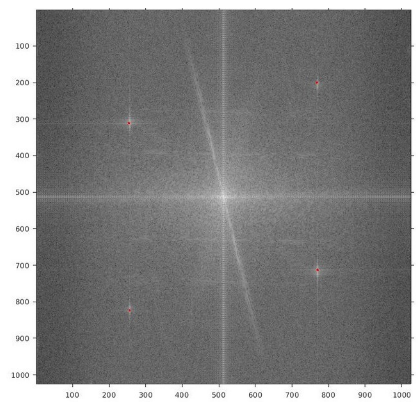
EE420 – Digital Image Processing – Homework 3

Question 1 - Suppressing Moire by notch filtering in frequency domain:

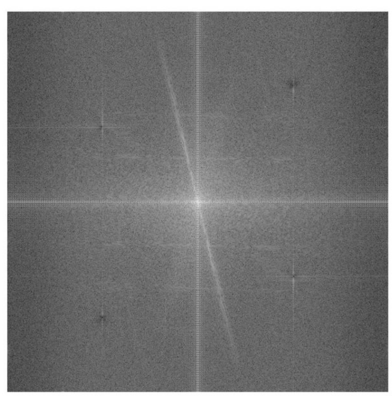
1. The MATLAB code is attached to this PDF in a ‘/code’ folder.

2. We need 4 center frequencies to build the notch filter, because our noise have 2 directions. The 4 center frequencies are +- [713,775] and +- [820,251].

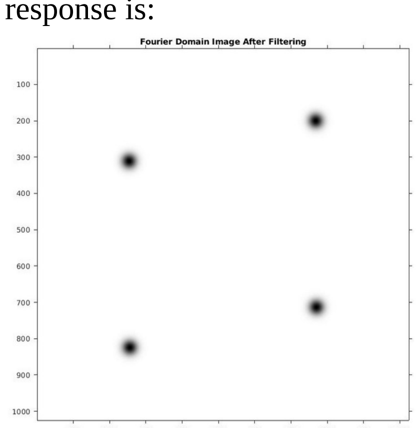
3. The Fourier modulus of the image before filtering:



The Fourier modulus of the image after filtering:



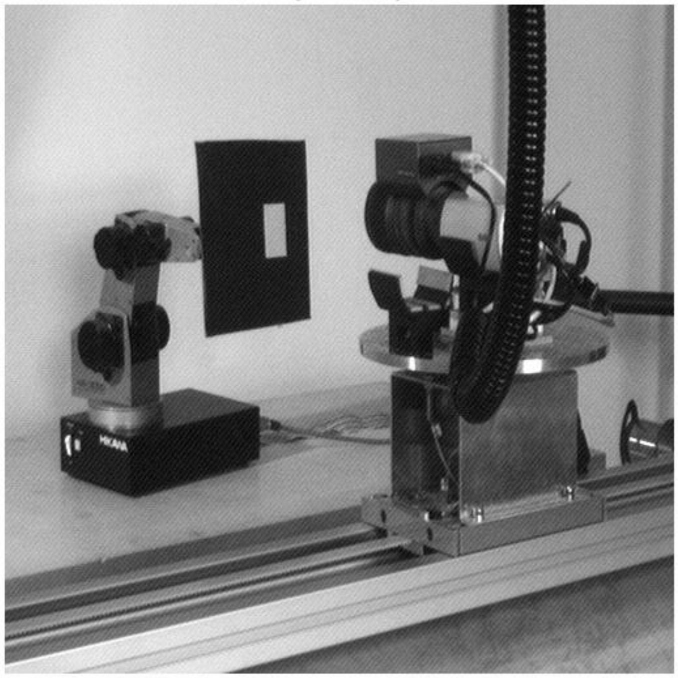
4. The Notch filter frequency response is:



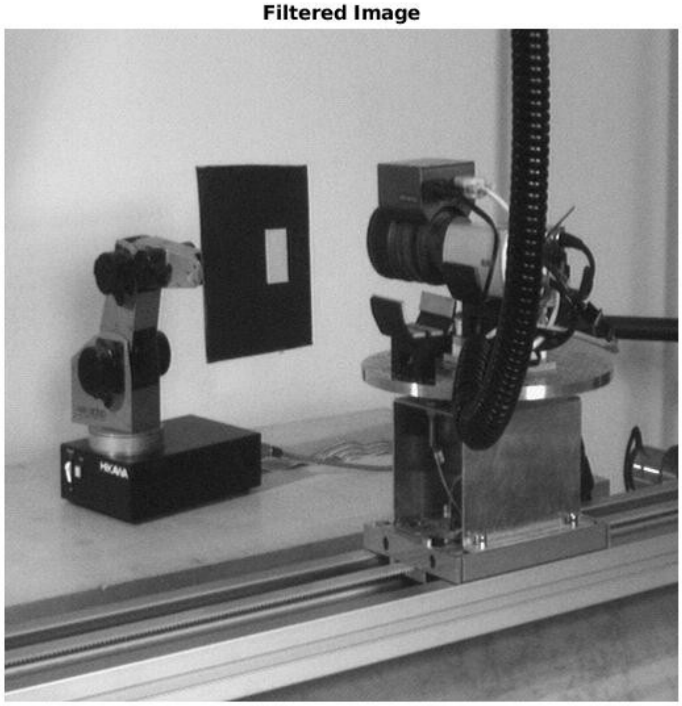
5. The value of D_0 did you choose is 20, and I chose it using trial and error – larger values resulted in lost of image details and with smaller values, the periodic noise was still available.

6. The original image and the filtered image:

Original Image

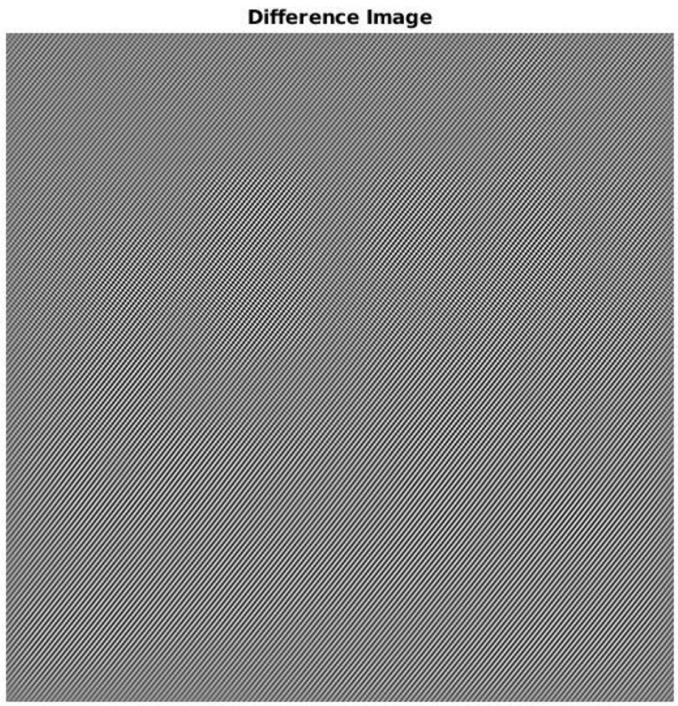


Filtered Image

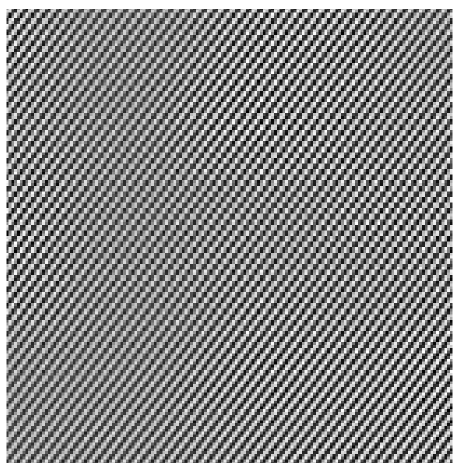


It is easy to see that the filtered image is much smoother and there is no sign for the periodic noise.

7. The difference image:



We can see clearly the two directional periodic lines in the difference image – what make sense, because we saw 2 sets of center frequencies in the Fourier modulus. You can see better the shape of the periodic noise in this enlarge difference image:



Question 2 - Image Restoration:

1. The MATLAB code is attached to this PDF in a '/code' folder.

2. The paper presents simple way of estimating the variance of additive zero mean Gaussian noise, where the variance is not uniform across the image. According to the paper, we can estimating the variance by estimating the standard deviation of the noise (sigma_n).
 The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by:

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

 We use discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. L1, L2 are standard mask for positive Laplacian – they are called masks because they evaluate the Laplacian of the image. The paper refers to the filters L1, L2 as Laplacian masks, because the noise is insensitive to the Laplacian of the image, and therefore, we can subtract the two masks to get the noise estimation operator N:

$$N = 2(L_2 - L_1)$$

 I can see a relation between the image derivative operators discussed in class and the masks L1, L2 – the derivative filter also uses a mask h that that responds strongly to changes in local intensity. I L1, L2 case, they also response to a local intensity change.
 Back to the noise estimation operator N - it has zero mean and a variance of 36*sigma_n^2. Using N we can convolve the image sum the pixel values to find the noise variance:

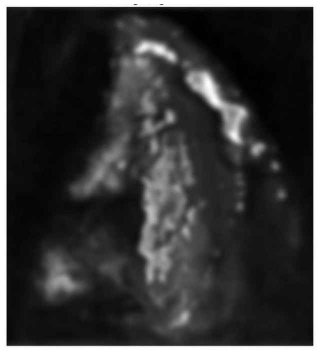
$$\sigma_n^2 = \frac{1}{36(W-2)(H-2)} \sum_{\text{image } I} (I(x,y) * N)^2$$

 A different way to find the noise variance without multiplication

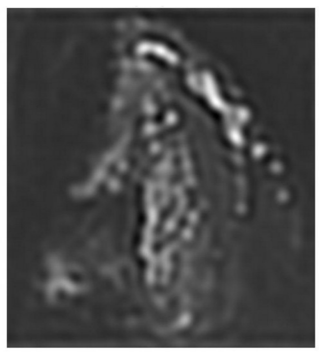
$$\sigma_n = \sqrt{\frac{\pi}{2}} \frac{1}{6(W-2)(H-2)} \sum_{\text{image } I} |I(x,y) * N|,$$

 The author of the paper conclude that the method performs well for a large range of noise variance values, but in textured images or regions, the noise estimator perceives thin lines as noise.

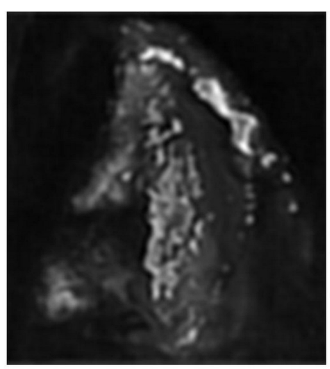
3. I chose sigma_b1 to be 5 and sigma_b2 to be 6. I got those values by Trial and error – I plugged in several different numbers and checked which number gave the best final result.
   In the following images, you can see my good and bad choices of sigma_b (for one image):
   sigma = 1 (too small):



   sigma = 10 (too large):



   sigma = 5 (OK image):



4. A screenshots of the image before and after image restoration: