Project-1

Ameya Konkar UID: 118191058

March 7, 2022

1 AR tag Detection

As per the task, AR tag detection using Fast Fourier Transform was performed on a selected frame of the video. The purpose of this task is to filter the image. Instead of convoluting the image using a kernel, converted the image in Frequency domain using Fourier transform. Hence, any filter can directly be multiplied on the image instead of iterating through the image. The steps followed are as follows:

1. Converted the image to gray scale.
2. Threshold the image.
3. Used Fast Fourier transform on the image.
4. Created a circular mask and added to the image.
5. Used Inverse Fast Fourier Transform to filter the image.

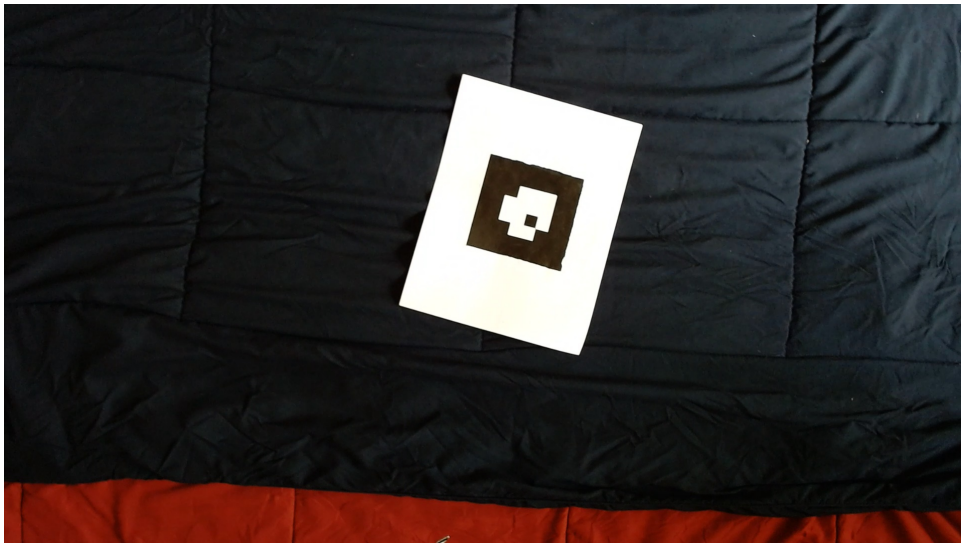

Figure 1: Original Image.

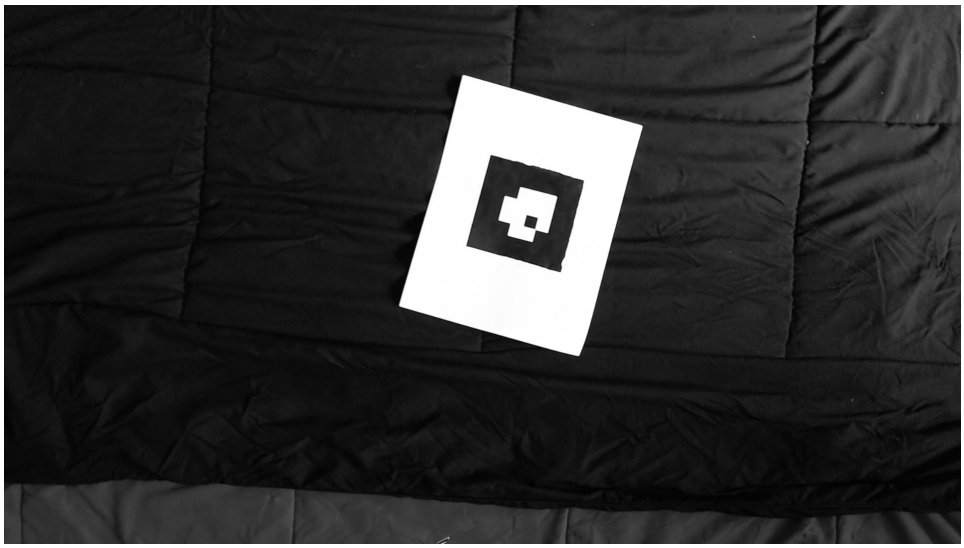

Figure 2: Grayscale Image.

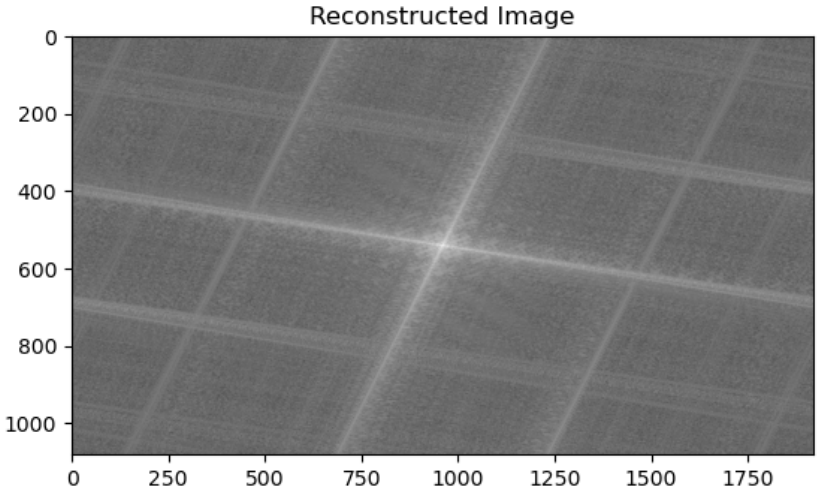

Figure 3: FFT Image.

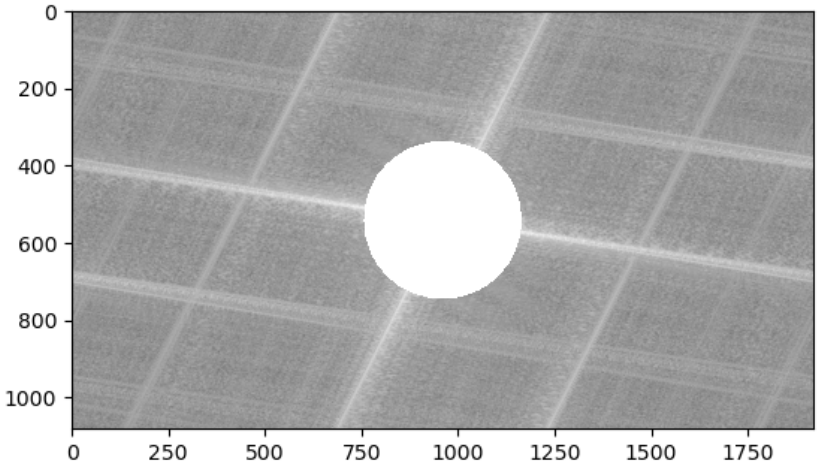

Figure 4: FFT with mask Image.

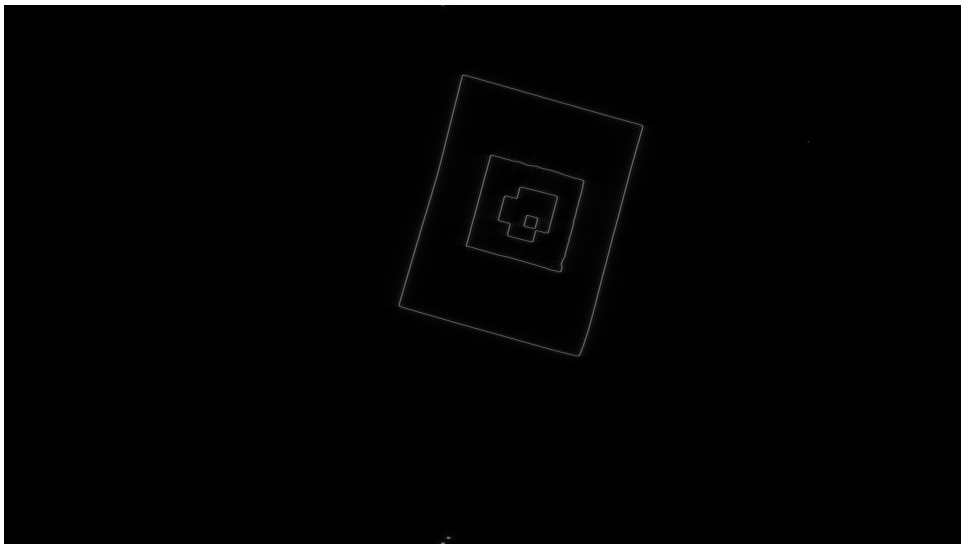

Figure 5: Filtered Image.

2 Decode custom AR tag

As per the given task, Id of the AR tag of a selected frame of the video was performed. The following steps were used to detect the AR tag:

1. Convert the image to grayscale.
2. Threshold the image.
3. Use cv.Floodfill to remove background(black) pixels outside the page containing the AR tag.
4. Detected Corners using cv2.goodFeaturesToTrack and selected four extreme corners representing the corners of the AR Tag.
5. Assumed four world coordinates as [**minx, miny**] ,[**maxx, miny**] ,[**minx, maxy**] [**maxx, maxy**] out of the corner of the AR tag.
6. Did compute the Homography matrix $\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$ using **A** matrix. The Homography matrix is eigen vector corresponding to the least eigen value.
7. Computed homography matrix multiplication on every pixel from the AR tag to get new coordinates in the world space. Filled colors of the respective image space pixel to a world space pizel.
8. After finding the isolated AR tag, divided the image into 8 parts. Checked the value of the pixels corresponding to the coordinates [3.5d, 3.5d] ,[4.5d, 3.5d] ,[4.5d, 4.5d] [3.5d, 4.5d] in clockwise order. If the value is 255, cosidered it as 1 else 0.
9. If a value of 7 is found by using bit operation counter-clockwise, then the orientation is correct, otherwise rotate the AR Tag till the bit operation of 13 is found.

Figure 6: Id Detected.

3 Superimposing image onto Tag

After detecting the AR Tag value, The following steps were followed to superimpose the Terp image onto the AR tag.

1. Rotate the terp image based on the number of rotations required to detect the AR tag.

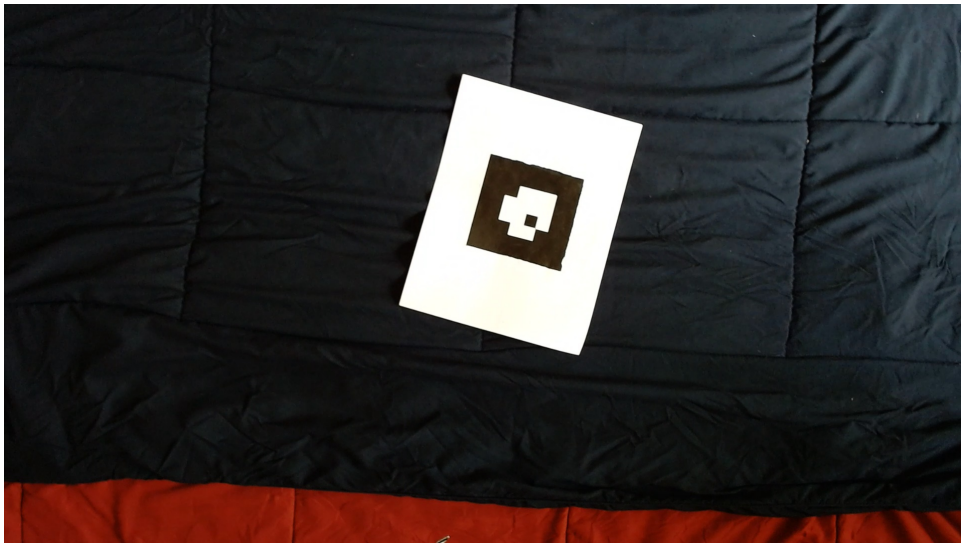

Figure 7: Original Image.

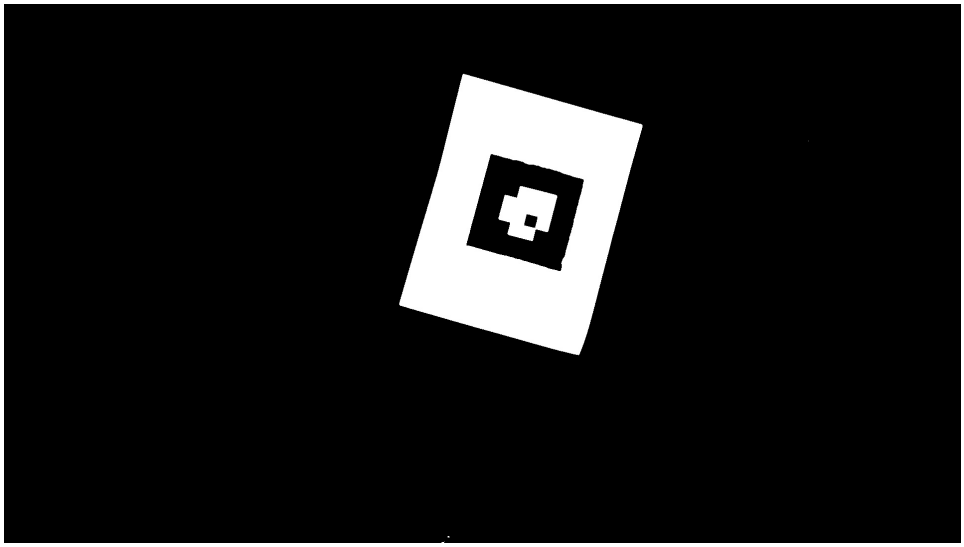

Figure 8: Thresholded Image

Figure 9: Flood fill algorithm to isolate the AR tag.

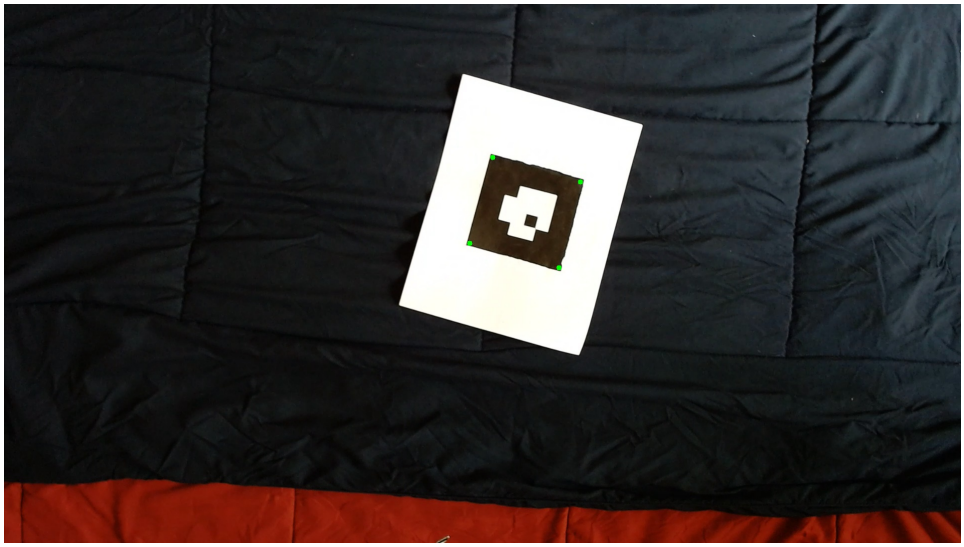

Figure 10: Corners Detected.

Figure 11: AR Tag detected.

2. Resized the terpImage to the AR tag image.
3. Used inverse homography to find image coordinates corresponding to world coordinates and filled the pixels with color of the terpImage pixels
4. Used the above process on all the frames of the video.

Figure 12: Superimposed Terp image.

4 Placing a virtual cube onto Tag

After detecting the AR Tag value, The following steps were followed to place the virtual cube on the tag.

1. Used the intrensic matrix $\mathbf{K} = \begin{bmatrix} 1346.100595 & 0 & 932.1633975 \\ 0 & 1355.933136 & 654.8986796 \\ 0 & 0 & 1 \end{bmatrix}$.
and Homography matrix $\mathbf{H}$ to compute the matrix $\mathbf{P} = \begin{bmatrix} r1_{00} & r2_{01} & r3_{02} & t_{02} \\ r1_{10} & r2_{11} & r3_{12} & t_{12} \\ r1_{20} & r2_{21} & r3_{22} & t_{22} \end{bmatrix}$.
such that

$$\mathbf{B} = lambda * K^{-1} * H$$

and

$$\mathbf{lambda} = [(||K^{-1} * h1|| + ||K^{-1} * h2||)/2]^{-1}$$

Here,

$$\mathbf{B} = \begin{bmatrix} b1 & b2 & b3 \end{bmatrix}.$$

and

$$\mathbf{r1} = lambda * b1,\ \mathbf{r2} = lambda * b2,\ \mathbf{r3} = r1 \times r2,\ \mathbf{r3} = lambda * b3$$

2. Multiplied the P matrix with the four world coordinates (height is taken to be -200)

3. Created a cube out of the corners of the AR tag and the new coordinates found.

4. Used the above process on all the frames of the video.

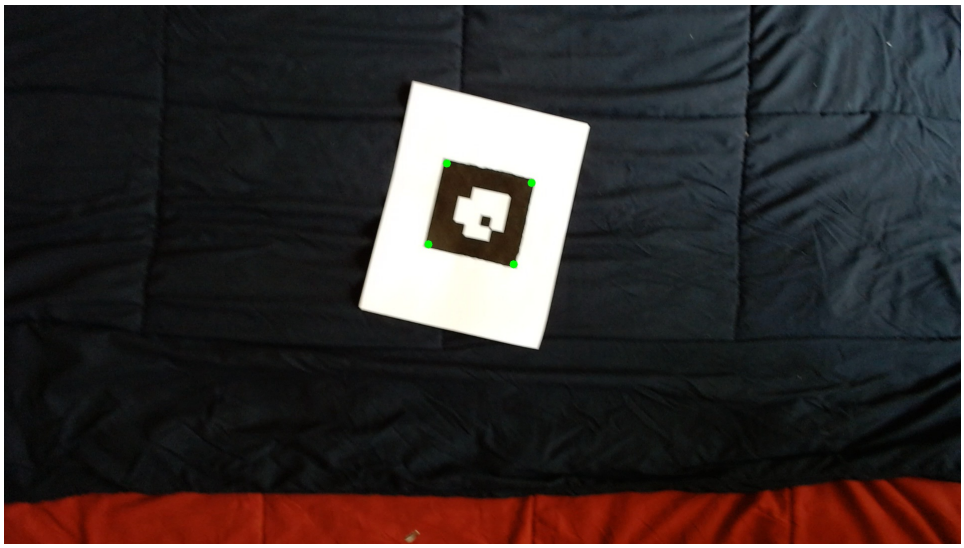

Figure 13: Corners detected.

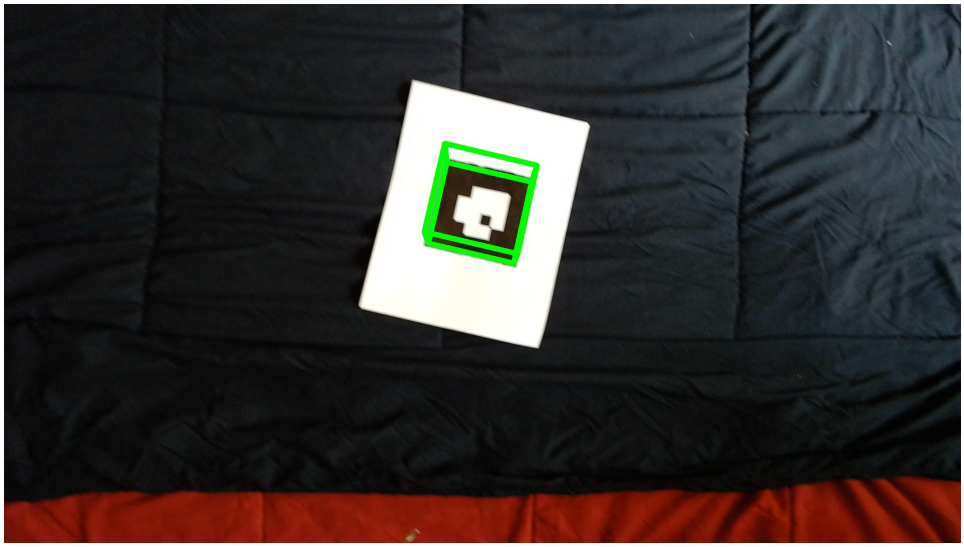

Figure 14: Cube superimposed on the AR tag.