EEE: 515 - Machine Vision - Homework 1

Praveen Paidi
Student ID: 1225713099
Fall 2023

October 10, 2023

1 Problem status

1. **Problem 1 : Completed**

2. **Problem 2 : Completed**

3. **Problem 3 : Completed**

4. **Problem 4 : Completed**
   Completed in the first commit. In this submission, I Corrected the diagonal band pass filter magnitude and filtered image.Provided the same at the end of the document on page 17.

5. **Problem 5 : Completed**

6. **Problem 6 : Completed**
   Completed 6c part and attached at the end of the report on page 16.

2 Problem 2:

A.

Original image is displayed in figure 1. By using plt.imshow, the image is being displayed as shown in figure2.

Figure 1: Original Image

Figure 2: Image display using plt.imshow

Matplotlib expects the image to be in RGB format whereas Opencv expects the image to be in BGR format.The colors seems inverted in figure 2 because I read the image using cv2.imread.Using the listing 1, I inverted the colors and displayed in figure 3. Similarly, Figure 4 displays output using listing 2.

Listing 1: Changing colorspace

```
cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Listing 2: Changing colorspace

```
cv2.cvtColor(image, cv2.COLOR_BGR2GREY)
```

Figure 3: Image display using BGR2RGB

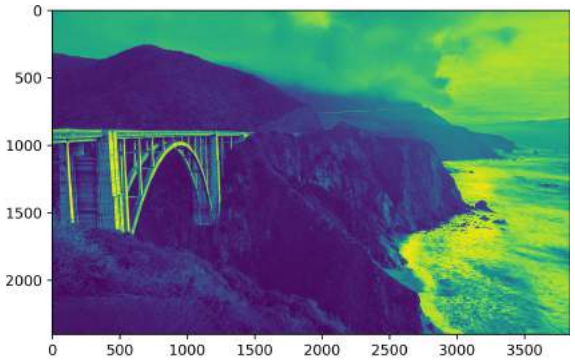


Figure 4: Image display using BGR2GREY

B . Cropping images

1.After cropping the image , I can see the edges of the rocks, color distinction between the weed and the rocks.
2.The colors are same on a whole part but small nuances in color are visible.

3.Although there is color differentiation, the edges of weeds and plants are hazy, which were not visible in the original photograph.
4.Since I kept the dimensions without stretching the image across the same dimensions, the resolution remained the same. I lose resolution in the cropped image if I zoom in more to spread the dimensions.

Figure 5: Image display using BGR2RGB

C. Resizing images
Figure 6 shows Original image, figure 7 shows downsampled image by 10 times using cv2.resize()

Figure 6: Original Image

Figure 7: Downsampled image

Reconstructed the original image by upsampling the downsampled image using following techniques
a.Nearest neighbours in figure 8 by using listing 3
b.Bicubic in figure 9 by using listing 4.

Listing 3: Nearest Neighbors

```
cv2.resize(downsampled, dim, interpolation = cv2.INTER_NEAREST)
```

Listing 4: BICUBIC

```
cv2.resize(downsampled, dim, interpolation = cv2.INTER_CUBIC)
```

Figure 8: Nearest neighbors upsampled image

Figure 9: Bicubic Umsampled image

The Difference between the sum of pixels of original image and bicubic reconstructed upsampled picture is 3020544906, whereas original image and Nearest neighbor reconstructed upsampled picture is 2980045591. Evidently Nearest neighbors method upsampled picture is better than bicubic upsampled picture by considering the error.

Fig 10 , 11 shows difference images of both methods.

Figure 10: Nearest neighbors difference image

Figure 11: Bicubic difference image

By observing figures, Bicubic is making picture more smoother than other one.It can be used in cases where less gradient, edges in images are present and smooth color transition is needed.It helps to maintain the quality of the images.Applications where interpolation is required.

Nearest neighbors is used for precise data information in image such as to find the hard edges in the image, boundaries.Applications such as finding edge coordinates in images.

3 Problem 3:

Kernel is zero padded, then teoplitz method is used to make double block teoplitz to convert the kernel into H matriz of shape 25 X 9. Image is vectorized and then mutliplied to H matrix.

Hand written calcuation is attached at the end the report.

```
[[-1  0  0  0  0  0  0  0  0]
 [ 0 -1  0  0  0  0  0  0  0]
 [ 1  0 -1  0  0  0  0  0  0]
 [ 0  1  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0]
 [-1  0  0 -1  0  0  0  0  0]
 [ 0 -1  0  0 -1  0  0  0  0]
 [ 1  0 -1  1  0 -1  0  0  0]
 [ 0  1  0  0  1  0  0  0  0]
 [ 0  0  1  0  0  1  0  0  0]
 [-1  0  0 -1  0  0 -1  0  0]
 [ 0 -1  0  0 -1  0  0 -1  0]
 [ 1  0 -1  1  0 -1  1  0 -1]
 [ 0  1  0  0  1  0  0  1  0]
 [ 0  0  1  0  0  1  0  0  1]
 [ 0  0  0 -1  0  0 -1  0  0]
 [ 0  0  0  0 -1  0  0 -1  0]
 [ 0  0  0  1  0 -1  1  0 -1]
 [ 0  0  0  0  1  0  0  1  0]
 [ 0  0  0  0  0  1  0  0  1]
 [ 0  0  0  0  0  0 -1  0  0]
 [ 0  0  0  0  0  0  0 -1  0]
 [ 0  0  0  0  0  0  1  0 -1]
 [ 0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  1]]
```

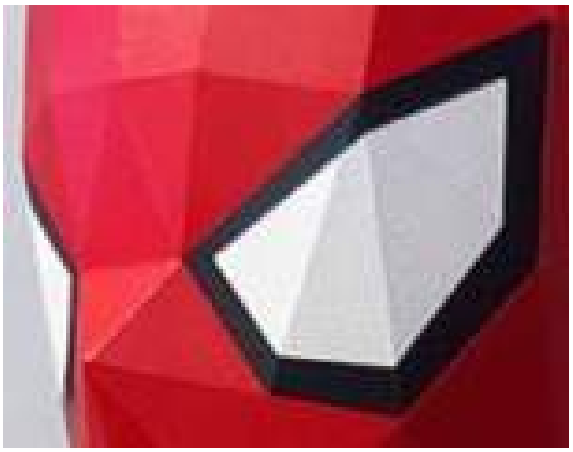
Figure 12: Image Padded

```
Convolve image is
 [[  1   2   2  -2  -3]
 [  5   7   4  -7  -9]
 [ 12  15   6 -15 -18]
 [ 11  13   4 -13 -15]
 [  7   8   2  -8  -9]]
```

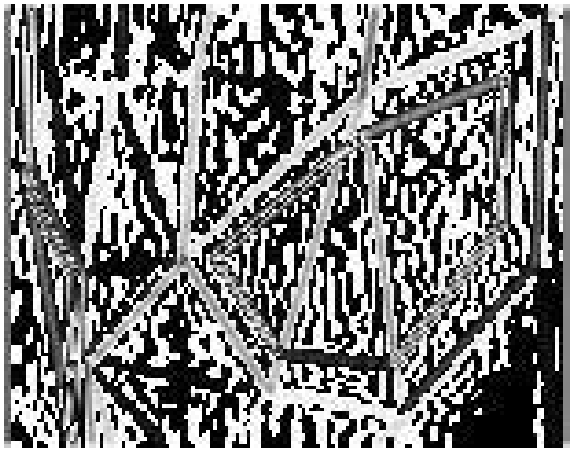
Figure 13: Output convolved image

Functionconv2d takes in vectorized image 9x1 and H matrix of 25x9 size.
Function outputs convolved image, time of calculation ad error.
H matrix is created by using toeplitz matrix method then doubly blocking it.
Scipy function of toeplitz is used to make matrices then stacked them using loops.

c . For generalization I have taken a picture and passed through the function to following output.



Arbitary Image



Edge filtered by Generalized code

This is giving an output of 0.126173019409180 ms as the time Error being nearly 3Some of the code snippets.

Listing 5: Changing colorspace

```
toeplitz_m=scipy.linalg.toeplitz(c,r)

for j in range(doubly_indices.shape[1]):
        start_i = i*b_h
        start_j = j*b_w
        end_i = start_i+b_h
        end_j = start_j+b_w
        doubly_blocked[start_i:end_i, start_j:end_j]= toeplitz_list[doubly_indices[
            i,j]-1]
```

4 Problem 4:

Fourier Domain

The original image taken for plotting the fourier domain is shown in figure 14. Image is converted into grey scale, and then into frequency domain by fourier transform and then to magnitude and phase. The grey scale image , magnitude and phase of image are shown respectively in fig 15, fig 16, fig 17. Some of code snippets are shown in listing 6.

Figure 14: Figure with Text Beside

Listing 6: Fourier

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
f = np.fft.fft2(gray)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log1p(np.abs(fshift))
phase_spectrum = np.angle(fshift)
```

Figure 15: Grey scale image of original

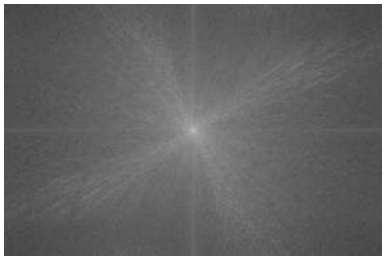


Figure 16: Magnitude of grey scale image

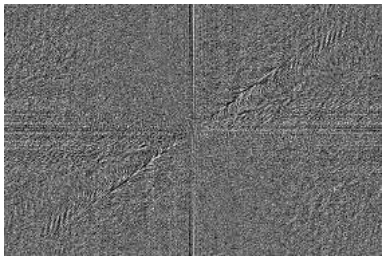


Figure 17: Phase of grey scale image

Magnitude spectrum of image in frequency domain after being lo pass filtered, high pass filtered , Diagonal filtered in fig 18, 19, 20 respectively.

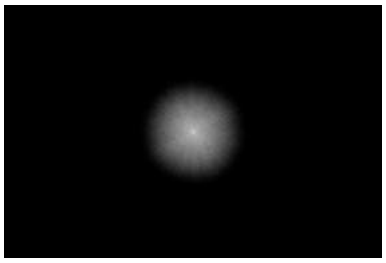


Figure 18: Magnitude of original Low pass filter

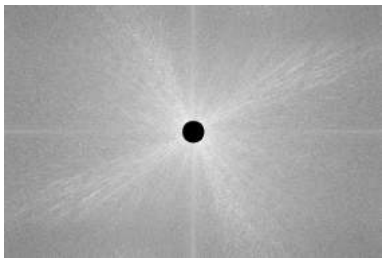


Figure 19: Magnitude of High pass filter

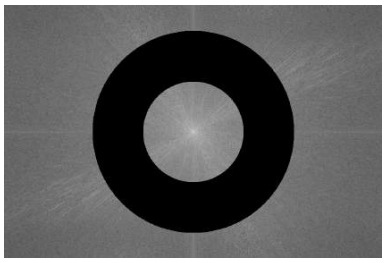


Figure 20: Magnitude of Diagonal pass filter

Images retrieved by performing inverse fourier transform after low pass , high pass, diagonal pass filters are shown in fig 21, 22, 23 respectively using listing 6.

Listing 7: IFFT

```
f_transform_inverse = np.fft.ifftshift(f_transform)
img_back = np.fft.ifft2(f_transform_inverse)
```

Figure 21: Low pass filter grey scale image

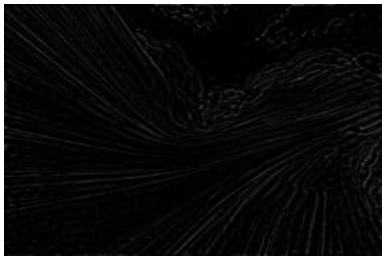


Figure 22: High pass filter grey scale image

Figure 23: Diagonal band pass grey scale image

C. Phase swapping of images.
Original images selected are fig 24 and fig 25 shown along with their respective magnitude and the phase.

Figure 24: IMAGE 1

Figure 25: IMAGE 1 Magnitude

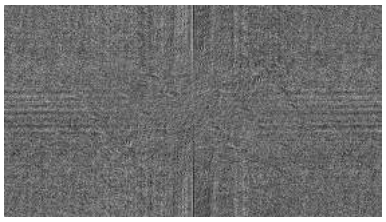


Figure 26: IMAGE 1 Phase

Figure 27: IMAGE 2

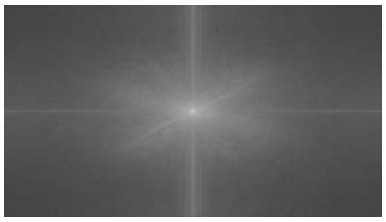


Figure 28: IMAGE 2 Magnitude

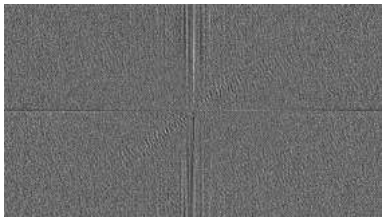


Figure 29: IMAGE 2 Phase

Reconstructed images after swapping the fig 24 and fig 27 phases swapped are shown in fig 30 and fig 31:

Figure 30: IMAGE 1 Reconstructed

Figure 31: IMAGE 2 Reconstructed

The modified images after amplifying phase by multiplying with constant results in fig 32 and fig 33. I multiplied with number more than 0 which causing edges and some darker information is being lost.Changing the phase result in losing information in repect to sharpness, amplitudes, spatial setup.

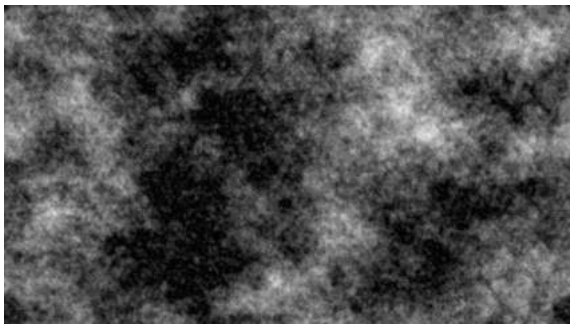


Figure 32: Modified IMAGE 1

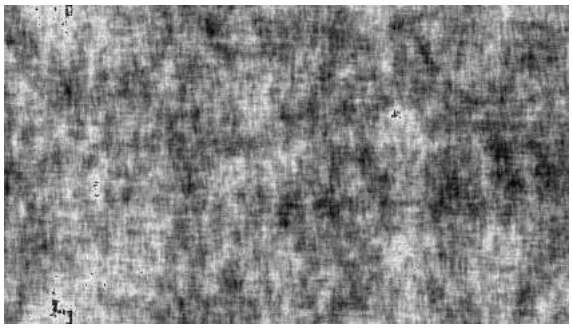


Figure 33: Modified IMAGE 2

Hybrid image technique : The two images selected are shown in fig 34 , fig 36 with magnitudes.

Figure 34: IMAGE 1

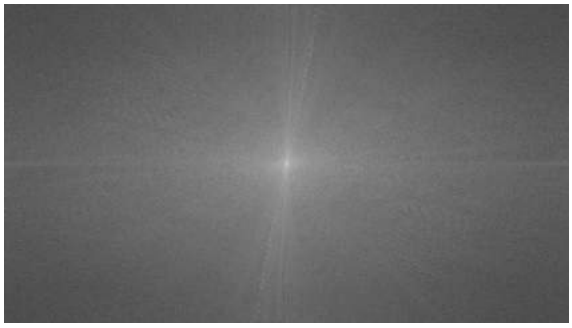


Figure 35: Magnitude IMAGE 1

Figure 36: IMAGE 2

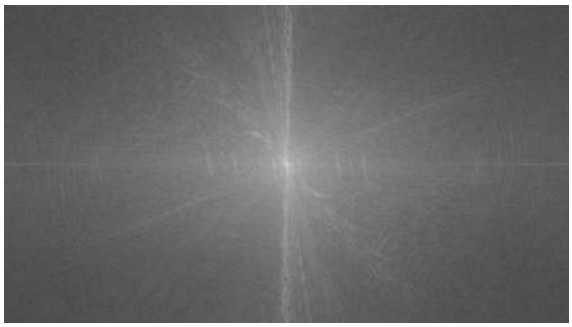


Figure 37: Magnitude IMAGE 2

The images are converted into frequency domain and then being added up as stated in hybrid image approach and performed inverse fourier transform to get the grey scale image in spatial domain. From near Fig 36 appears, from far fig 34 appears clearly.

Figure 38: Hybrid Image

5 Problem 5:

Figure 39 and Figure 40 are the 2 images being selected for blending. I selected these images as faces because there will be lot features to match such as skin color, hair texture and low level facial features at the edge.

Reconstructed image from by collapsing the laplacian pyramids as shown in figure 41 and figure 42.

Figure 39: IMAGE of face 1

Figure 40: IMAGE of face 2

Figure 41: Reconstructed face 1 using Laplacian pyramid

Figure 42: Reconstructed face 2 using Laplacian pyramid

Display of pyrmaid images at same size though dimensions got reduced by half follows:
Figure 43 shows fig 40 gaussian pyramid using plt.imshow

Figure 43: Gaussian of fig 40

Figure 44 shows fig 40 laplacian pyramid using plt.imshow

Figure 44: Laplacian of fig 40

Figure 45 shows fig 39 gaussian pyramid using plt.imshow

Figure 45: Gaussian of fig 39

Figure 46 shows fig 39 laplacian pyramid using plt.imshow

Figure 46: Laplacian of fig 39

Figure 47 shows faces being added using direct blending,by preparing a binary mask and adding at the centre. This blending is visually not appealing as there is sharp change in the features at the edge and stitch line is visible.

Figure 48 shows faces being blend using alpha blending by gaussian blurring the edges of two faces and multiplied using the binary masks.

Figure 49 shows faces blend with multiblend resolution using laplacian pyramids.

The images are blend better in multi resolution due to better color blending and smoothening gradient better than others by keeping important details in image.

Figure 47: Direct blending

Figure 48: Alpha blending

Figure 49: Multiresolution blending

Listing 8: Changing colorspace

```
# Multi resolution

for i in range(5):
    f = cv2.pyrDown(f)
    g = cv2.pyrDown(g)
    gpA1.append(f)
    gpA2.append(g)

for i in range(5,0,-1):
    GE = cv2.pyrUp(gpA1[i])
    L = cv2.subtract(gpA1[i-1],GE)
    lpA1.append(L)

    GE = cv2.pyrUp(gpA2[i])
    L = cv2.subtract(gpA2[i-1],GE)
    lpA2.append(L)


# Direct blending

mask1 = cv2.rectangle(mask1, (0, 384), (176, 0), (255,255,255), thickness=-1)

mask2 = cv2.rectangle(mask2, (177, 384), (384, 0), (255,255,255), thickness=-1)

result1 = cv2.bitwise_and(image, mask1)

result2 = cv2.bitwise_and(image2, mask2)


# Alpha blending

result1[0:376, blur_start_value:blur_end_value] = a[0:376, blur_start_value:
    blur_end_value]
result2[0:376, blur_start_value:blur_end_value] = b[0:376, blur_start_value:
    blur_end_value]

final= result1+result2

# adding blurred masks
```

6 Problem 6:

I chose 2 frames as reference frames.
A.Highest intensity frequency frame.
B.Average of first 50 frames.

I subtracted the video and amplified the resulting frames by performing division with aperture size to get better results between by both A and B frames.

The final best output is shown in figure in comparison to background image.This is obtained by choosing the Method A in this case.

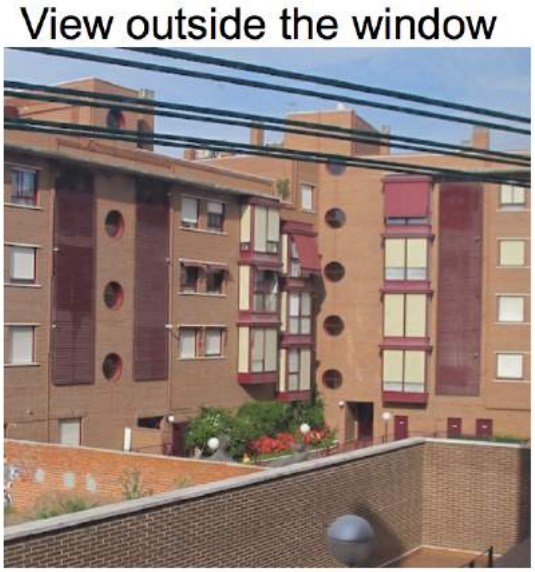


Figure 50: Outside view

Figure 51: Image obtained

Imitating pinspeck camera,figure 52 is view outside of apartment and figure 53 is image formed after processing the video.

Figure 52: Outside view

Figure 53: Image obtained

(3) Image = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ Kernel = $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$

Kernel padded = $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \end{bmatrix}$.

As generally → (Image size + Kernel size − 1) ⇒ Kernel padded size

Using Toeplitz construction.

for each row in Kernel padded

Rule of Toeplitz is constant in diagnol and Subdiagnol.

Toeplitz matrix results in building f_0, f_1, f_2, f_3, f_4.

for 1st row suppose $[0\ 0\ 0\ 0\ 0] \rightarrow f_4$

Transposing and Constructing:-

$f_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{5\times3}$ → Column Size should be equal to input size

and last row suppose $[1\ 0\ -1\ 0\ 0] \rightarrow f_0$

$f_0 = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ -1 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

Similary every f_0, f_1, f_2, f_3, f_4 will be 5×3

$f_1 = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ -1 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ $f_3 = \begin{bmatrix} 0 & & \\ 0 & 0 & \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $f_2 = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ -1 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

Figure 54: Image obtained

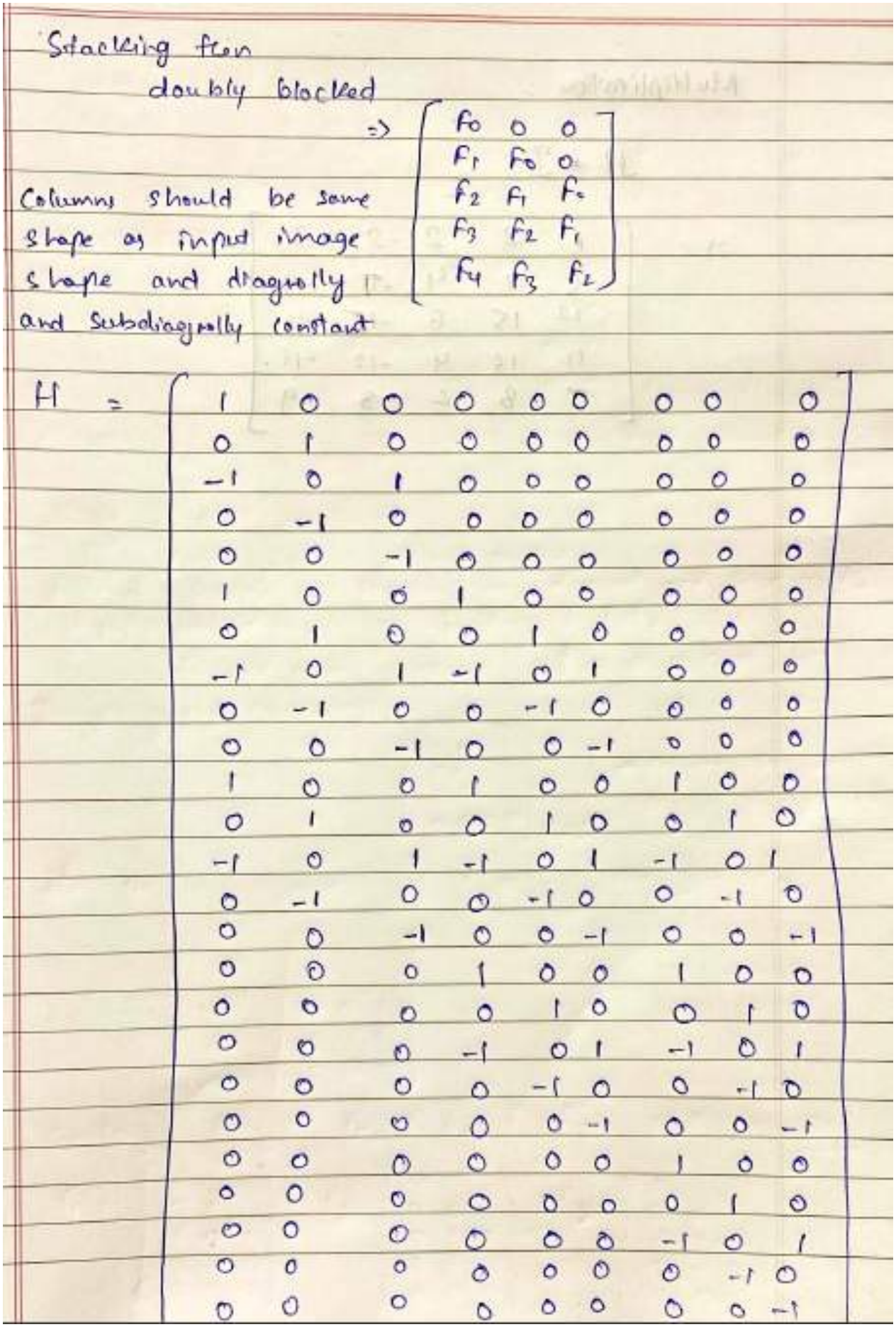


Figure 55: Image obtained

Multiplication :-

$\underline{A} * \underline{I}$

$$\Rightarrow \begin{bmatrix} 1 & 2 & 2 & -2 & -3 \\ 5 & 7 & 4 & -7 & -9 \\ 12 & 15 & 6 & -15 & -18 \\ 11 & 13 & 4 & -13 & -15 \\ 7 & 8 & 2 & -8 & -9 \end{bmatrix}$$

7 Problem 6c part extended :

I chose highest intensity frame as the frame to be subtracted from the all other frames and printed out the difference video.It is smoother and has less noise than provided.

The following are some of the pictures showing original picture, subtracted video provided and my subtracted video.

Figure 57: Original video frame

Figure 58: Given subtraction video frame

Figure 59: My subtraction video frame

Figure 60: Original video frame

Figure 61: Given subtraction video frame

Figure 62: My subtraction video frame

Figure 63: Original video frame

Figure 64: Given subtraction video frame

Figure 65: My subtraction video frame

Figure 66: Original video frame

Figure 67: Given subtraction video frame

Figure 68: My subtraction video frame

Listing 9: highest intensity frame

```
highest_intensity_frame_index = intensities.index(max(intensities))
highest_intensity_frame = frames[highest_intensity_frame_index]
cv2.imwrite('average_frame.png', highest_intensity_frame )
```

8 Problem 4 part band pass filter:

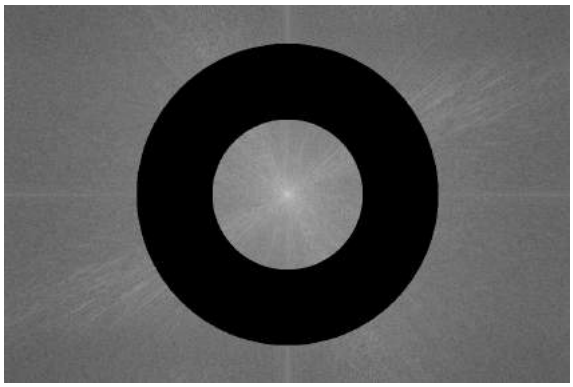


Figure 69: Magnitude of band pass filter

Figure 70: Grey image after Band pass filtering

Supporting code for the band pass filter.

Listing 10: Band pass

```
def apply_annulus_filter(image, center, inner_radius, outer_radius):
    rows, cols = image.shape
    crow, ccol = int(rows * center), int(cols * center)

    # Apply the Fourier Transform
    f = np.fft.fft2(image)
    fshift = np.fft.fftshift(f)

    # Create a mask with low values inside the annulus ring
    mask = np.ones((rows, cols), np.uint8)
    y, x = np.ogrid[-crow:rows-crow, -ccol:cols-ccol]
    mask_area = x*x + y*y >= (int(rows*inner_radius))**2
    mask_area &= x*x + y*y <= (int(rows*outer_radius))**2
    mask[mask_area] = 0

    # Apply the mask
    fshift = fshift * mask

    magnitude_spectrum = 20*np.log1p(np.abs(fshift))

    magnitude_spectrum = ((magnitude_spectrum - np.min(magnitude_spectrum)) / (np.
        max(magnitude_spectrum) - np.min(magnitude_spectrum)) * 255).astype(np.
        uint8)

    cv2.imwrite("MagnitudeDiag" + ".jpg", magnitude_spectrum)

    # Inverse Fourier Transform
    f_ishift = np.fft.ifftshift(fshift)
    img_back = np.fft.ifft2(f_ishift)
    img_back = np.abs(img_back)

    return img_back.astype(np.uint8)
```