# EE420– Digital Image Processing

# Homework-3

Assigned: Feb 9, 2021 Due: Feb 18, 2021

Maximum Possible Grade: 50 points

# General guidelines

1. Please upload your response to Canvas as a ZIP file with the following filename convention

* Name of ZIP file FIRSTNAME\_LASTNAME\_Homework3.zip
* Replace FIRSTNAME with your first name
* Replace LAST NAME with your last name

1. **Failure to adhere to the filename convention will result in deduction of points.**
2. Provide a detailed response to each question, including supporting mathematical arguments. Include screenshots of image before and after image processing. Failure to do will result in deduction of points.

# Submitting your solutions

Please ensure that the ZIP file uploaded to Canvas, includes the following components:

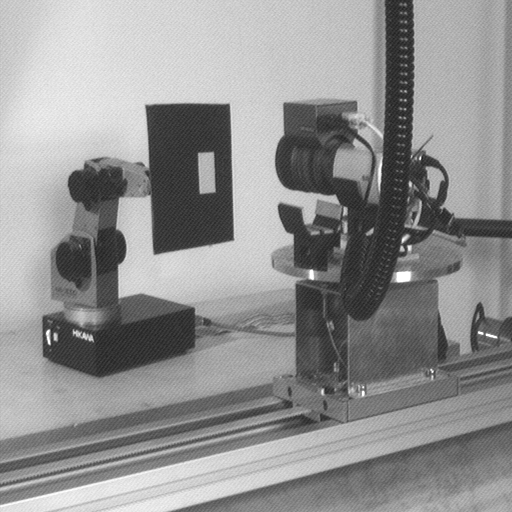
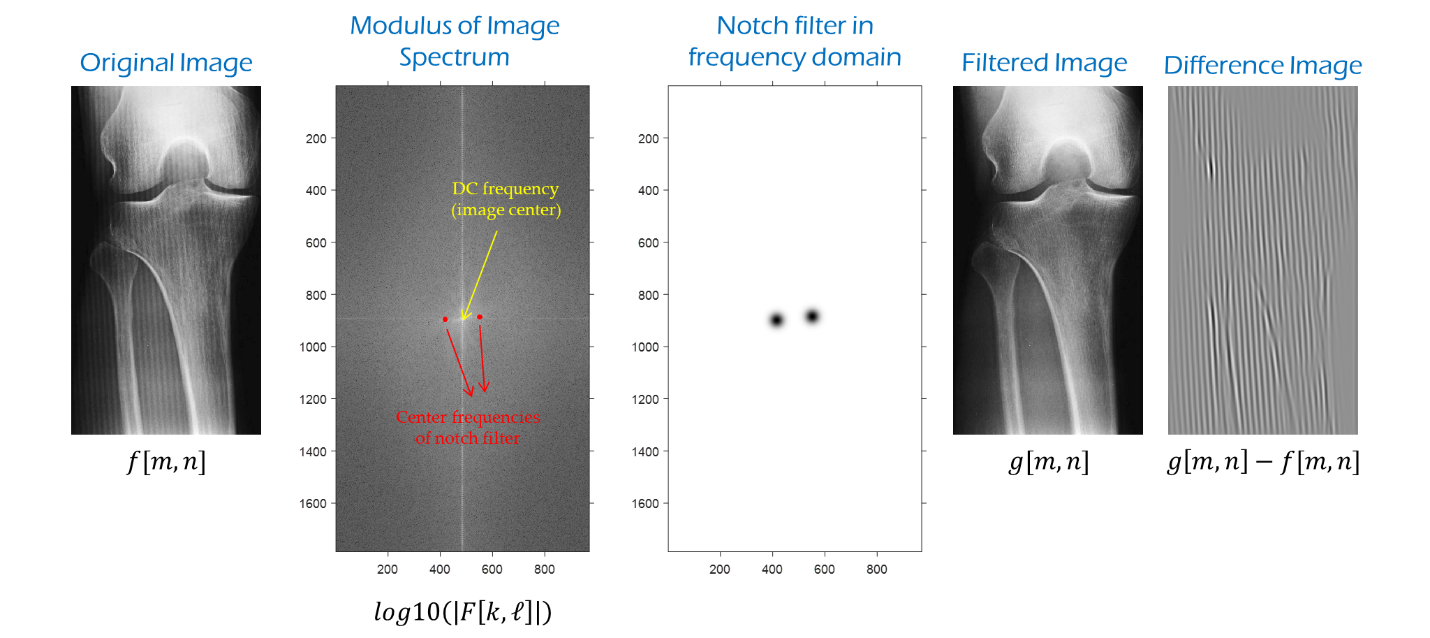
* PDF file of your write-up including response to all questions.
* Completed MATLAB Source code for
  1. HWK3\_SuppressingMoire.m
  2. HWK3\_ImageRestoration.m

**Failure to adhere to the filename convention will result in deduction of points.**

# Objective

This homework is designed to introduce you to image filtering in the frequency domain and image restoration techniques such as Inverse Filtering and Wiener Filtering.

# Suppressing Moire by notch filtering in frequency domain (25 points)

1. The image **Moire\_Example.png** included with the assignment includes an image of a robotic arm with periodic noise, called Moire. The noise resembles a pattern of closely spaced lines running diagonally from the bottom left of the image to the top right. The image inset provided below clearly illustrates the pattern. Your task is to implement a notch filter in the frequency domain that removes this noise.
2. The figure below visually illustrates the process of notch filtering:
3. 
4. The center frequencies of the notch filter appear as distinct peaks in the Fourier modulus (magnitude of the Fourier spectrum of the image) and are labelled in red. The DC frequency is in the image center.
5. The frequency response of each of the notch filters used in the above example is derived from a Gaussian as follows: where are the center frequencies of the notch filter. The final notch filter is obtained as the product of the individual notch filters.
6. The frequencies correspond to the row, column coordinates of the peaks in the Fourier modulus of the image, illustrated in red. It is necessary to subtract the offset from the center frequency of the notch filter as represent the row, column coordinates of the DC frequency component. represent the number of rows, columns in the zero-padded image representation of the original image. In the above example, the image has 893 rows and 485 columns, so that zero-padded image has 1786 rows and 970 columns. Notice that the peaks in the Fourier modulus are distributed symmetrically about the DC frequency, a consequence of the real-valued nature of images.
7. **WARNING:** Exercise caution when choosing the parameter , which represents the bandwidth of the notch filter. A very small value of will fail to remove the periodic noise. A large value of will result in image detail being filtered. This is most evident when examining the difference between the filtered image and the original image. Pick a value of so that the difference image largely resembles a periodic pattern such as the one shown in the example.
8. Tips: The MATLAB functions fft2,abs can be to used to compute the Fourier spectrum of the image and its modulus/magnitude. Please remember to compute the logarithm of the Fourier modulus before attempting to display it using MATLAB’s imshow or imagesc commands. Failure to do so will produce dark images instead of the ones shown above.
9. The starter code supplied with the assignment loads the image and outlines the key steps in the algorithm. It also includes a function find\_nearest\_peak that identifies the row, column coordinates of the closest peak in the Fourier spectrum, given the row, column coordinates of a candidate peak in the Fourier domain. Use this function when trying to identify the center frequency of the notch filters.

**Deliverables & Questions**

1. Completed MATLAB code for HWK3\_SuppressingMoire.m (10 points)
2. How many center frequencies do you need to build the notch filter? What are these center frequencies? (5 points)
3. Screenshots of the Fourier modulus of the image before and after filtering. Label each screenshot clearly. Failure to do so will result in deduction of points. (3 points)
4. Screenshot of Notch filter frequency response (see example above for reference) (2 points)
5. What value of did you choose and why? Provide as much detail as possible. (2 points)
6. Screenshots of the original image and the Label each screenshot clearly. Failure to do so will result in deduction of points. (2 points)
7. Screenshot of difference image and explanation for why the image makes sense. (1 point)

# Image Restoration (25 points)

This portion of the assignment is intended to familiarize you with approaches to compensate image degradations using the Inverse Filter and the Wiener Filter discussed in Lecture-7. Your task is to deblur the images **Image1\_Degraded.tiff** and **Image2\_Degraded.tiff.** You are not required to implement the Inverse and Wiener Filtering algorithms. You will instead be using MATLAB’s built-in function deconvwnr to implement deblurring. Please read the MATLAB help page for this function before attempting to restore the images (<https://www.mathworks.com/help/images/ref/deconvwnr.html>).

Recall that the Inverse filter and the Wiener filter expect knowledge of the blur kernel. In the present case we do not know the blur kernel, also referred to as psf in MATLAB. The term psf is an abbreviation for point spread function and represents the impulse response of the imaging optics. The lectures have been referring to this as the blur kernel .

Since you do not know the blur kernel, you will have to experiment with different choices of the blur kernel. It is recommended that you experiment with a Gaussian blur and different choices of the standard deviation . The MATLAB function fspecial will prove useful in this context. A reasonable choice for the size of the Gaussian blur is . It is also a good practice to pick a filter size with odd number of pixels, so that you have an identical number of neighboring pixels, surrounding the central pixel. The following code snippet may help in your attempts to identify the correct filter size given :

filt\_size = 2\*ceil(3\*sigma\_b)+1; % filter size

The Wiener filter also accepts a nsr parameter which represents the ratio of the noise power to signal power, referred to as the parameter in Lecture-9. The signal power may be estimated as the variance of the image intensities (var(I(:) for an image I). Estimating the noise variance is a bit more involved. Please read the paper “**Fast Noise Variance Estimation**” included with the assignment for details on identifying the noise variance of an additive white gaussian noise degradation.

Recommendations: Please lookup Slide-46 of the Lecture-9 slide deck for potential artifacts that can arise when attempting to restore blurry images using the Wiener filter. Although the slide examines the impact of on Wiener filtering, similar artifacts arise when the blur is estimated incorrectly.

**Deliverables & Questions**

1. Completed MATLAB code for HWK3\_ImageRestoration.m (5 points)
2. Explain in words the approach described in the paper “**Fast Noise Variance Estimation**”. Provide as much detail as possible in your explanation. Why does the paper refer to the filters as Laplacian masks? Do you see any relation between the image derivative operators discussed in Lecture-7 and the masks ? If so, what is it? (10 points)
3. What value of did you end up choosing to model the blur in each image? How did you choose the value? Provide as detailed an explanation as possible, including screenshots of good and bad choices for . (6 points)
4. Screenshots of the image before and after image restoration. Label each screenshot clearly. Failure to do so will result in deduction of points. (4 points)