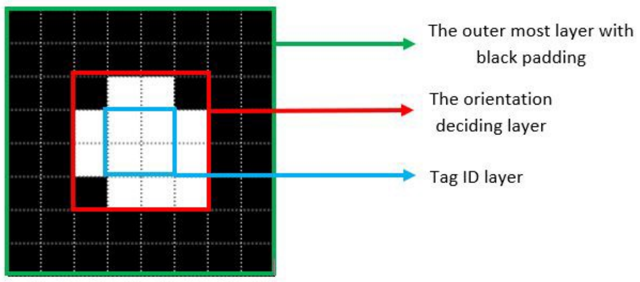


2.2 1.b Decode custom AR tag:

To decode the tag we initially take the size of the tag which is 160 and we are told that the outer 2 squares are simply the padding on the tag and then we proceed to remove the padding and check only the inside of the tag.

2.2.1 Inner tag mapping:

The the orientation and the tag id is given by inner 4 X 4 image.Then we take the 8x8 matrix that encodes the tag data into bigger blocks which finally get replaced by binary values. The inner 4x4 have rotational information. Since we know that the right bottom indicates the position needed to decode we check for that.


The outer most layer with black padding
The orientation deciding layer
Tag ID layer

2.2.2 Bit Significance

The Binary Code, which is read in a clockwise direction of the inner 2 X 2 matrix which has the least significant bit at the top left. Using the required most significant and the least significant bits we proceed to get the tag information.

Chapter 3

Problem 2 – Tracking

3.1 2.a) Superimposing image onto Tag:

3.1.1 Homography:

For the super imposition of image onto tag we need the 4 corners of the image as well as the tag. We from previous steps get the corners of the tag which we use to compute homography. The difference in dimensions between the images causes any disturbance or noise in the output.

$$A = \begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1*xp1 & y1*xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1*yp1 & y1*yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2*xp2 & y2*xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2*yp2 & y2*yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3*xp3 & y3*xp3 & xp3 \\ 0 & 0 & 0 & -x3 & -y3 & -1 & x3*yp3 & y3*yp3 & yp3 \\ -x4 & -y4 & -1 & 0 & 0 & 0 & x4*xp4 & y4*xp4 & xp4 \\ 0 & 0 & 0 & -x4 & -y4 & -1 & x4*yp4 & y4*yp4 & yp4 \end{bmatrix}, x = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix}$$

which is the homography matrix and allows us to get the value of H matrix using the equation

$$Ah = 0$$

and finally we obtain the matrix

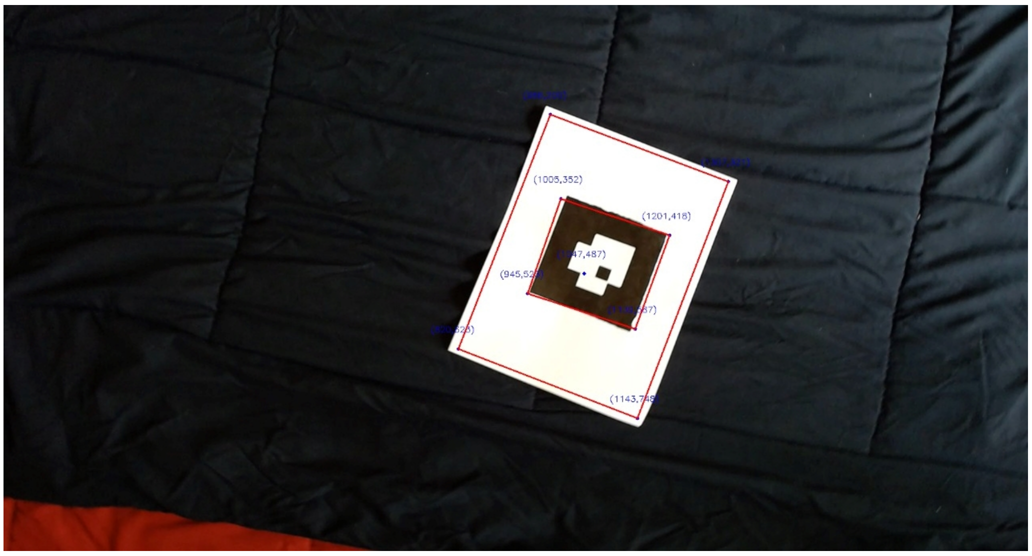
$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

3.1.2 Orientation of the tag

Once we rotate our corner points in that direction. The tag must be reoriented to an upright position. The testudo image is superimposed on the AR tag . The order of the points are used based on the sorting function. Then the orientation of the tag is known from the set of sorted points.

3.1.3 Image warping:

As we already saw that homography was the basis for the warping then we use the same to warp the testudo image using the out generated from the homography matrix. After finding homography from source coordinates to warped(destination) coordinates, we perform warping by multiplying the source coordinates to the obtained homography matrix.
As mentioned in the question pdf, the scaling of the 2 image coordinates, cause white spots which are essentially blank pixels in the warped image. The coordinates due to multiplication gives us values rounded to the nearest integer and due to upscaling and the difference in the image that is being stretched, there are bound to be blank pixels which look like the mentioned "white spots". The warped image coordinates when multiplied by the inverse of the homography matrix lets us find coordinate of the source image matching with warped image matches. The pixel value from the source is reverted to warped closing the blank pixels.



3.2 2.b Placing a virtual cube onto Tag:

The inverse of the homography calculated above is used to calculate the new projection matrix to transform a 3D object to the image plane given by

$$P = K[R]T$$

Where the K matrix is the Calibration matrix which is the intrinsic parameter of the camera. Rotation matrix R and the translation matrix T are the result of the the homogeneous matrix.We use a 3 $\times$ 4 projection matrix to project 3D points into the image plane as cube is a three dimensional entity. Now as we perform homography we cannot take the assumption that the points are planar and will have to take into account another dimension while processing a homography matrix. To steps to obtain a projection matrix(P) from a homgraphy matrix(H) and camera calibration matrix(K) are:

1. the Homography matrix as $H = K\tilde{B}$
2. Then, $\tilde{B} = K^{-1}H$
3. Compute mod of $\tilde{B}$. if it is negative, then $\tilde{B} = -1 * \tilde{B}$.
4. We will calculate λ as $\lambda = \frac{2}{||b_1||+||b_2||}$, where $b_1 and b_2$ are column vectors of $\tilde{B}$
5. Therefore, $B = [\lambda b1, \lambda b2, \lambda b3]$
6. B can be decomposed and written as [r1, r2, t], where r1 and r2 are rotational components and t is translational component.
7. We know that the rotation matrix is orthonormal; the last column must be perpendicular to the first two[2]. Hence, r3 = r1 $\times$ r2
8. The final projection matrix can be written as P = K[r1, r2, r3, t].