CMPEN/EE455: Digital Image Processing I

Computer Project # 3:

Filtering using Frequency-Domain Analysis

Ritvik Muralidharan, Georges Junior Naddaf, Zian Wang

Date: 10/20/2017

A. Objectives

This project's main purpose is to

- Introduce to DFT-based frequency analysis using MATLAB
- Introduce to image corruption and Notch filtering using MATLAB

B. Methods

We have created 3 files in our directory. The first is Project3.m, which is the code implemented for part 1 of the assignment and the second is Project3_2.m, which is the code implemented for part 2 of the assignment. We also have notchfilter.m, which implements the notch filter used in the second part of the assignment. To run the code simply run either Project3.m or Project3_2.m depending on the part desired. The code is sequential and runs in roughly the same order the questions were asked in both parts. Once run, the code outputs the images needed in the same directory in which the code file is located. It was difficult to use the imwrite() function since MATLAB does not support .gif files for this function. Note that to use the Gaussian low-pass filter, we use the file given by the professor called dftuv.m.

Part 1 a):

For this question, we start by inputting the checkers.gif image. We then perform the Fourier transform using fft2 of the input cast as a double for increased precision. We then cast it to uint8 to be able to view it using the imtool and store it in F. We then compute the logarithmic expression that includes F and store the result in LF. We then perform a loop and traverse our original image performing modulation by multiplying by -1^(x+y) and storing the result in f_m. We then, as before, compute the fourier transform and the logarithmic expression of the modulated image f_m and store the results respectively in F_m and LF_m. We then recalculate the fourier transform of the input simply because we used uint8 for display purposes in the previous allocation making it not usable for future operations and so the redefinition was needed. We set the F(1,1) = 0.0 as requested and perform the inverse Fourier transform of F to get g.

<u>Part 1 b):</u>
For this part, we input our original image again. We perform a Fourier transform on it and get F. We then create a Gaussian low pass filter using the same method described in the demo. We use a loop to apply the filter H on F and get the resulting G2. We then get g2 by performing an inverse transform on G2. We then create a loop to modulate this g2 and store the result in g2_m. Afterwards, we compute the absolute value of g2_m and the logarithmic expression containing g2_m and store the results respectively in G2_abs and LG2_m.

<u>Part 1 c):</u>
For this part, we just have to zero-pad similar to the example in the demo. Due to our previous explanations, it should make sense what each line in this code does.

<u>Part 2 b):</u>
To give suitable pictures of the requested images, we performed modulation then the absolute value of each function to get the clearest and most distinguishable images. The process of completing operations here is the same as the one used in part 1. I will explain the operations not covered previously. To create the corrupted image, we simply create a loop, traverse it and perform the calculation specified in the assignment prompt for each pixel. To create the notch filter, we utilize the notchfilter() function and then apply it to the Fourier transform of the corrupted image and store the result in G. To display G, we had to perform a scaling of 15, giving us a suitable image.

C. Results

Part 1 a):

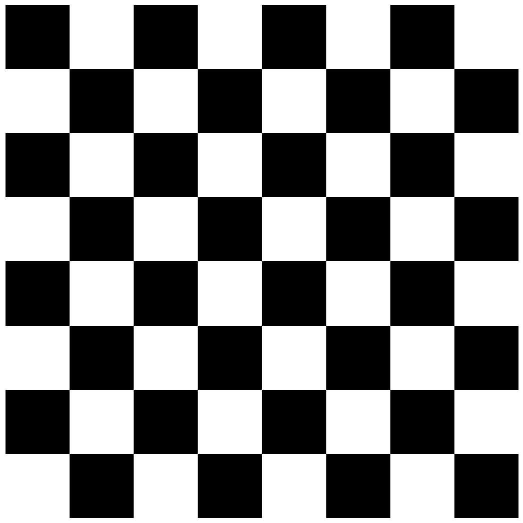


Figure 1. Original image

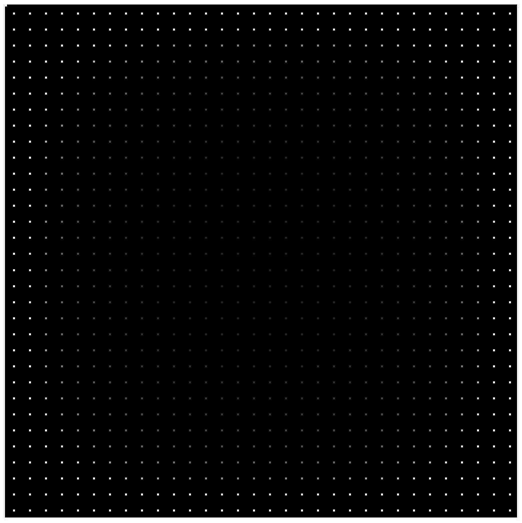


Figure 2. 1.(a) |F(u,v)|

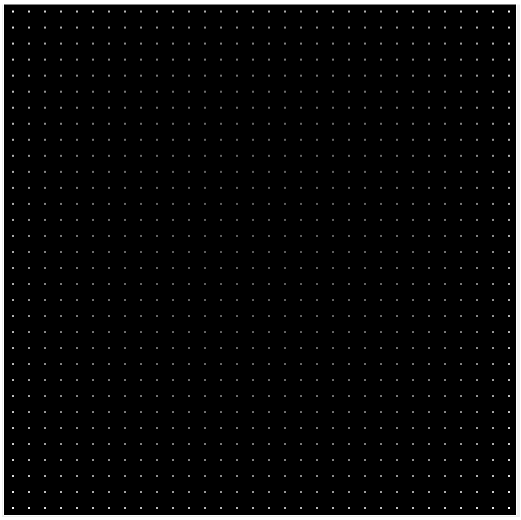


Figure 3. 1.(a) log(1+|F(u,v)|)

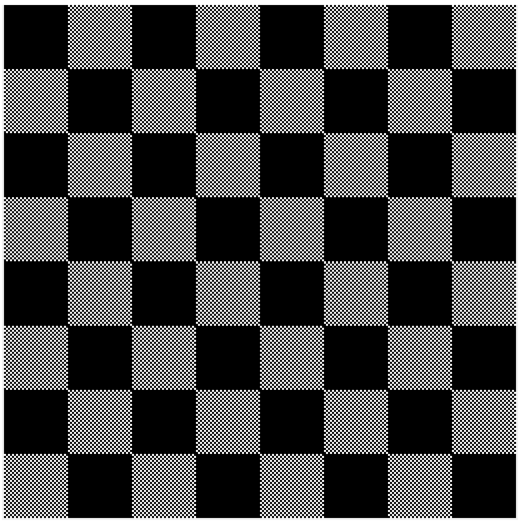


Figure 4. 1.(a) fm(x,y)

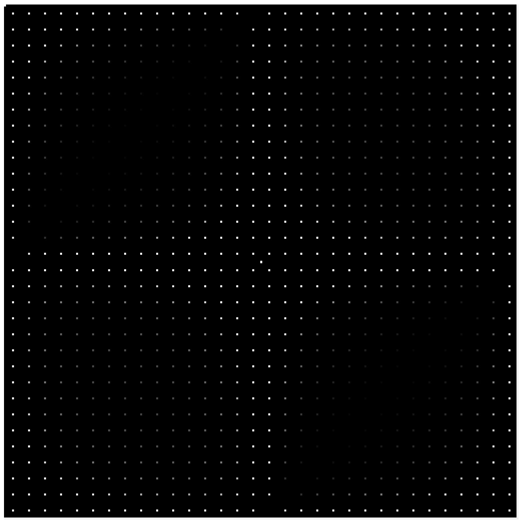


Figure 5. 1.(a) |Fm(u,v)|

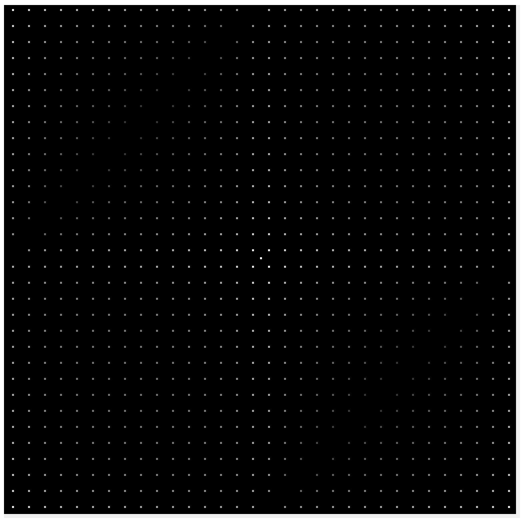


Figure 6. 1.(a) log(1+|Fm(u,v)|)

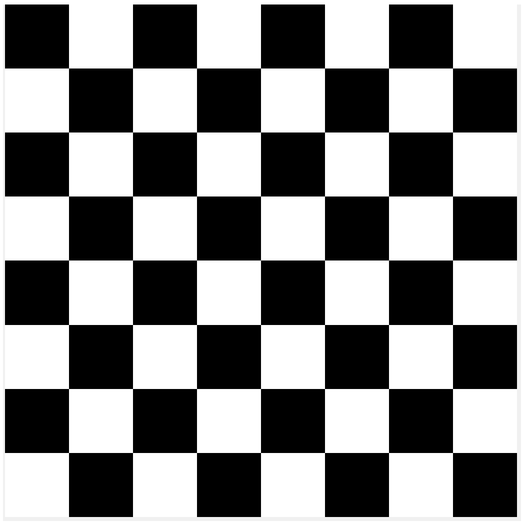


Figure 7. 1.(a) g(x,y)

Part 1 b):

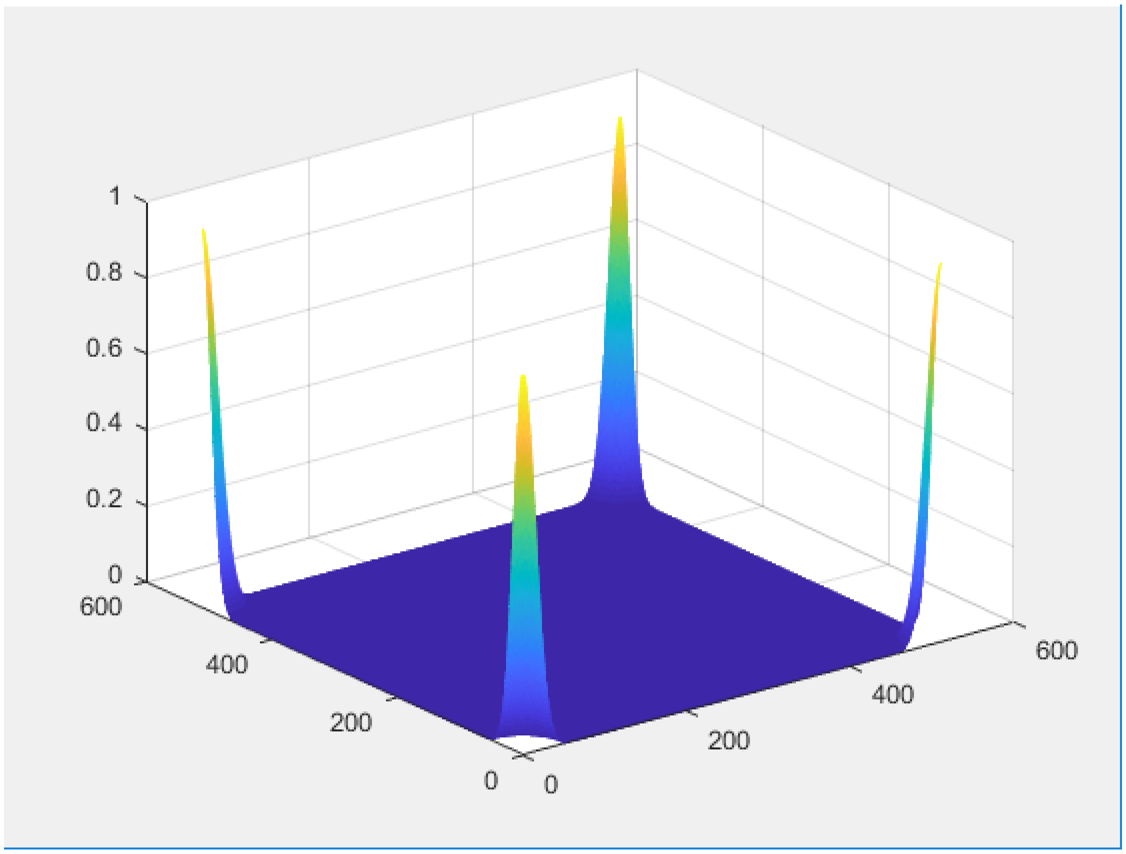


Figure 8. 1.(b) |H(u,v)|

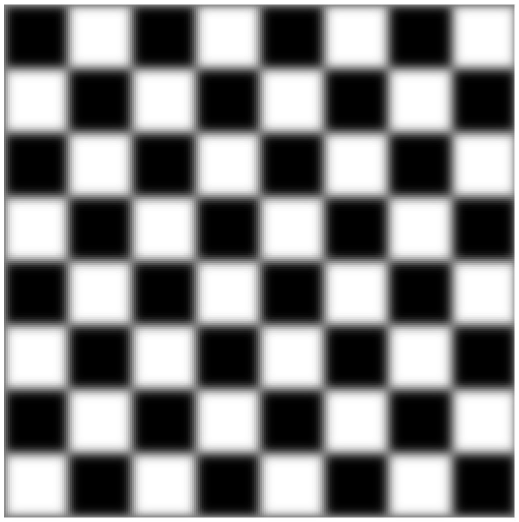


Figure 9. 1.(b) g(x,y)

Wraparound error is present.

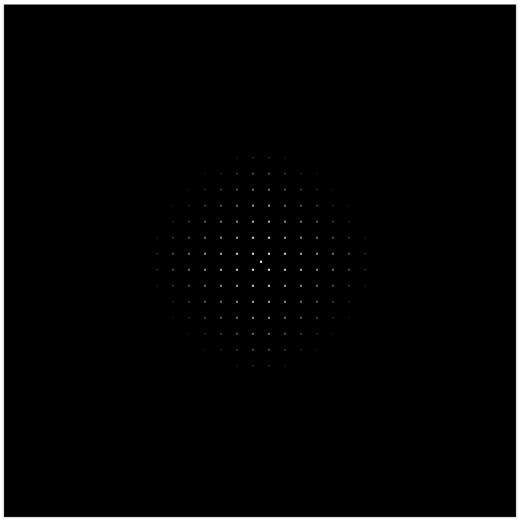


Figure 10. 1.(b) |G(u,v)|

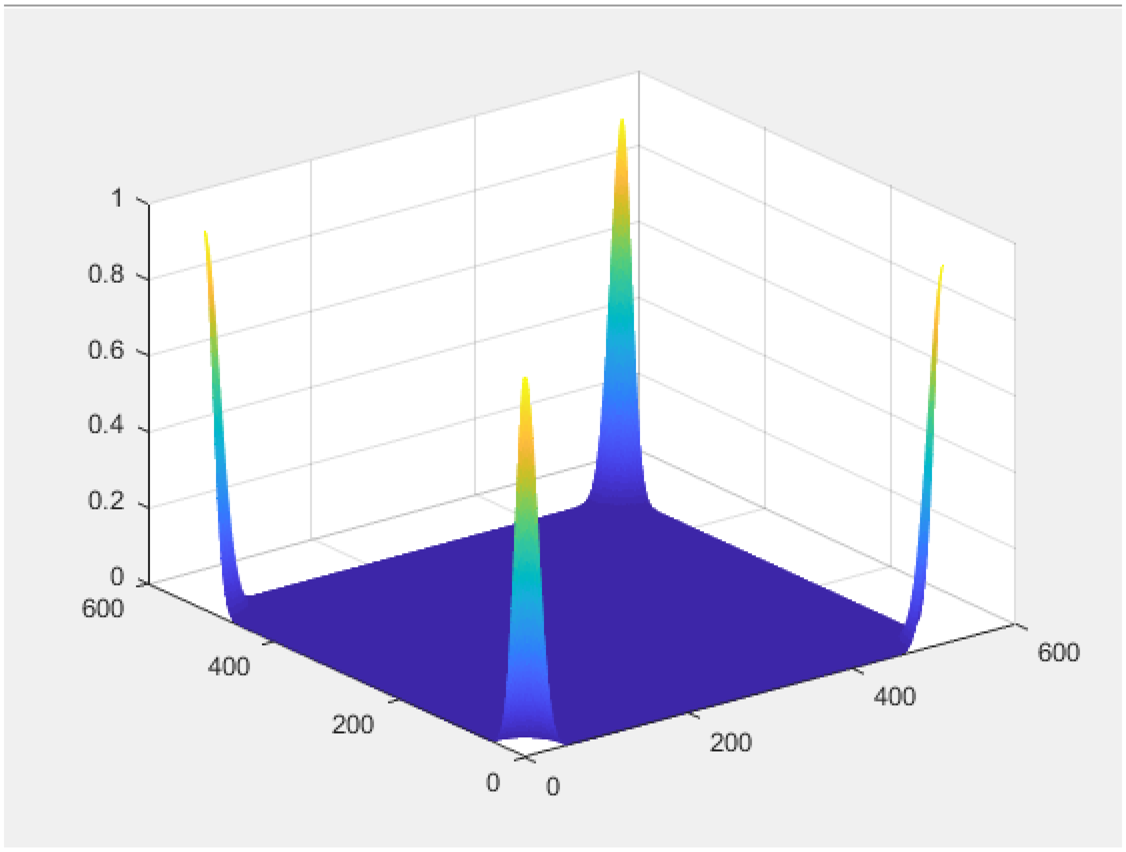


Figure 11. **1.(c)** |H(u,v)|

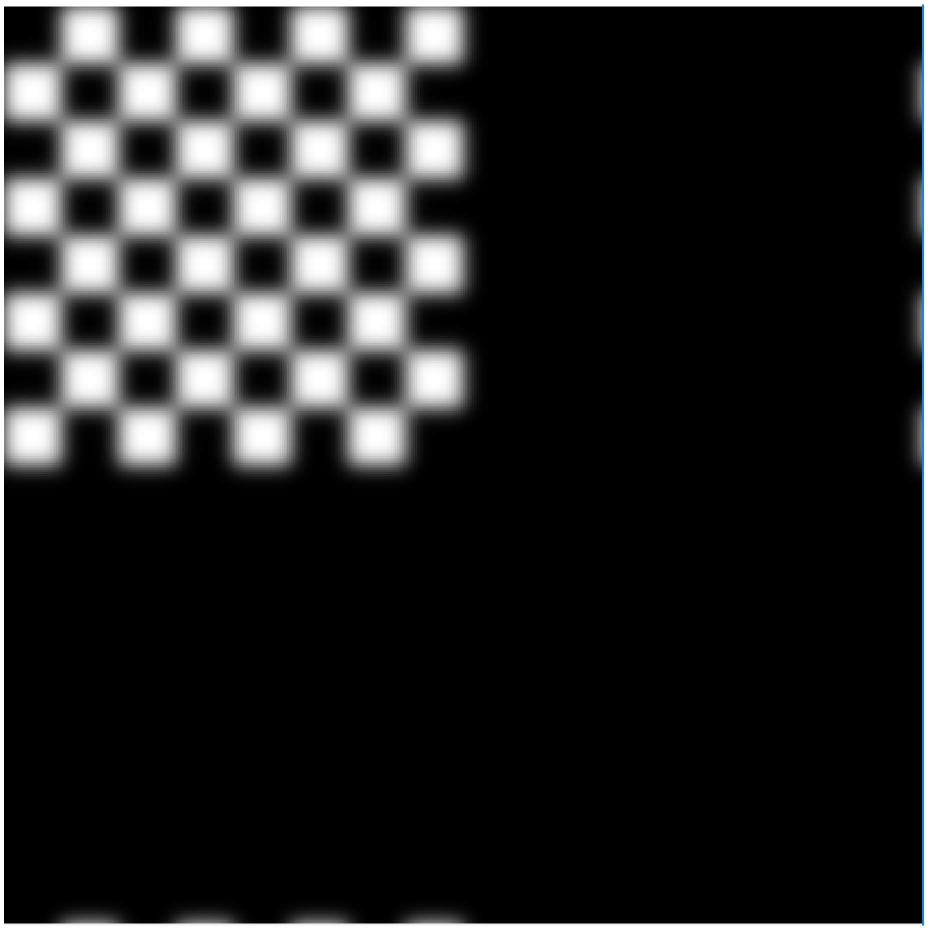


Figure 12. 1.(c) g(x,y)

No wraparound error is detectable.

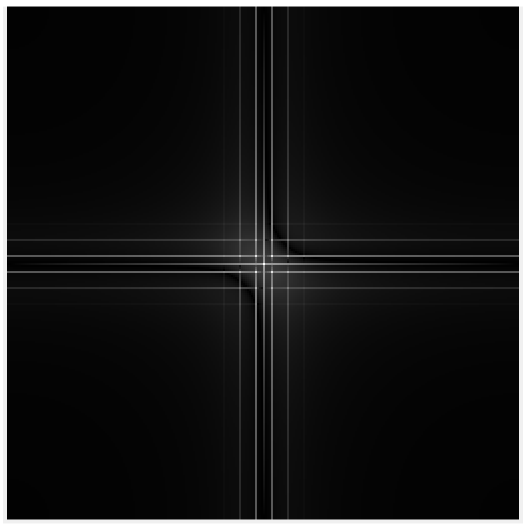


Figure 13. **1.(c)** |G(u,v)|

Part 2:

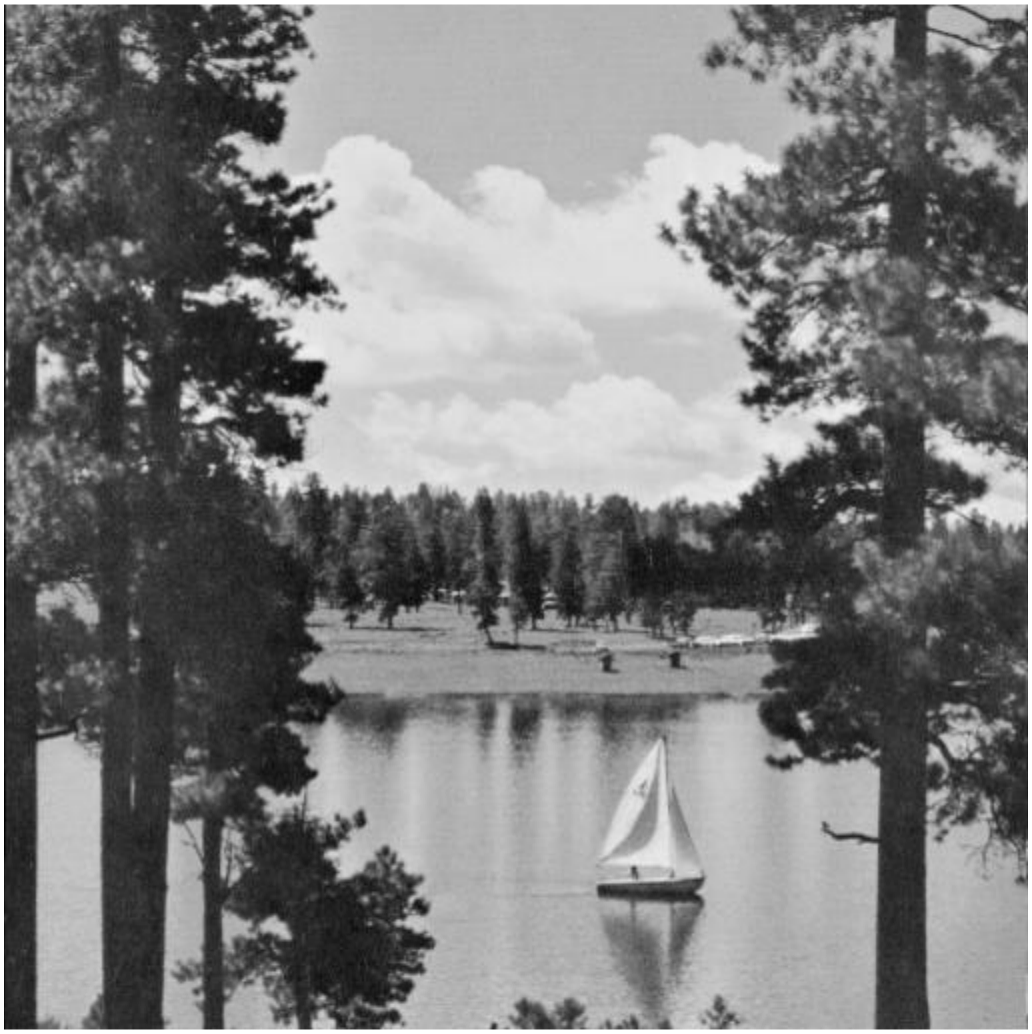


Figure 14. **2.(b)(i)** f(x,y)

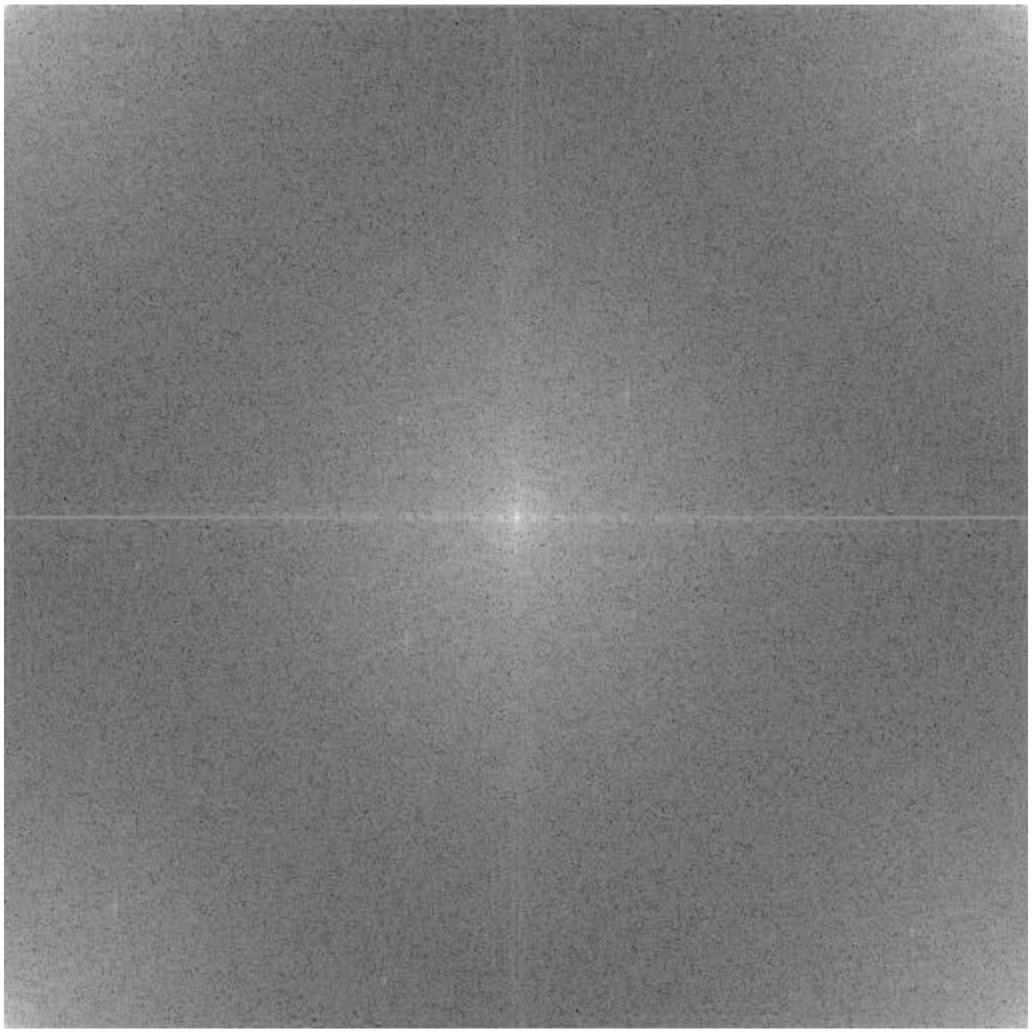


Figure 15. **2.(b)(i)** magnitude of F(u,v)

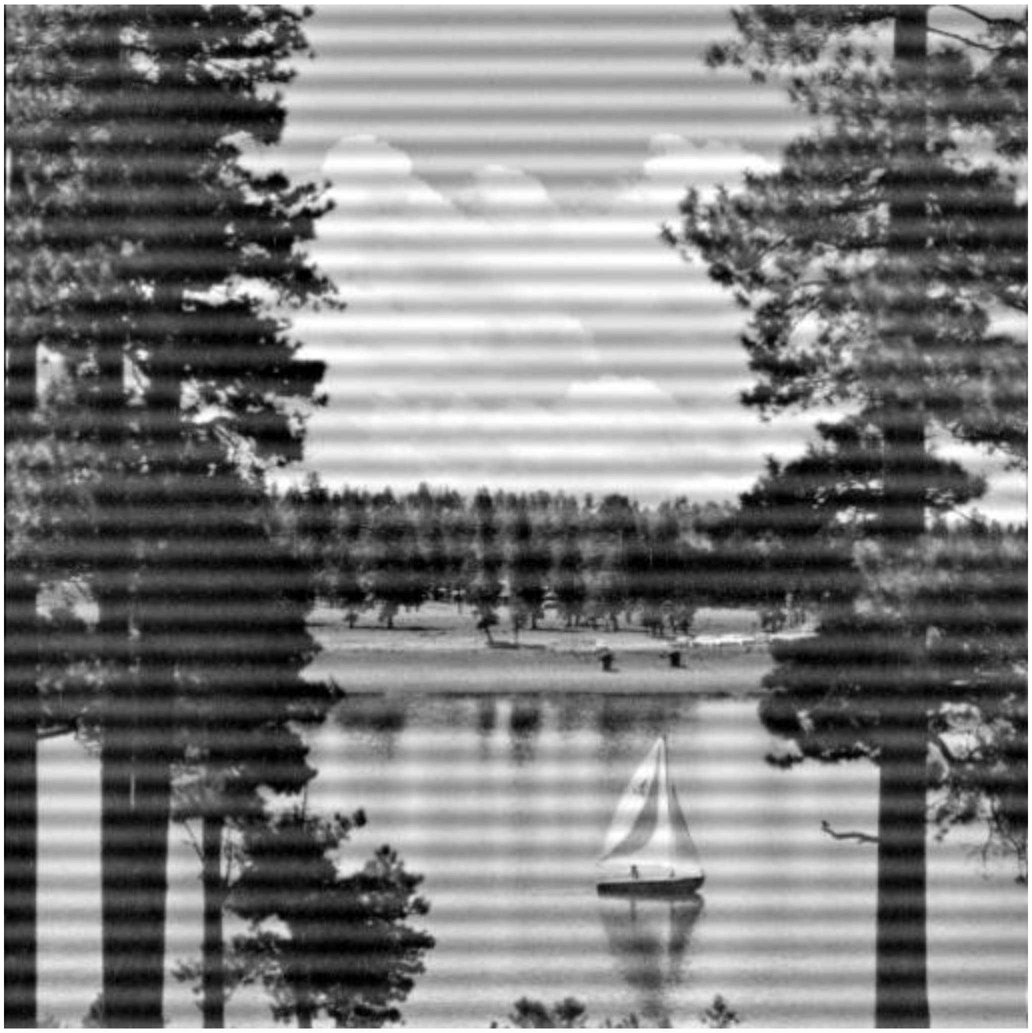


Figure 16. **2.(b)(i)** Corrupted image c(x,y)

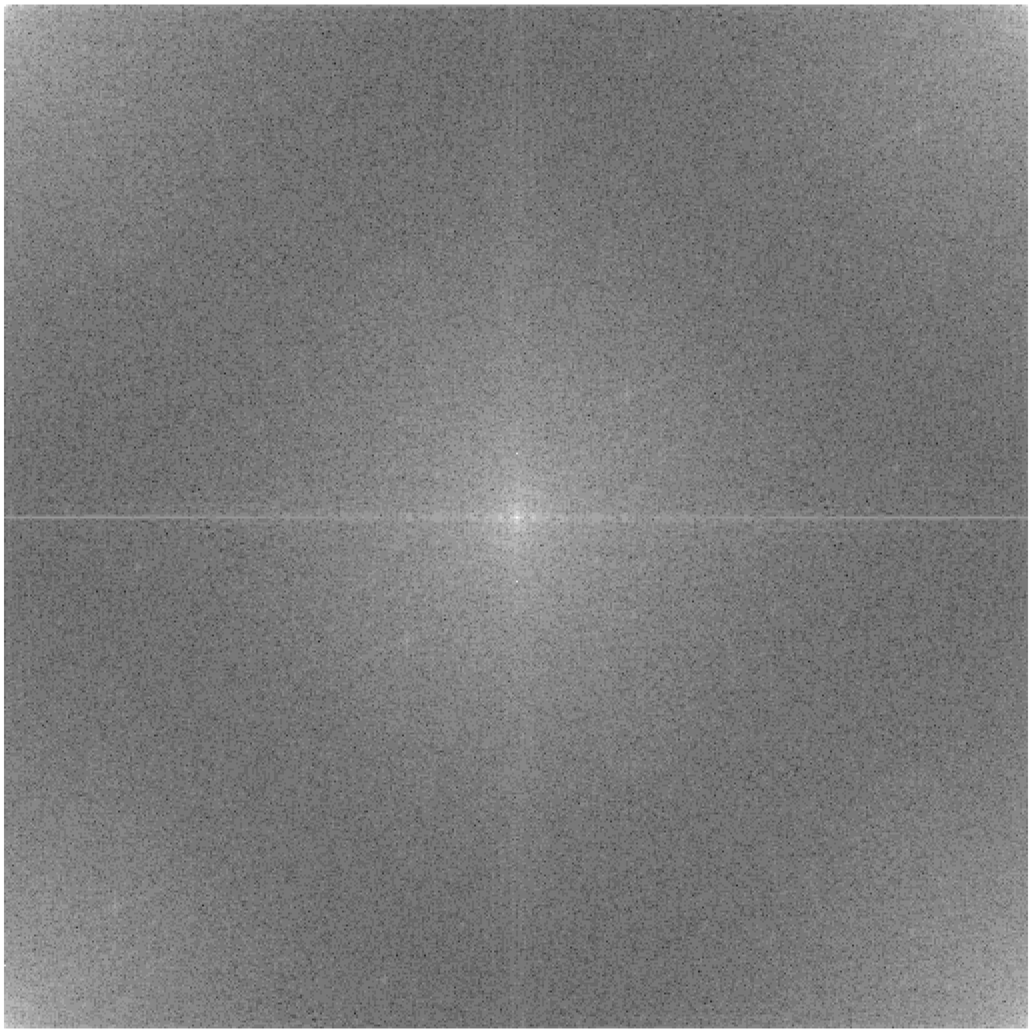


Figure 17. **2.(b)(i)** magnitude of C(u,v)

Figure 18. **2.(b)(ii)** magnitude of H(u,v)

There are two points present in the middle of the image but they are hard to perceive. This is an accurate representation of the notch filter since it eliminates two sets of frequencies.

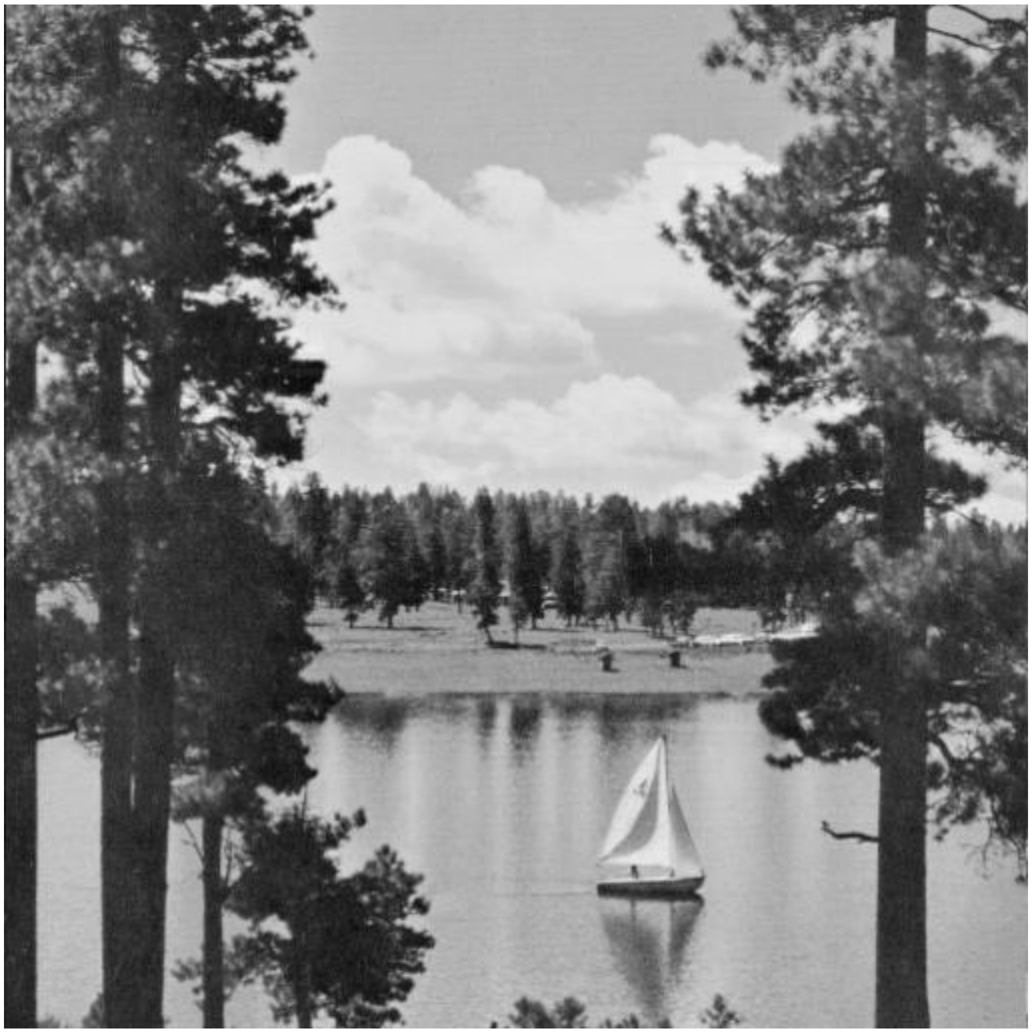


Figure 19. **2.(b)(ii)** g(x,y)

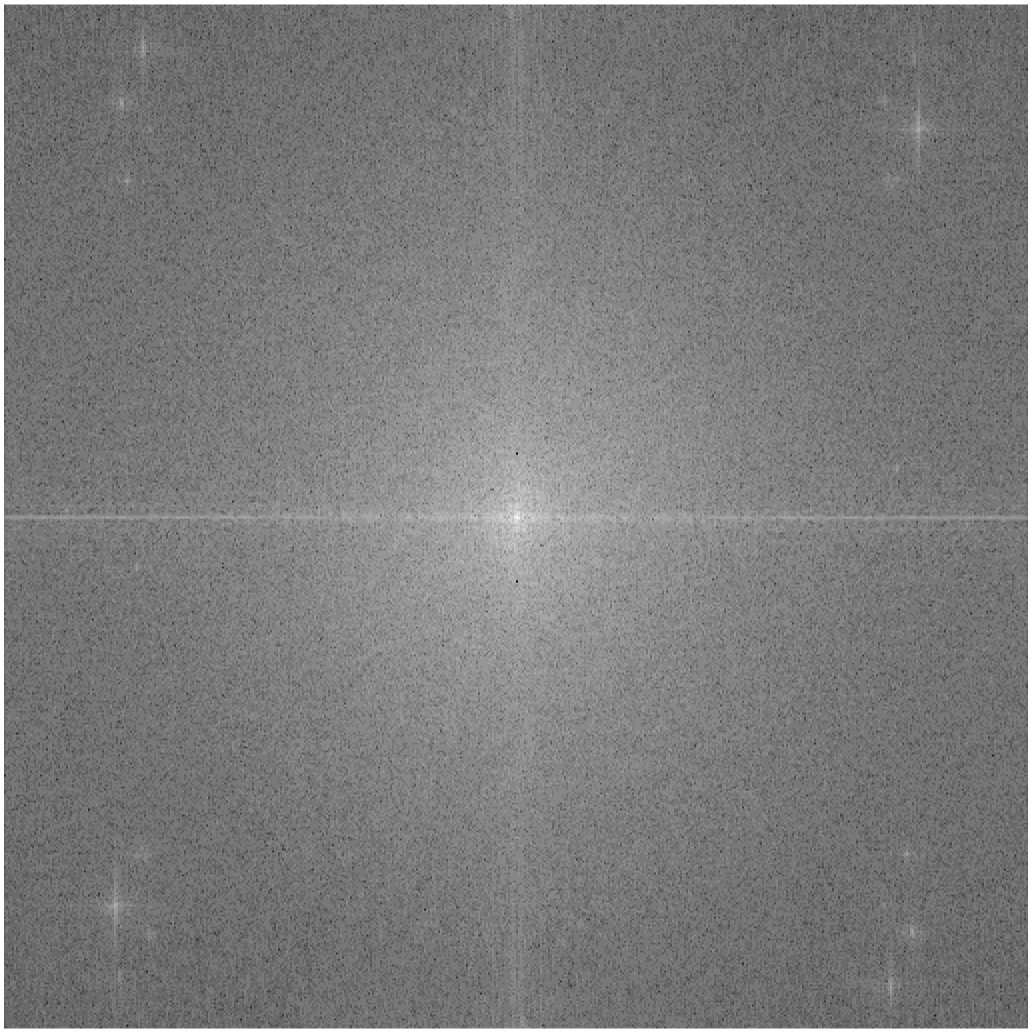


Figure 20. **2.(b)(ii)** magnitude of G(u,v)

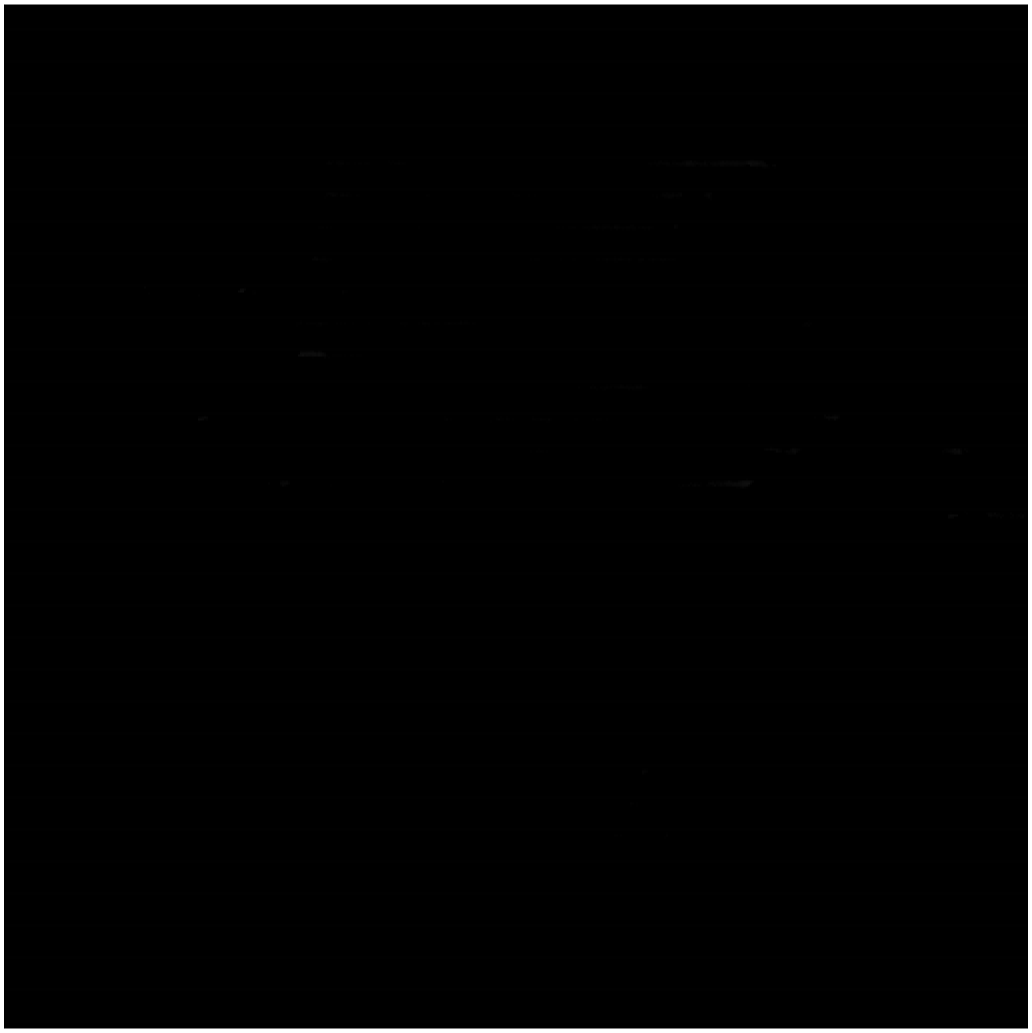


Figure 21. **2.(b)(iii)** Difference image (f(x,y) - g(x,y))

As we can see, the difference between the original image and the filtered corrupted image is non-existent. This showcases the efficiency of our notch filter.

Questions to answer:

Part 1 a)

i) What do the log and modulation operations do?

These two operations produce clearer versions of the original image in which discrepancies are easily detectable.

ii) What is the observable and analytical difference between g and the original f?

There is hardly no observable difference between f and g. However, in reality, we set F(1,1) = 0.0 for G and this removes the DC component from the new image.

Part 1 d)

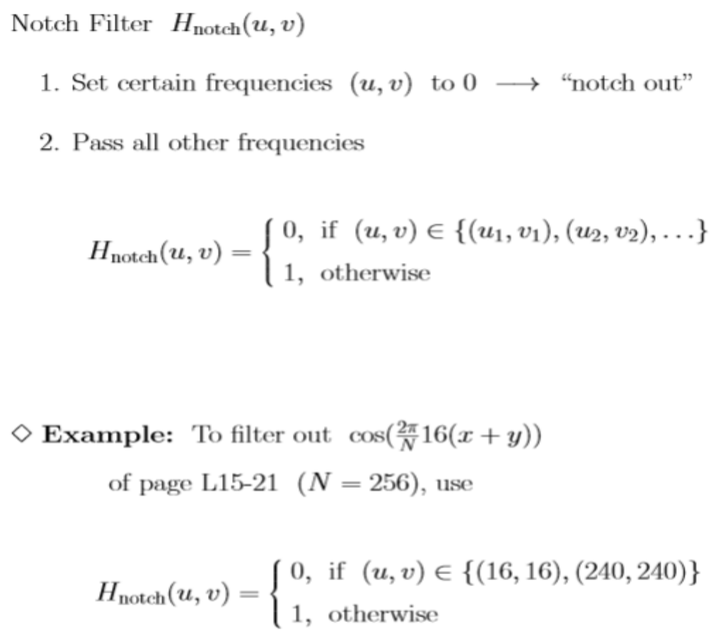
For the results of parts (b-c), what impact does a filter have on the output image? Do you observe wraparound error? Discuss the nature of the zero-padded results.

The application of the Gaussian filter made the images appear blurrier. This is because it restricts some of the frequencies in the image. For the first, non-padded image, we observe a wraparound error when zooming in on the pixels. When we zero-pad the image however, we no longer see this error making this image more favorable and correct.

Part2

You must describe how you designed your filter H by giving analysis to back up your design. Lecture notes L15-21 → L15-24 are very helpful here!

To design the notch filter we relied heavily on the lecture notes. The following slide was the main source of help:



Notch Filter $H_{\text{notch}}(u,v)$

1. Set certain frequencies (u,v) to 0 $\longrightarrow$ "notch out"
2. Pass all other frequencies

$$H_{\text{notch}}(u,v) = \begin{cases} 0, & \text{if } (u,v) \in \{(u_1,v_1),(u_2,v_2),\ldots\} \\ 1, & \text{otherwise} \end{cases}$$

◇ **Example:** To filter out $\cos(\frac{2\pi}{N}16(x+y))$ of page L15-21 $(N = 256)$, use

$$H_{\text{notch}}(u,v) = \begin{cases} 0, & \text{if } (u,v) \in \{(16,16),(240,240)\} \\ 1, & \text{otherwise} \end{cases}$$

Since in our case we had the double of the example, we had to multiply 16 and 240 by two and added 1. These were the frequencies we aimed to eliminate so that we get rid of the corruption in the image. A simple set of if statements do the trick.

Is it possible to completely recover f from c? Why or why not?

It is possible to completely recover an image from a corrupted one to the extent of human observation. Analytically however, it is incredibly difficult to fix an image pixel-wise to its original state. If we had the corruption factor, it is possible to revert it. However, in real scenarios, that factor is almost always unknown making analysis like the one done in this experiment a must.

D. Conclusions

The project addresses peculiar questions concerning DFT-based frequency analysis and corruption minimization from an image. The key points to take out are:

- There is no observable difference between an original image and a transformed but reverted image. However, analytical differences may be present such as the absence of the DC component.
- The application of a Gaussian filter on an image produces a wraparound error that can be resolved by zero-padding the image.
- A notch filter is a filter that suppresses certain frequencies in an image and it can be very helpful in many scenarios such as when dealing with a corrupted image
- A corrupted image can be reverted to its original state by using a notch filter on it specifically constructed using mathematical expressions that can be derived.

MATLAB allows the usage of powerful mathematical transformations that allow complex image analysis and processing with minimal code.