Project 1 Report

March 8, 2022

Student:
Vishwanath Datla
UID:117705928

Semester:
Spring 2022

Course code:
ENPM673

Contents

1 Problem 1

1.1 1a

Detection and decoding of AR tag. Detection technique implemented is fast fourier transform and corner detection technique implemented is Harris corner detector (cv2.cornerHarris(...))
The np.fft.fftshift function shifts the zero frequency component to the centre of the spectrum. A circular mask was designed to help with fourier filtering.
The output when an image of the AR tag from the video is input to the code is as follows:

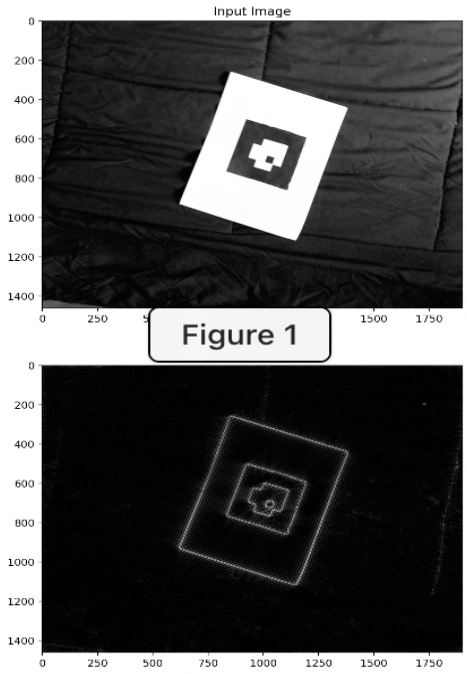

Figure 1: FFT Detection

When Harris corner detection was implemented and the corners were highlighted on the image, the output was as follows: The red and green dots on the image are the corners that were detected.

1.2 1b

Decoding the AR tag:
The white square in the corner of the outer layer indicates the upright position of the tag. The 4 squares in the middle of the tag determine the value of the tag (Binary encoding)
In order to decode the tag, the orientation should be determined first because the tag is rotating in the video frame. After finding out the correct orientation of the tag, the value was computed. Homography matrix is used to map between two planes, the image plane and the physical plane. It can be computed as follows:

```
def computeHomography(c1, c2):
    if (len(c1) < 4) or (len(c2) < 4):
        print("Need atleast four points to compute SVD.")
```

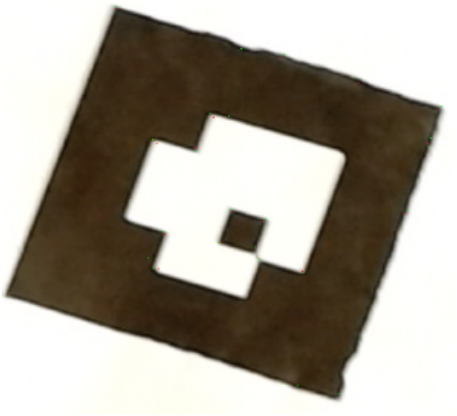

Figure 2: Corner detection

```
        return 0
    x = c1[:, 0]
    y = c1[:, 1]
    xp = c2[:, 0]
    yp = c2[:,1]
    nrows = 8
    ncols = 9
    A = []
    for i in range(int(nrows/2)):
        row1 = np.array([-x[i], -y[i], -1, 0, 0, 0, x[i]*xp[i], y[i]*xp[i], xp[i]])
        A.append(row1)
        row2 = np.array([0, 0, 0, -x[i], -y[i], -1, x[i]*yp[i], y[i]*yp[i], yp[i]])
        A.append(row2)
    A = np.array(A)
    U, E, VT = np.linalg.svd(A)
    V = VT.transpose()
    H_vertical = V[:, V.shape[1] - 1]
    H = H_vertical.reshape([3,3])
    H = H / H[2,2]
    return H
```

In order to align the tag correctly to accurately decode it, a warp_perspective function was used. After finding the homography from source coordinates to warped coordinates, warping was performed by multiplying the coordinates with the homography matrix. Due to the scaling of the image coordinates =, the warped image can develop blank pixels sometimes.

```
def warp_perspective(image, h_matrix, dimension):

    warped = np.zeros((dimension[0], dimension[1], 3))
    for index1 in range(0, image.shape[0]):
        for index2 in range(0, image.shape[1]):
            new_vec = np.dot(h_matrix, [index1, index2, 1])
            new_row, new_col, _ = (new_vec / new_vec[2] + 0.4).astype(int)
            if 5 < new_row < (dimension[0] - 5):
                if 5 < new_col < (dimension[1] - 5):
```

```
                warped[new_row, new_col] = image[index1, index2]
                warped[new_row - 1, new_col - 1] = image[index1, index2]
                warped[new_row - 2, new_col - 2] = image[index1, index2]
                warped[new_row - 3, new_col - 3] = image[index1, index2]
                warped[new_row + 1, new_col + 1] = image[index1, index2]
                warped[new_row + 2, new_col + 2] = image[index1, index2]
                warped[new_row + 3, new_col + 3] = image[index1, index2]

    return np.array(warped, dtype=np.uint8)
    #Transpose the image
```

The smallest contour in the image is the small square in the inner layer of the AR tag. The inner grid was then assigned binary values based on the grayscale pixel values so it's easier to calculate the tag value. Using 5 if-elif-else statements, the conditions for bits to be assigned was decided. The tag value was computed by the code to be 7. This is printed in the output along with the frame number when the code is run.

2 Problem2

2.1 2a

The tag_id function returns the inner grid contour and the testudo image can be superimposed on this contour.
Computing homography and warping are the first steps to superimposing the testudo image. After the corners of the testudo image are warped, the image can be added to the original image with pixel values of 0 to superimpose the image. Following is what the tag with testudo superimposed looks like:

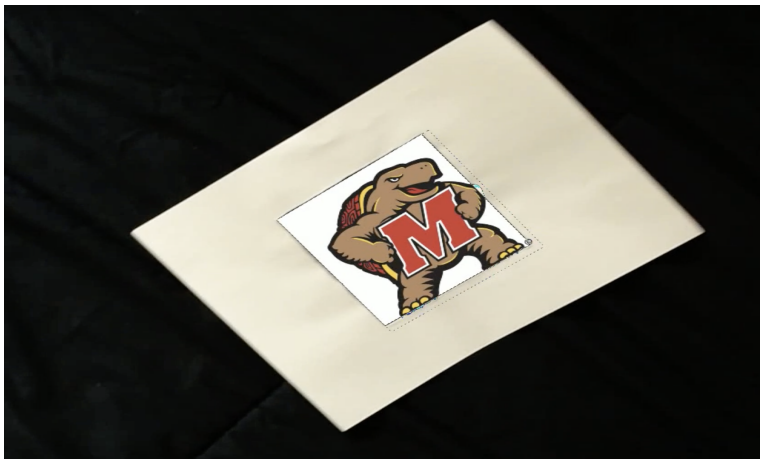

Figure 3: Testudo superimposed

2.2 2b

Since the 4 corners of the AR tag are available, the projection matrix can be used to get the corners in the camera frame. The Cube3D function can be used to render a 3D cube in the 2D image plane.

Following are the links for the videos generated to superimpose testudo and project the cube respectively:

2.2 2b

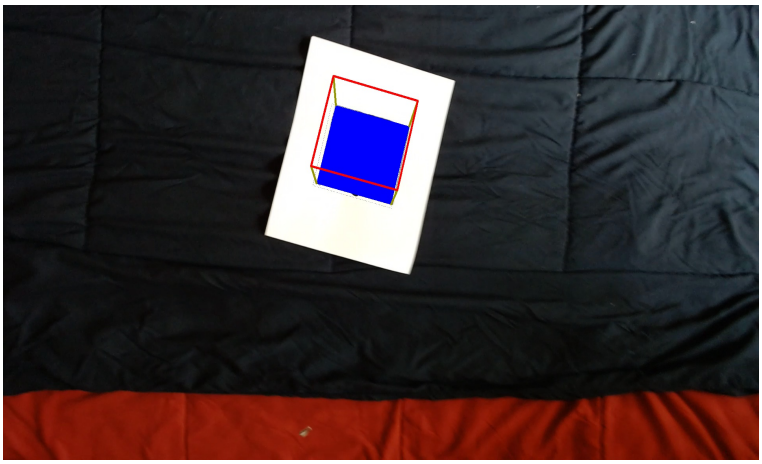

Figure 4: Cube

https://drive.google.com/file/d/12QpqGPV_rsv1dvweGml0l4s5QdhEozgL/view?usp=sharing
https://drive.google.com/file/d/1Gg5FVql1OPApud0sQbj4ISN7MFH4a01O/view?usp=sharing