ENPM673

Perception

Project 1

ADITYA VARADARAJ

(UID: 117054859)

(SECTION: 0101)

M.Eng. Robotics (PMRO),

University of Maryland - College Park, College Park, MD - 20742, USA.

Date of Submission: 7th March, 2022

LIST OF FIGURES

CONTENTS

I. Problem 1 - Detection

A. 1(a) FFT based edge detection of April Tag

- For one of the frames in the video, we first convert to grayscale.
- Then we apply **FFT** using scipy.fft.fft2()
- Then we **shift** the origin to center of plot using scipy.fft.fftShift()
- Then we calculate **magnitude** of the fftShift as $20 \log ||fftShift||$
- Then we create a **High Pass Filter (HPF)** by creating a mask which subtracts the frequencies in a circle of radius 80 pixels (low frequency pixels) from center. Thus, it only allows the high frequencies to pass
- Edges have a high frequency in images. Thus, the resultant frequencies of the High pass filter represent edges.
- Now, we apply **Inverse FFT Shift** using scipy.fft.ifftshift()
- Finally, we apply **Inverse FFT** using scipy.fft.ifft(). The magnitude (norm) ($||ifft||$) of the result of inverse fft gives us a binary image with only edges in white and rest of backround is black.
- We can see all the stages in the result of the code shown in Fig. 1

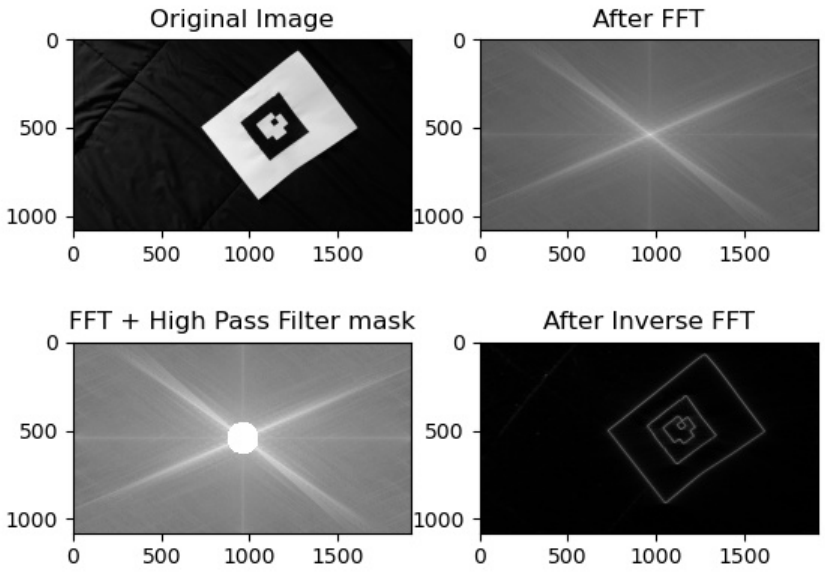

Fig. 1. Various stages of FFT, HPF and IFFT to get edges

B. 1(b) Decoding Reference AR Tag

Decoding Process:

- From the tag corners detection, homography and warping pipeline, we get a binary image of the AR tag of size 128×128 in case of Problem 2 and in 1(b), we perform decoding on the binary reference AR tag image provided resized to 128×128.

- This image can be divided into 8×8 grid with each block having 16×16 pixels as shown in Fig. 2.

- In this, the inner 2×2 grid at the center encodes the ID of the tag.

- The blocks between the 4×4 grid and the 2×2 grid encode the orientation.

- If the white block in that area is in bottom right, then tag is in upright position.

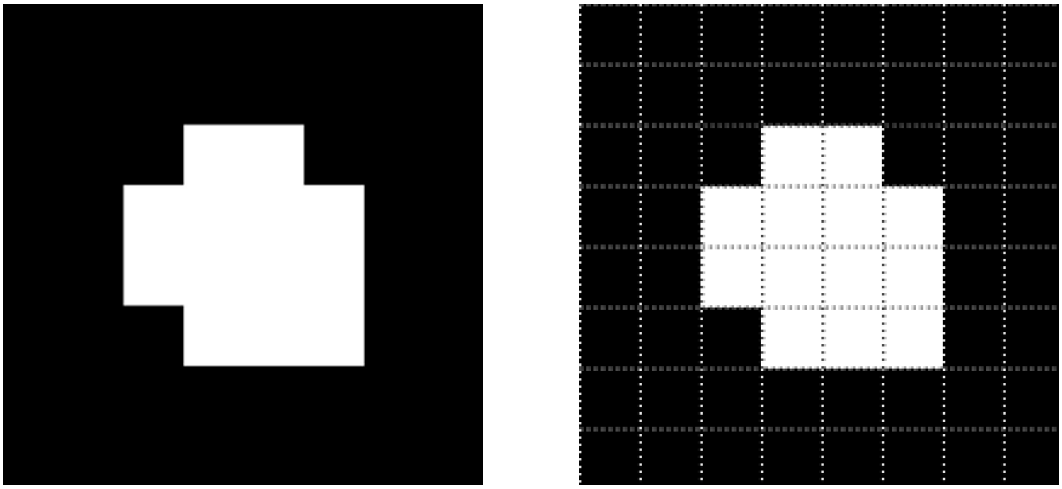

Fig. 2. (a) Reference AR Tag Image and (b) Grid bifurcation of tag for decoding

```
------------ 1b: Decode Reference Tag Image ----------
ID:  15
Orientation:  0
------------ 2a: Decode Frame Tag and superimpose testudo ----------
Orientaton:  1.5707963267948966
ID:  7
```

Fig. 3. Output of Decoding for Reference AR tag image (above) and frame of the video (below)

- The 2×2 grid is a binary representation of the ID in clockwise direction from least significant to most significant bit. In the tag's upright orientation, the top left block in the 2×2 grid is the least significant bit and the bottom left block is the most significant bit. The most significant bit will be multiplied by 2^3 and least significant bit will be multiplied by $2^0 = 1$.

The output of decoding is shown in Fig. 3.

II. Problem 2 - Tracking

A. 2(a) Superimposing Testudo image onto Tag

Pipeline:

- First, I took in each frame of the video and convert it to **grayscale** image.

- The, I used **bilateral filtering** to blur/smooth the image for better corner detection.

- Then, I used **Harris Corner Detection** on the bilateral image. To get the coordinates of the corners from output of cv2.cornerHarris(), I had to do some extra steps, namely, dilation, thresholding, cv2.connectedComponentsWithStats() and cv2.cornerSubPix().

- These points do not come in any specific order. So to **sort them in clockwise order**, I have taken 1st element as the one with minimum $|x + y|$, second element as the one with minimum $|x - y|$, third element as one with maximum $|x + y|$ and fourth point as the one with maximum $|x - y|$.

- We take **4 corners of a** 128×128 **image** and **find homography matrix** between those corners and the detected **tag corners**.

- Then, we **warp (inverse warping)** ($frame\ coords\ =\ H^{-1}\ reference\ image\ coords$) the image and the **copy the pixels from image to the new image** to get a 128×128 warped binary image of the tag detected in the frame. (Straightening the tag).

- Now, we apply **decoding** similar to problem 1(b) to get the ID and orientation of the tag.

- Based on orientation, we decide which corners of testudo image the corners of tag correspond to, i.e., in which order. Then we **find the homography matrix** between the corners of **tag** and **testudo** corners using SVD. (Using Smallest Eigenvector scaled/divided by its last value).

- Then, we apply inverse **warping** to the frame and the testudo image ($frame\ coords\ =\ H^{-1}$ $testudo\ image\ coords$) to get corresponding pixel coordinates and **copy those pixels from testudo image into frame.**

- Using this, we get result as seen in Fig. 4

- **Output video:** https://drive.google.com/file/d/1VS8h-v8H2a8SIFxEMmAc3PoTdHxMtQbN/view?usp=sharing

Fig. 4. Output of part 2(a)

B. 2(b) Placing Virtual Cube on Tag

- Now, for cube, based on orientation of tag found by decoding, we decide in what order we give the 4 corners of the cube of the face that will be on the tag, i.e., whether to give as $[[0,0],[127,0],[127,127],[127,0]]$ or $[[127,0],[127,127],[127,0],[0,0]]$ or something else.

- Then, we find **homography matrix** H between these 4 points of the cube and the 4 corners of tag in the frame of the video.

- Then, we find the **Projection matrix**.

$$
\begin{aligned}
H &= \lambda K[r_1\ r_2\ t] \\
,where,\ H &= [h_1\ h_2\ h_3] \\
By\ solving\ simultaneous\ equations,\ \lambda &= \frac{||K^{-1}h_1|| + ||K^{-1}h_2||}{2} \\
[r_1\ r_2\ t] &= \frac{1}{\lambda}K^{-1}H \\
P &= K[r_1\ r_2\ (r_1 \times r_2)\ t]
\end{aligned}
\tag{1}
$$

- Then, we **pre-multiply projection matrix** P with cube corner coordinates in form $[x\ y\ z\ 1]$ to get $x,\ y\ and\ z$ coordinates of the cube corner in video frame.

- Then, we **scale** x and y as $\frac{x}{z}$ and $\frac{y}{z}$.

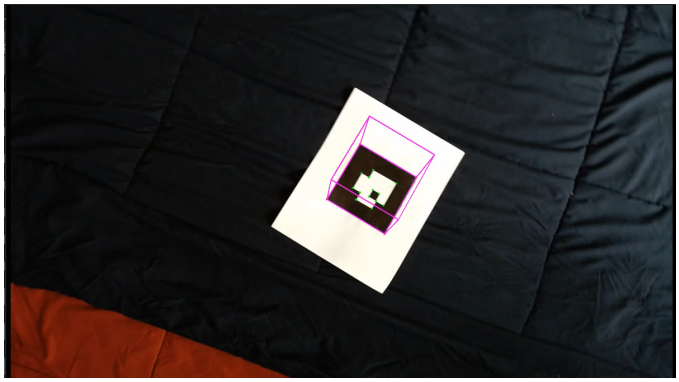

Fig. 5. Output of part 2(b)

- Then we **draw** the 2D lines (,i.e., **projected edges of cube**) on the frame.

- The output frame looks as shown in Fig. 5. **Output Video:** https://drive.google.com/file/d/1dXL-k5LvM3gwg3nyIS4t1nSouRdGymeI/view?usp=sharing