Project 1

0410886 何哲佑

Introduction

In this project we are applying our basic knowledge on image processing in order to process distinct kinds of image with different types of alterations. In the first part of the project we'll see how does an image responds to different types of transformations. Secondly, we'll band-limit the image to a certain region in its frequency domain. Finally, we will practice noise removal techniques on different images with different sorts of distortions.

Part 1

For starting we present our image to be processed:

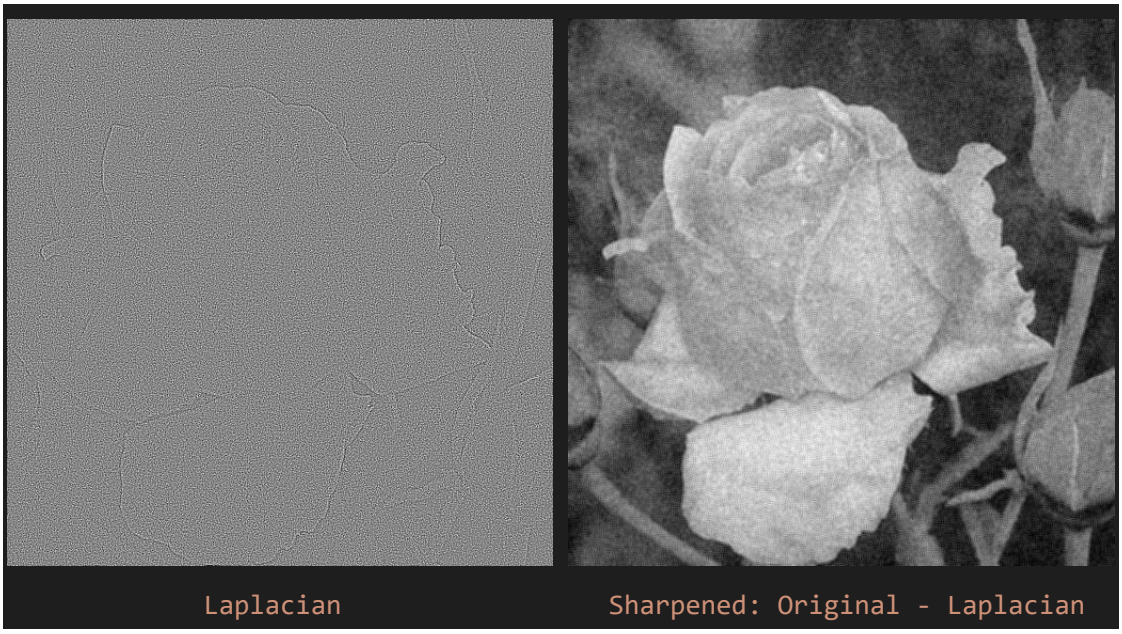
Following this we are going to take the Laplacian of the image and use it to sharpen our image. The Laplacian kernel we are applying is the following:

−1	−1	−1
−1	8	−1
−1	−1	−1

This is going to be convolved with our original image in order to obtain our Laplacian domain result. Which is equal as performing the following gradient on image "I".

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

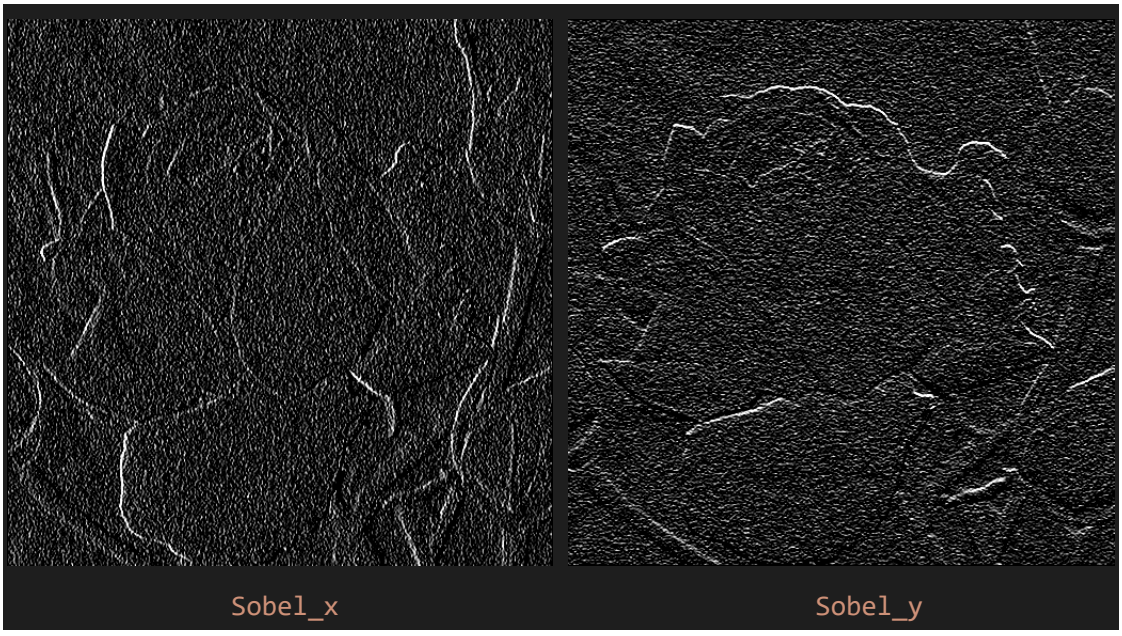
Laplacian will highlight the borders of our image. After finishing our Laplacian calculations we are gonna substract our Laplace domain image from our original image in order to obtain a high boost filter.



Laplacian Sharpened: Original - Laplacian

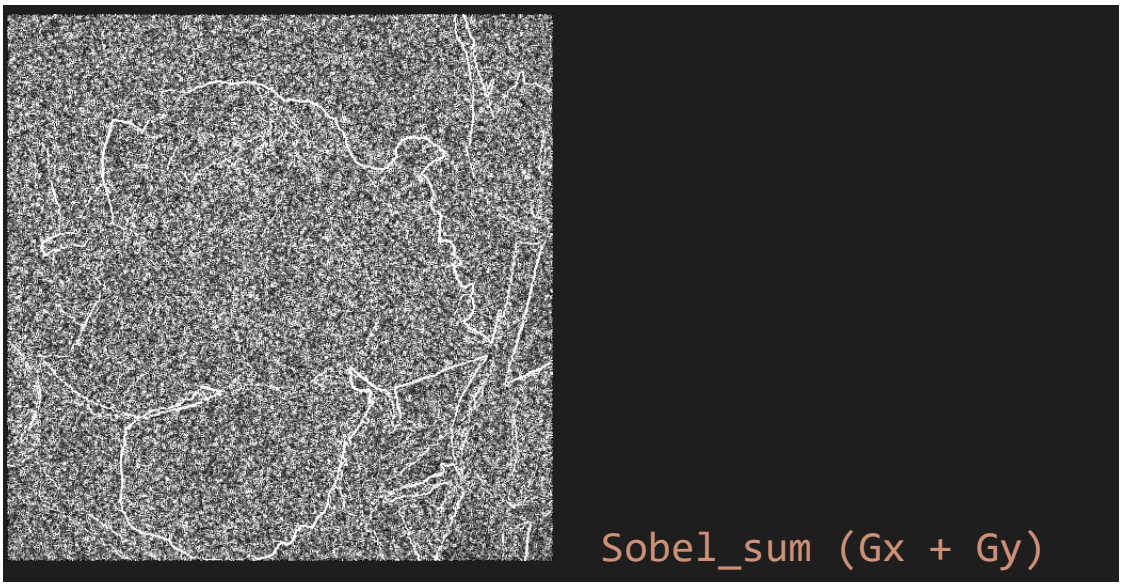
Next, we are going to apply a Sobel gradient to our original image. An operator that is designed to for edge detection. We take separately Sobel in X direction and Sobel in Y direction. After getting each, we add them so that we end up with our final product of Sobel gradient in X and Y direction to our image A.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

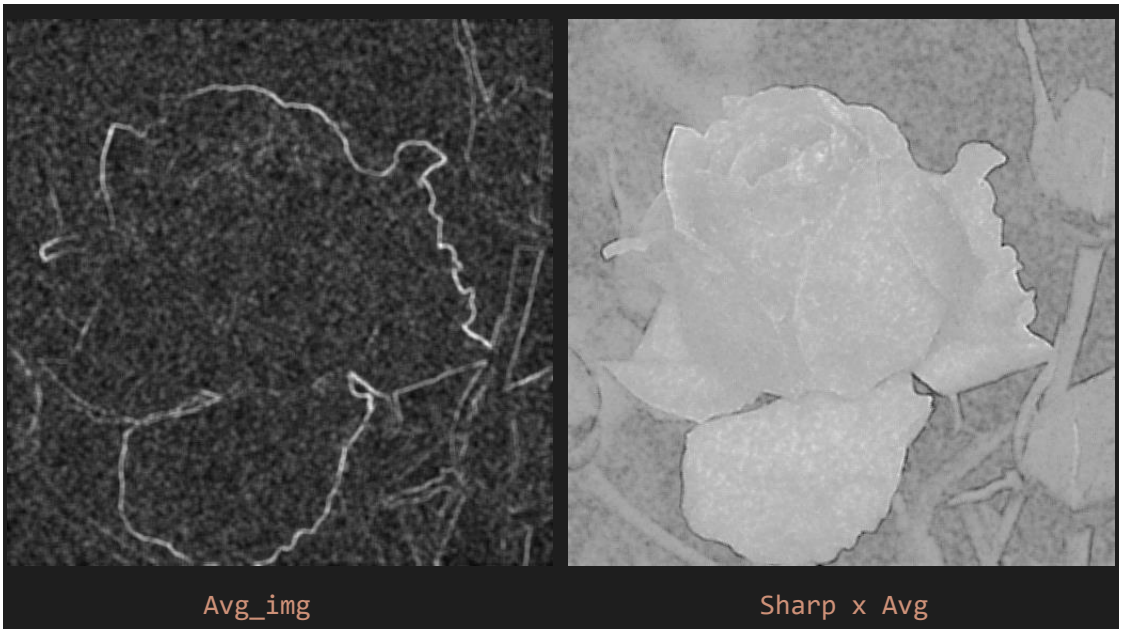


Sobel_x Sobel_y

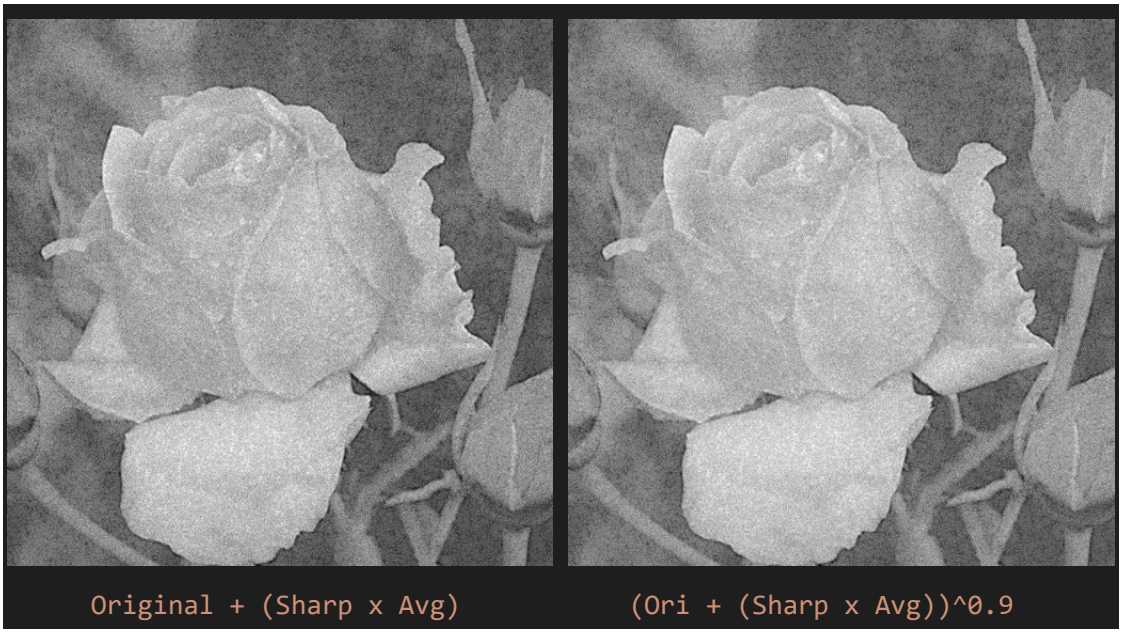
Now the result of the sum:



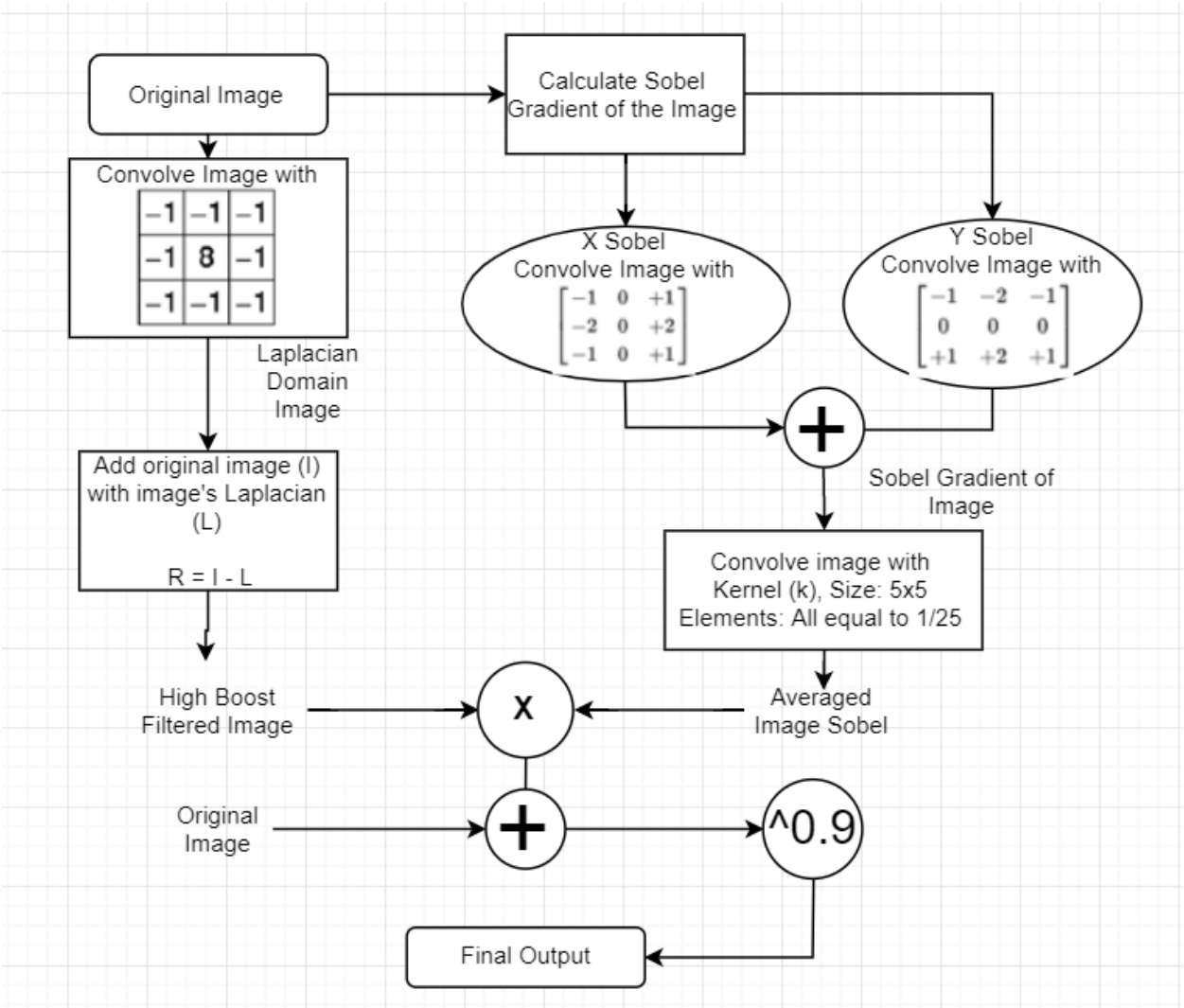
Now we average the Sobel image with a 5x5 kernel. Then we multiply our sharpened image with our averaged image.



For the last 2 transformations we will add our original image to the previous result (Sharp x Avg). Lastly, we will take the power of this result, we use 0.9 as power.



Flow Diagram of part 1



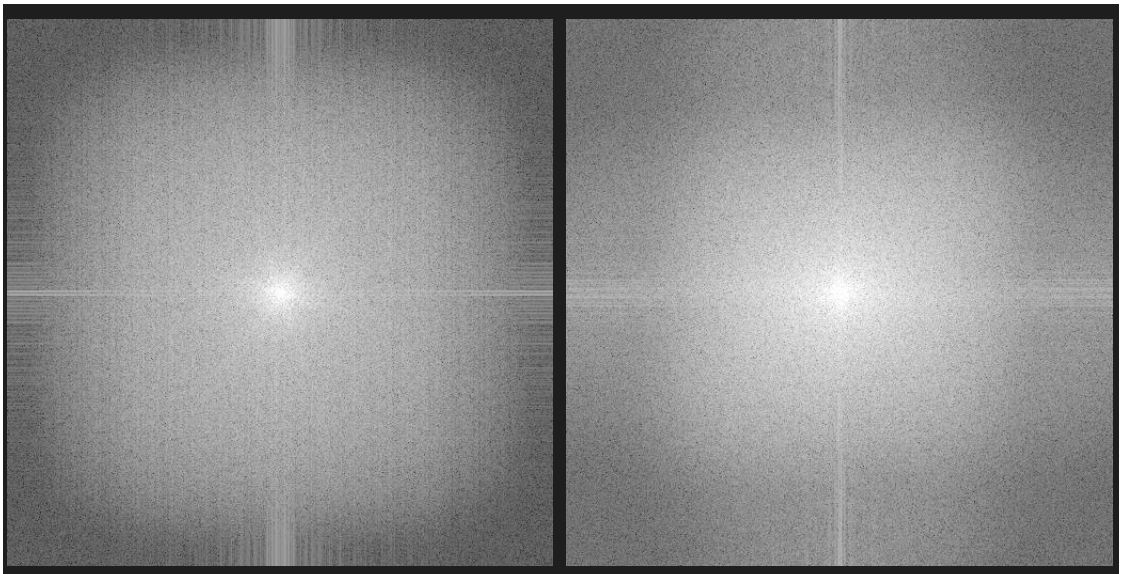
Part 2

On this problem we are going to process 2 images with high and low pass filters. We first apply DFT to both our images. The filters are worked in the frequency domain by restricting frequencies within a certain sector of our DFT.

dew on roses (noisy).tif

tulips irises.tif

DFT's of Images



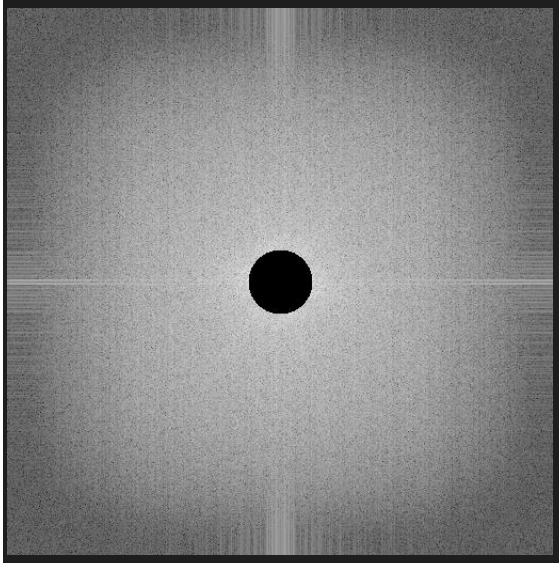
dew on roses (dB)

tulips irises (dB)

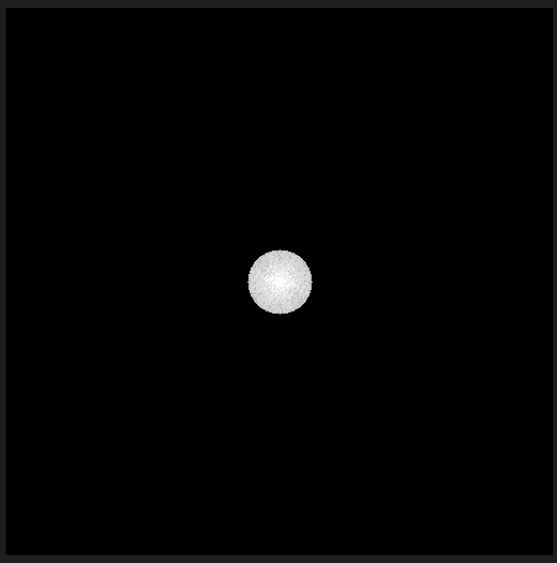
DFT of image "f":

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi \left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

Dew on roses (Frequency Domain)

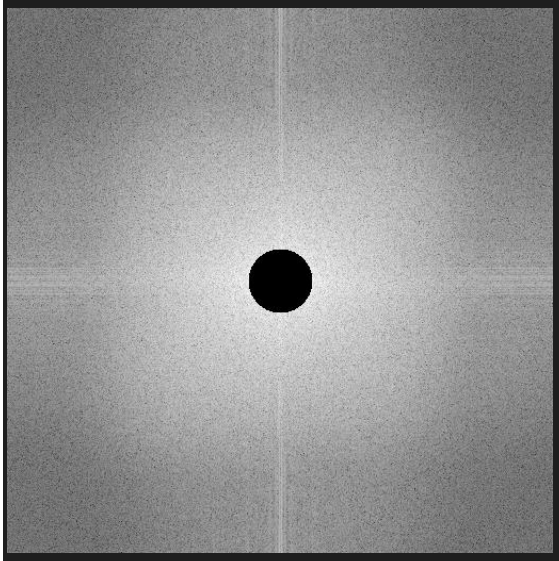


(HP filter, r = 30 px)

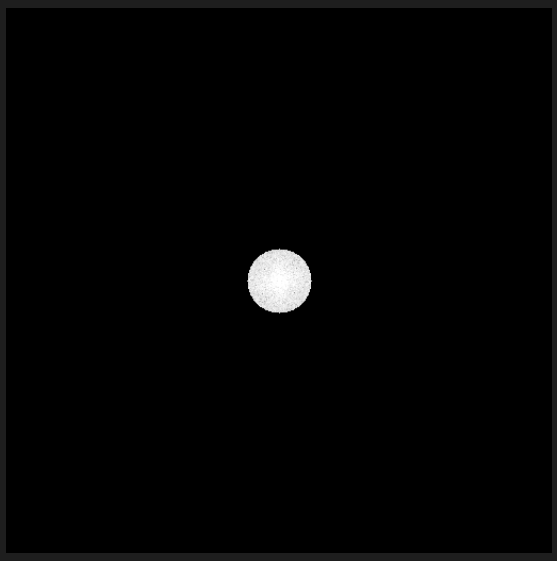


(LP filter, r = 30 px)

Tulips irises (Frequency Domain)



(HP filter, r = 30 px)



(LP filter, r = 30 px)

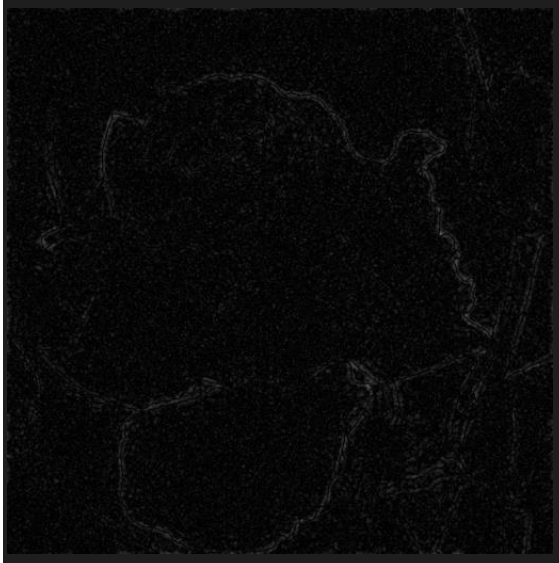
HPF Example:

```
for all_elements in image_DFT:
    if square_root((element_row-(img_height/2))^2 +
        (element_col-(img_width/2))^2) < r:
             element = 0
```

Note: Low Pass is generated in the same way with the exception that we take all pixels inside the region.

As we can see in the results High Pass highlights the borders of our images. On the other hand, the Low Pass filter reduce the details on the images.

Dew on roses (Space Domain)



(HP filter, r = 30 px)

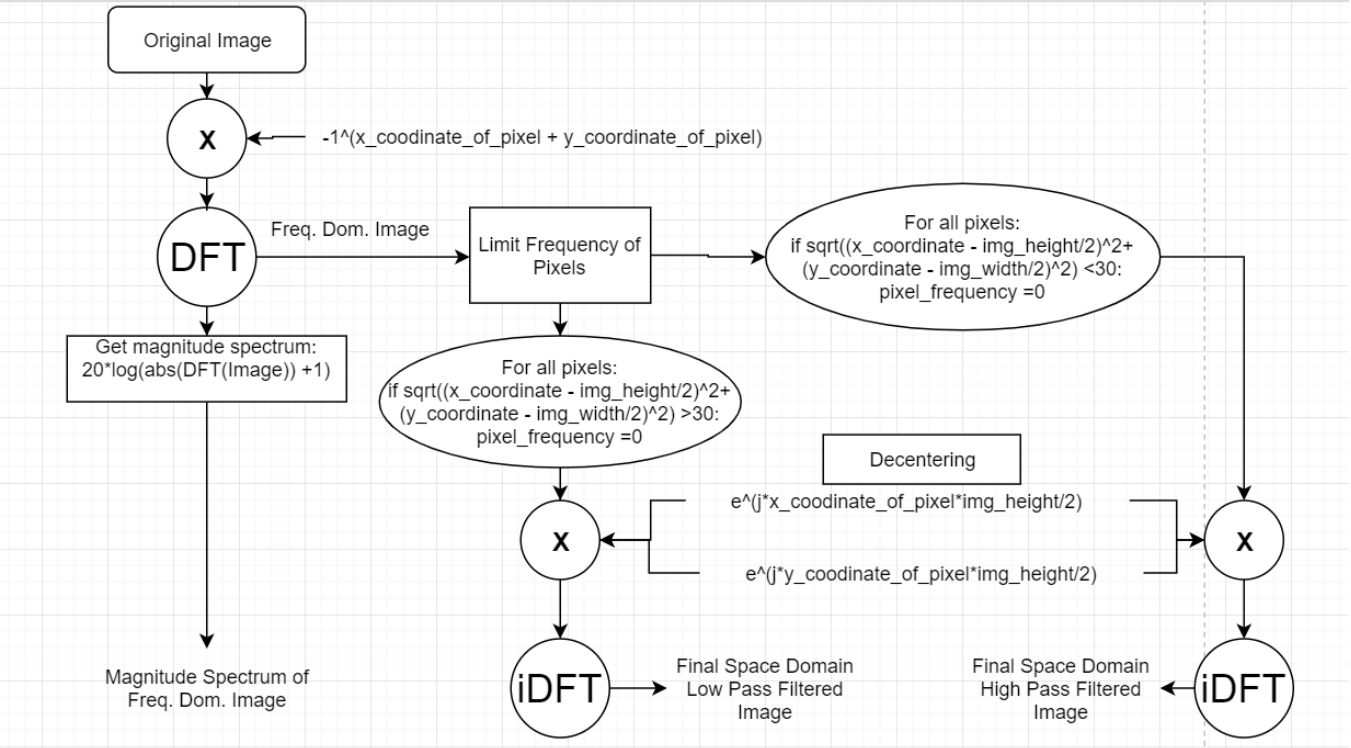
(LP filter, r = 30 px)

Tulips irises (Space Domain)

(HP filter, r = 30 px)

(LP filter, r = 30 px)

Flow Diagram of part 2

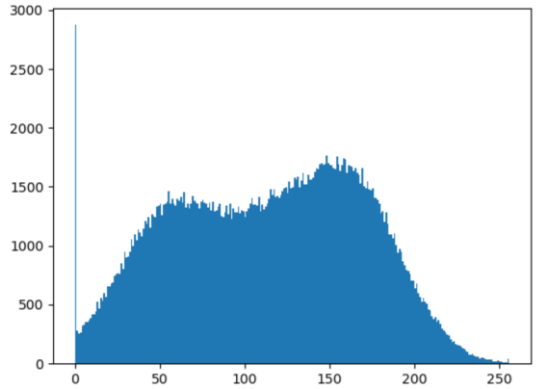


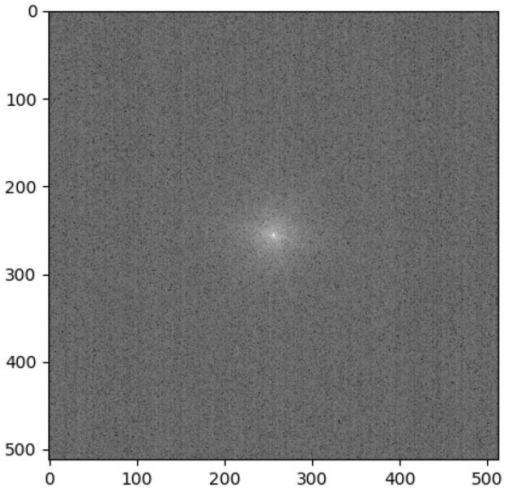
Part 3

De-noising

Original Image

After histogram looking at the histogram and DFT it is presumed that our image has being Gaussian blurred with small std and zero mean. Furthermore, we see some dots around our image, therefore we think the dots must be the spike we see at zero.



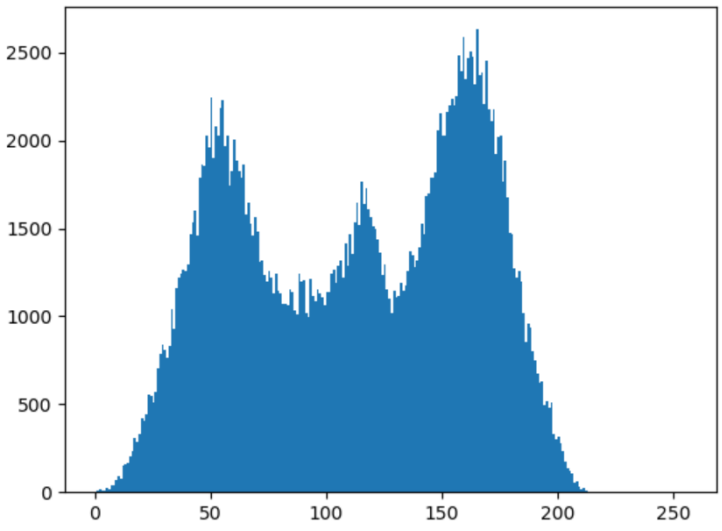


Strategy: Applying a moving average kernel (median filter). After multiple attempts we think that the most suitable kernel size is 5x5.

Results

Size: 3x3 Size: 5x5

Size: 7x7 Size: 9x9



5X5 MEDIAN HISTOGRAM

As we can see we successfully removed the spike from our histogram.

De-blurring

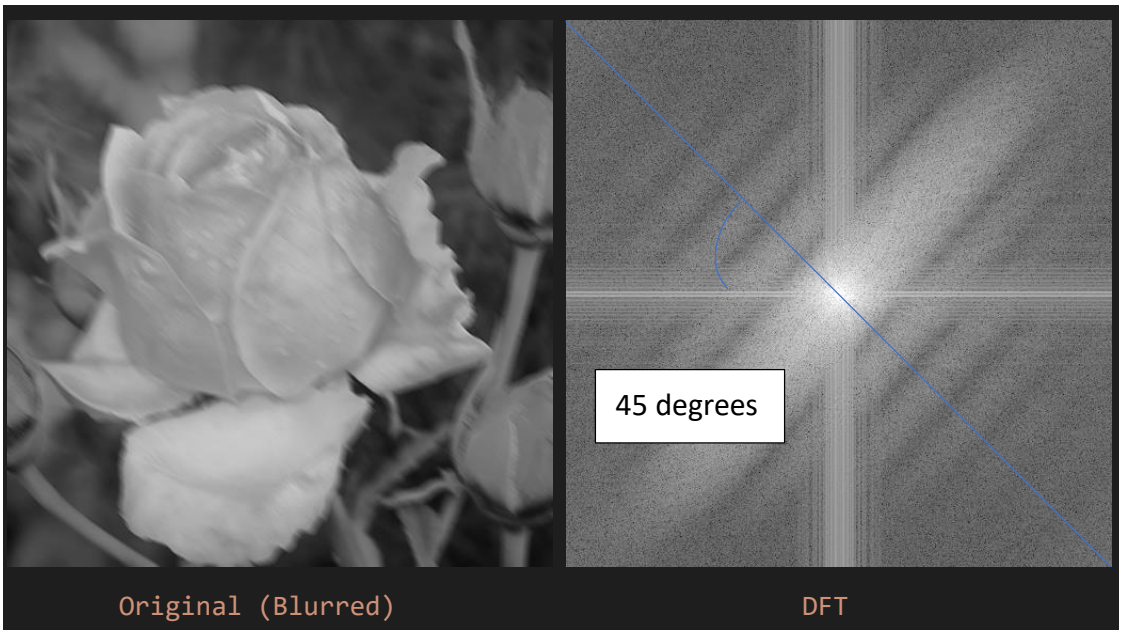
In order to take care of the motion blur. We first need to estimate a degradation function H. The degradation function is assumed to have the following shape.

```
for u in [0 to image_height]:
    for v in [0 to image_width]:
        t = u - image_height/2
        s = v - image_width/2
        T = 1
        term = 3.14 * ( t*a + s*b )
        kernel[u][v] = (T/term) * sin(term) * e^(-1j*term)
```

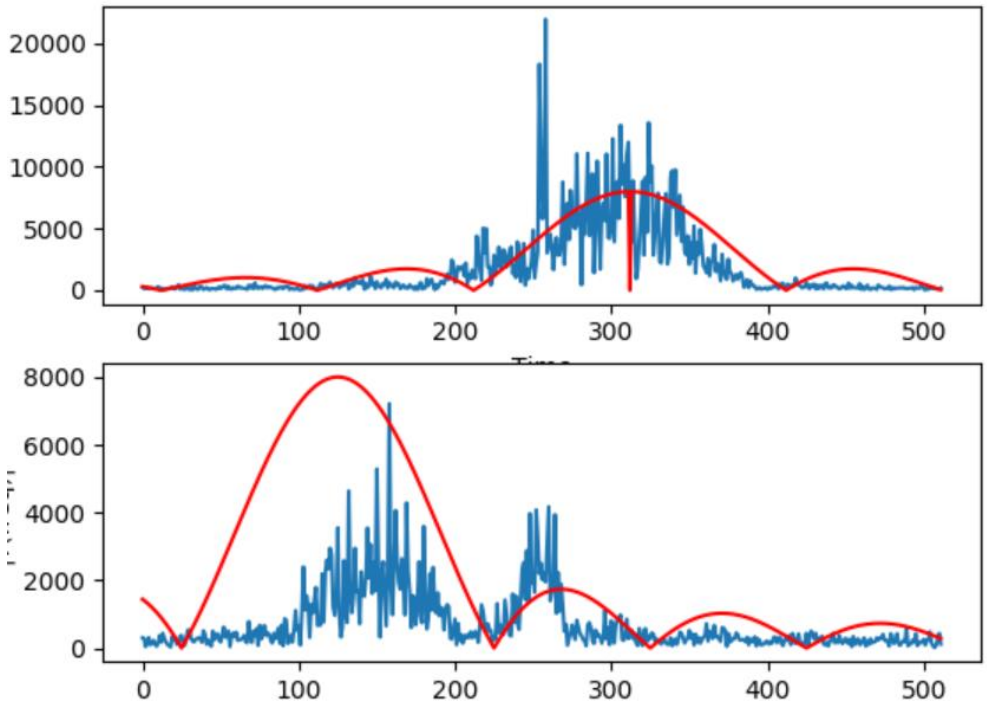
We can estimate values for either “a” and “b” in the following manner.

```
a = X
b = X / tan(angle*3.14/180)
```

To get an approximation we first take the DFT of our blurred image. As a reminder, our estimation kernel is already in frequency domain so it can be worked directly in this domain. In order to get the angle to calculate “b” we can look at the DFT of our blurred image and estimate the angle to the horizontal. We draw a line perpendicular to the stripes the we see in frequency domain as reference. For our case we found the angle to be 45.

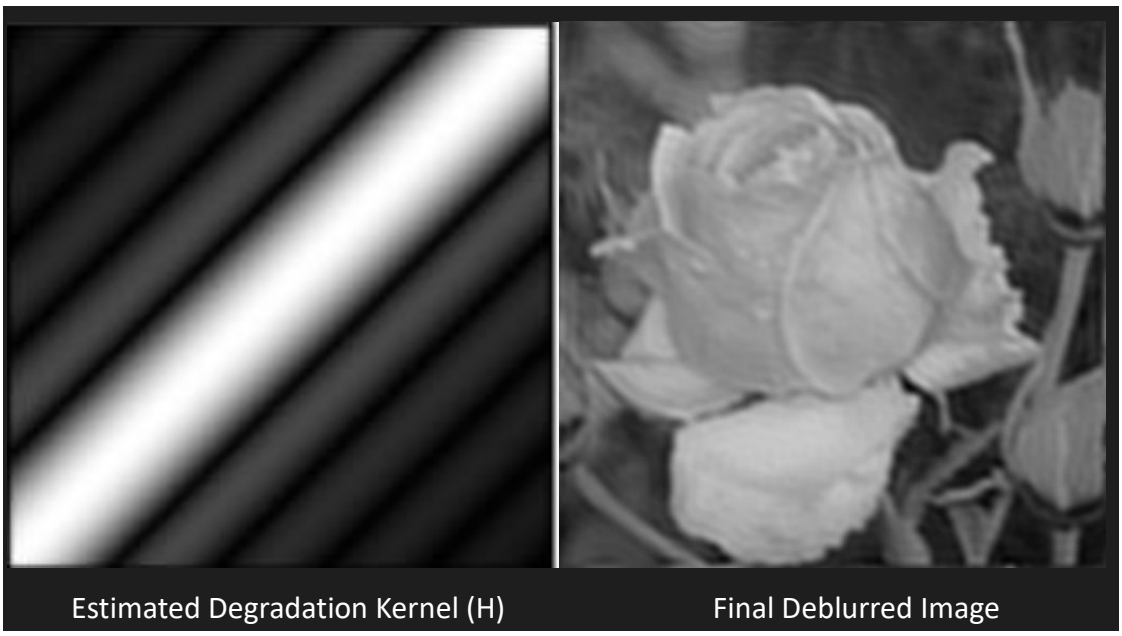


After obtaining the angle we attempt different values for “a” until we get a good estimate. A strategy that might come handy is picking a linear section from our DFT and one from our kernel. Then we modify until they resemble close enough.
We determined that a good estimate for “a” might be 0.01.



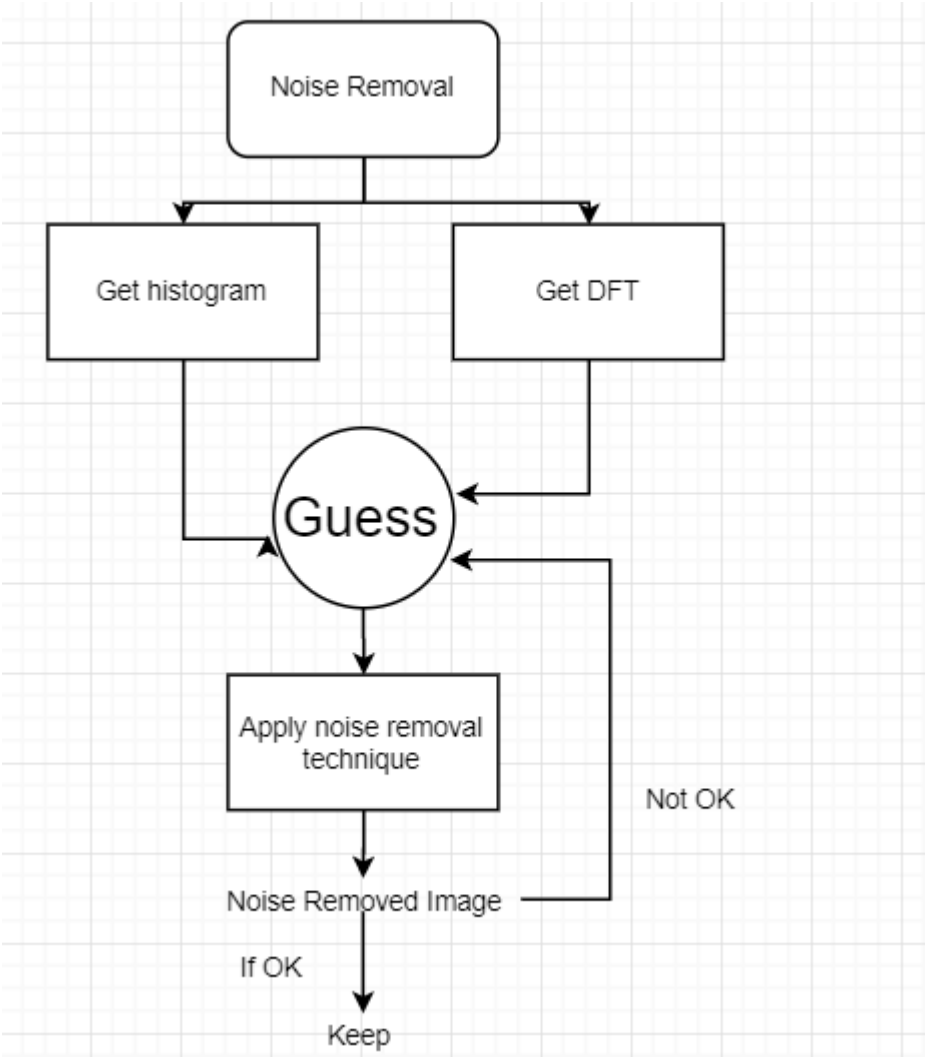
These are 2 slices, one horizontal and one vertical. As we can see plots appear to be similar with a factor of 0.01. The period (T) of our function was increased up to 7000 in order to scale-compare in the plots of our kernel and our image_DFT slices. When doing the final calculations T=1 should be used.

Finally, we divide our blurred image by our kernel. This is the result:

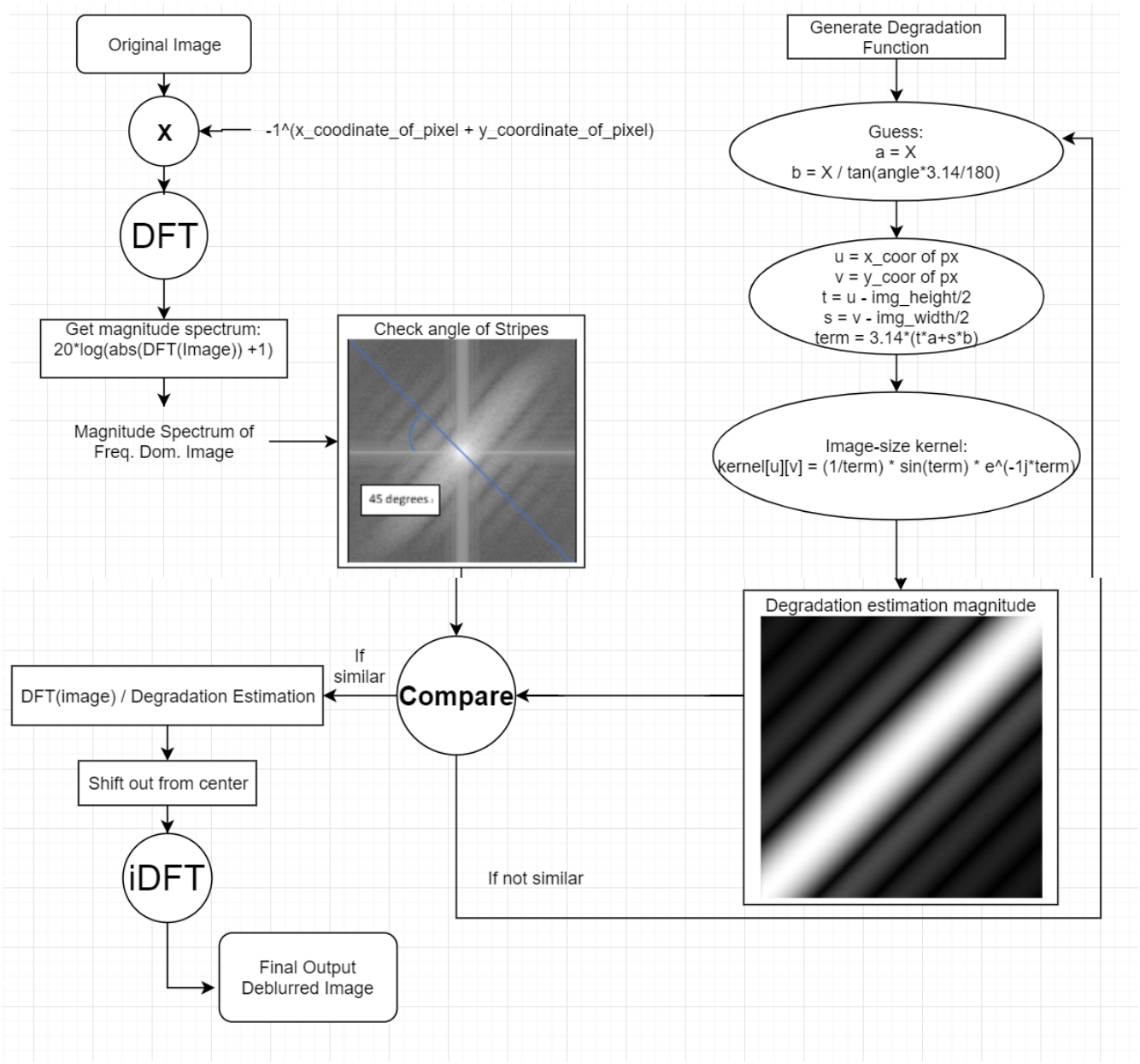


Estimated Degradation Kernel (H) Final Deblurred Image

Flow Diagram of part 3A



Flow Diagram of part 3B



Final Discussions

Part 1

In this part we attempt image enhancement, but mainly we are looking forward to observe different image transformations. During the processing of this part we mainly used concepts from the book and basic matrix/image operations. During the programming, there were some technicalities that needed to tackle such as remapping the pixel values from floating to int and vice-versa when needed. We also saw some applications of the Laplacian, which include image enhancement. Later we used it in other operations such as addition, power, averaging and product (from either the original image, the Laplacian, or a result of these operation).

Part 2

In this part we practice filtering frequencies and processing our image in the frequency domain. After obtaining our Discrete Fourier Transform we limit pixel frequencies to a certain region. Everything outside/inside this region is overlooked for finally obtaining a high pass filter or low pass filter. As we saw in the 2 images that were processed, one was full of detail, whereas the other was much simpler and plain. As result we see how HPF effect is clearer our detailed images, whereas LPF suits more our simpler image.

Part 3

As we saw in this part denoising and deblurring are not exact sciences. Many estimations were done during the process. The most difficult step in this part is to subjectively determine is the image is better or not. Since we don't count with any original picture we can't say if the restoration was more successful or not.

Ideally, in this part we could also compare our noise-removed image with our deblurred image and see if they match by subtracting them. Problem is that we don't know if the noise was well removed. Moreover, the deblurred image may have ringing effect which is inevitable, therefore even if we successfully removed the noise from our image, we can't completely restore the blurred image.

We concluded that a median filter could be use to denoise the image.
For the blurred image we concluded that the motion angle was 45 degrees and the parameter "a" = 0.01.