ENPM673 Project1 Report

Bhargav Kumar Soothram
bsoothra@umd.edu

7 March 2022

1 Problem-1: Detection

1.1 AR code detection

This part of the project deals with identifying the AR tag in the scene. To do this, a random frame is selected from the video provided and the corresponding image vectors are stored. We then apply Fast Fourier Transform on the image and filter only the high-frequency components. This filtering can be done by shifting all the low-frequency components to the centre and then applying a circular mask of sufficiently large radius at the centre. The results can be seen after inverting the Fourier transform. The figure below shows all the stages in the pipeline mentioned above.

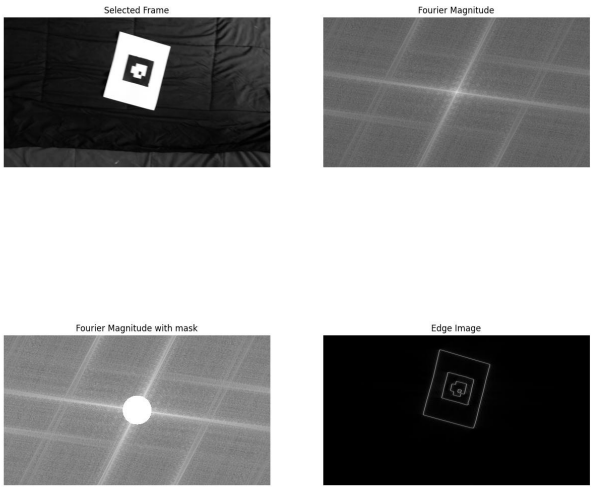

Figure 1: Illustration: Stages in the pipeline

1.2 Decoding the custom AR tag

AR tags are fiducial markers that support augmented reality. They allow for video tracking capabilities that calculate a camera's position/orientation relative to physical markers in real time and can be used to facilitate the appearance of virtual objects the real world, which is what we will do in the Problem 2.

The tag is first divided into a an 8x8 grid where each grid element contains equal number of image pixels. Then, the average of all the intensity values at each grid element in the inner 4x4 grid is calculated and the grid element value is determined accordingly. Upon doing this, we find the orientation of the tag.

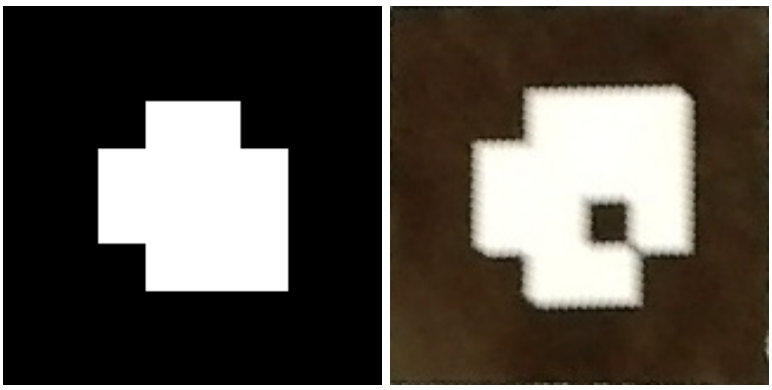

Figure 2: Given reference tag and warped reference tag

For example, the tag on the left in Figure 2 denotes the upright orientation and the tag shown in the right side of figure 2 (obtained by warping an image frame) is rotated by 270 degrees.

2 Problem 2: Tracking

2.1 Superimposing an image onto the tag

In this section of the project, we were tasked with superimposing an image at the location of the marker in each frame of the video. To do this, we follow the following steps:

- Get the corners of the tag from the scene
- Find the homography between the obtained tag corners and the upright (ideal) tag corners
- Warp the image into the global coordinate system
- Find the orientation by decoding the AR code in the warped image and rotate the corners of the image accordingly
- Determine the pixels in the frame that need to be replaced using inverse homography and replace them with those of the desired image.

The results are shown in Figure 3.

Figure 3: Actual frame (left), Superimposed frame (right)

The major issue here is the corner detection itself. For the purposes of this project, the image was first blurred using median blur - to remove the noise and then the Shi-Tomasi corner detection algorithm was employed to detect the corners.

Although my implementation provided with some decent results, I feel that the parameters could use a bit more tuning to get more noise out of the way and better detect corners.

2.2 Placing a virtual cube onto the tag

In this section, we attempt to place a cube onto the tag. This process is similar to that of 2.1, and the pipeline can be inherited until the homography matrix is obtained. We know that the cube is a 3-dimensional quantity and requires a projection matrix to project its vertices onto the image plane (2-dimensional).

For this, we add a z-component to all our planar coordinates and this changes our homography matrix. The projection matrix P can be obtained using the homography matrix H and the camera calibration matrix K (given in prior).

The results are shown below:

Figure 4: Cube placed on the AR tags

The problems here are similar to the ones in 2.1. Since the corners in a few frames do not get detected properly, the cube gets distorted. I have tried many things tuning the parameters, like applying a series of filters to better smoothen the image and changing the Shi-Tomasi parameters, but the results did not change much. Overall, the pipeline could use a bit more tuning.

3 Video links

- Testudo image on AR tag
- Cube on AR tag