Image Processing – HW4

Submitters :

Maya Atwan , ID:314813494

Obaida Khateeb , ID: 201278066

Problem 1 – Understanding Fourier

a. We would prefer to scale with the Fourier transform based method. It's less likely to produce artifacts compared to the geometric scaling. In addition, it is better than the interpolation in preserving the sharpness and fine details of the image. However, this not always the case, and sometimes we could prefer the geometric operation method, especially if quality is less critical or if we're limited in time or complexity.

b. Since f,g are the inverse Fourier transforms of F,G respectively, then:

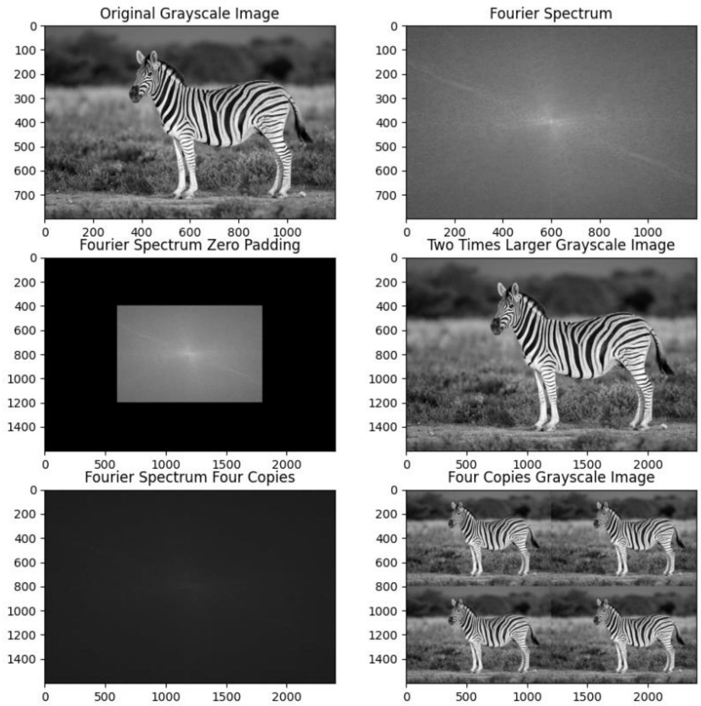
$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v) * e^{-i2\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)}$$

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v) * e^{-i2\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)}$$

Given that F(u,v) = G(u,v) for each u,v then:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v) * e^{-i2\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)} = g(x,y)$$

Problem 2 – Scaling the Zebra



We implemented the 5 parts of the figure (other than the original image) such, and they include:

- **Fourier spectrum representation image** (Top right): we converted the image to the spatial domain using the built-in function fft, then shifted it so the low frequencies placed in the centered and the higher ones toward the border. Then, to visualize it we replaced each cell value x by its 1+log(x).
- **Fourier spectrum with zero padding** (Middle left): this has done by initializing a zero's 2M*2N matrix. Then replacing the M*N pixels in the center by the Fourier transform matrix. To visualize it we used the same trick of computing the log of the values as before.
- **Upscaled image** (Middle right): since it's supposed to be the inverse Fourier transform of the previous image, we used the built-in functions ifftshift and ifft2 to shift it back and build the spatial domain image. We

multiplied the result by 4 in order to adjust its brightness, and ensured that the pixels value lie in the range 0-255 using np.clip function.

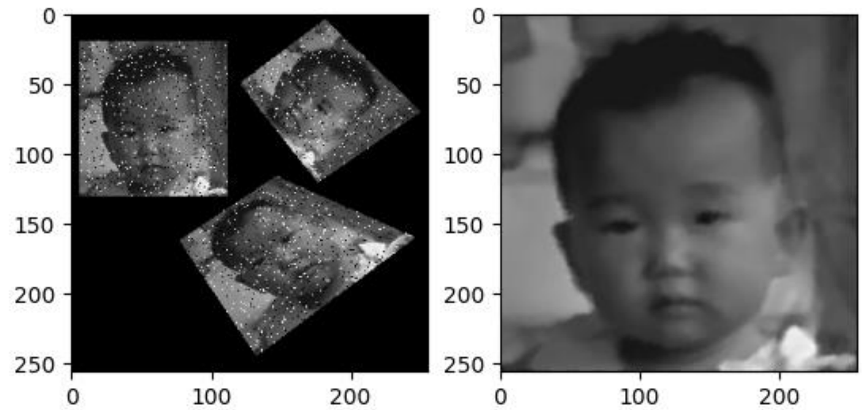
- **Fourier spectrum with four copies** (bottom left): As with the zero padding, we initialized a Zero's 2M*2N image. Then, each Fourier spectrum (i,j) cell value was placed in (2*i, 2*j) cell in the newly created matrix. The result is a matrix that composed of the same rows and columns of the original Fourier spectrum matrix but each two of its rows/columns are separated by zero's row/column. Then, as stated by the Fourier transform scale formula, we scaled each of the amplitudes by 4, and computed the log of the values for the purpose of representation.
- **Four copies Image** (bottom right): because it's the equivalent of the previous image in the spatial domain, we computed the inversed Fourier transform of it using the same method as with the padding.

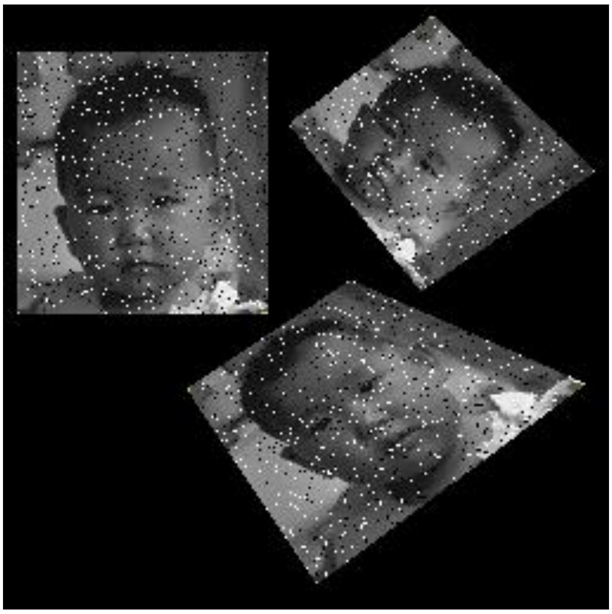
The difference between the two scaling methods is that the zero-padding method produces a smoother upscaled version of the image, where the one received using the "Fourier transform scale formula" produces 4 copies of the same image. Another difference that worth mention is that with the zero-padding we expect artifacts in the image due to the upscaling, where the other one doesn't have such artifacts and maintains the sharpness of the original image.

Problem 3 – Fix me!

1- Baby - There are 3 images, utilize all three:

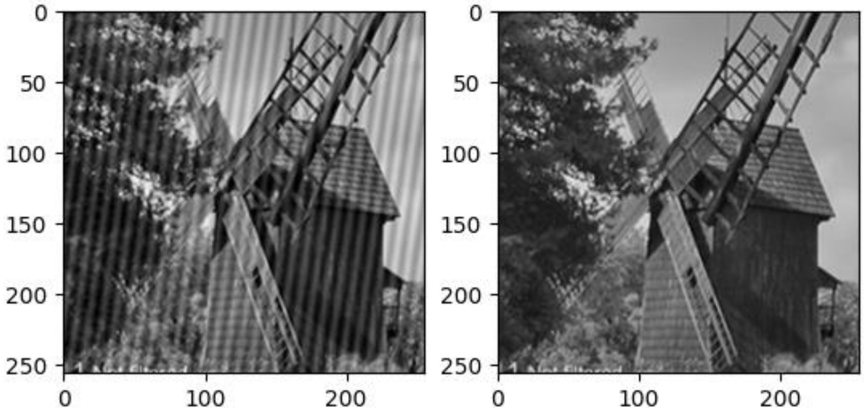
There are 3 images of the same picture but they are noisy(salt pepper noise) and misaligned. We applied median filter to remove the noise (in the image). then extracted the 3 images using geometric operation. (like we did in assignment 2 -the points we choose colored yellow in the second image). At the end we merged all the three images by averaging them.



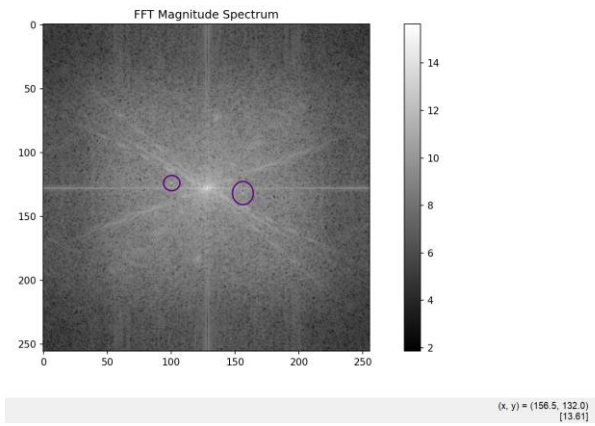


2- Windmill - This is an image with a noise of a very specific frequency:

The image contains noise at a specific frequency so as we learned to remove it we identify the 2 high value points in the fft image then replace their values with the average of the surrounding points to smooth out the noise.

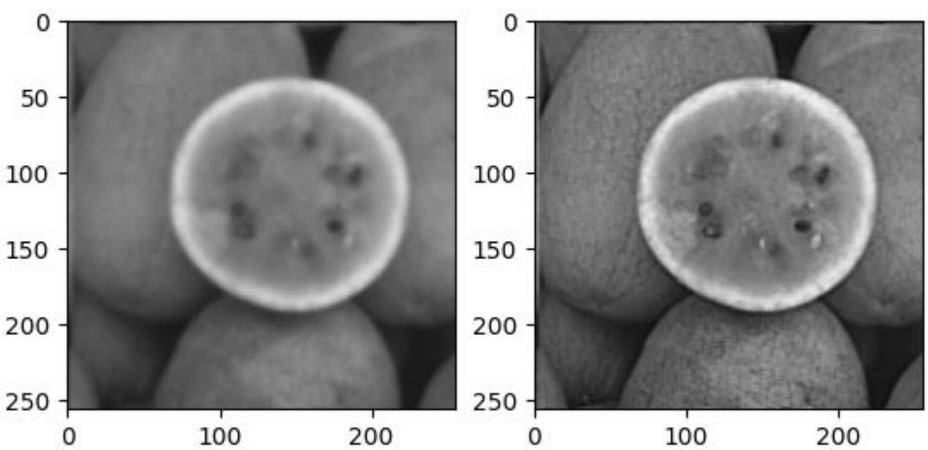


The 2 points we changed their values:



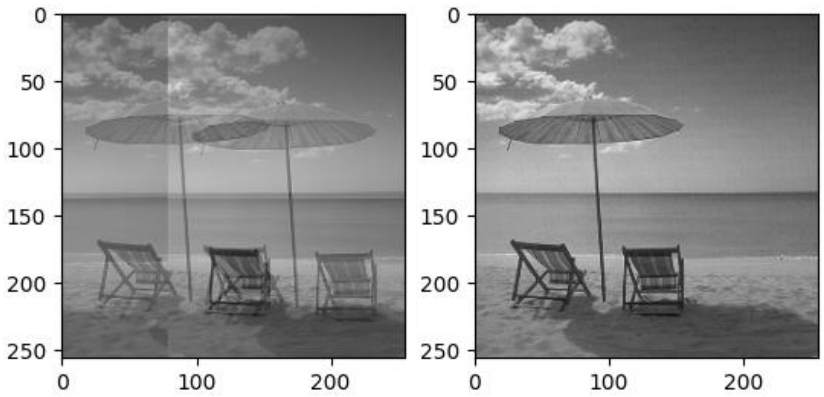
3- Watermelon - Which filter sharpens an image (Went over this in the tutorial):

From the hint and the tutorial we saw that this kernel " [[-1, -1, -1],[-1, 9, -1],[-1, -1, -1]] " enhances image sharpness so we applied this filter to sharpen the image.(this kernel is sum of 2 kernels [[0,0,0],[0,1,0],[0,0,0]] + [[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])



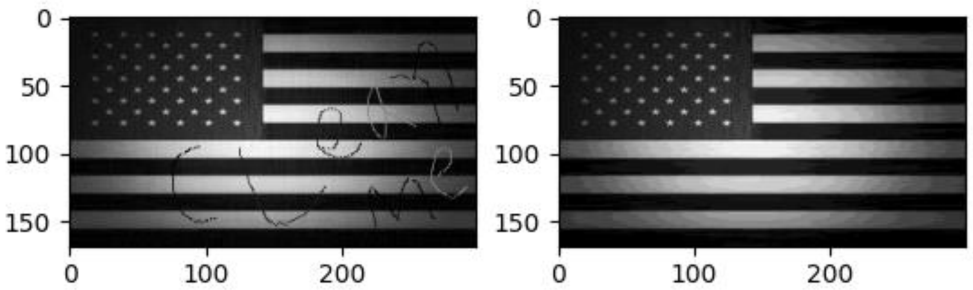
4- Umbrella - The noised image is a result of an average between original image and a shift of the original image:

We used FFT division with mask to remove the unwanted frequency caused by the shift.(The mask to target the unwanted shift frequencies.)



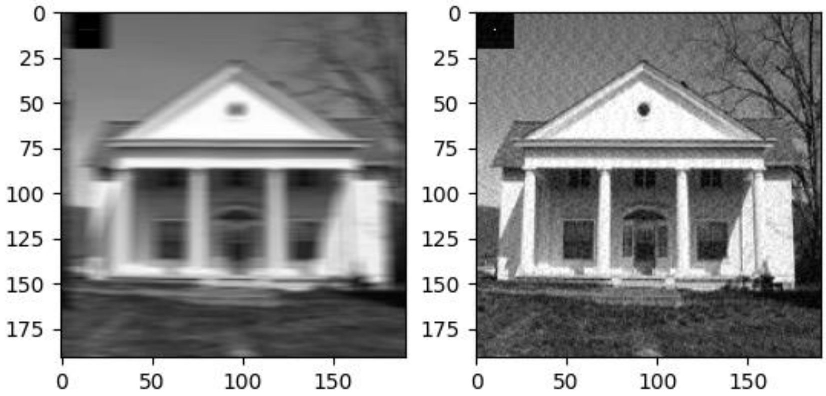
5- USAFlag - Clean me:

The left top corner remains unchanged so we saved it before processing. then we applied median filter on the image to remove the noise and smooth image horizontally .after that we restored the saved corner to maintain its original details.(the size of the corner we check the pixels as we can see its almost 90 * 145 almost)



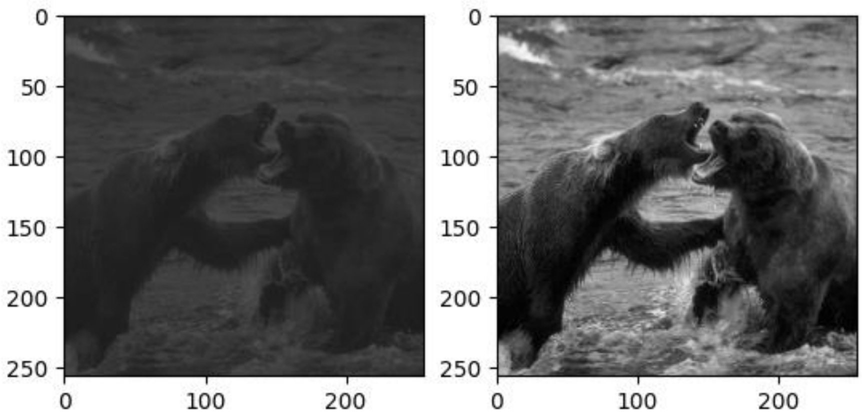
6- **House -** Someone took 10 images of a house while moving (motion blur), the result is 10 shifted images averaged in one

As we saw in the lecture to fix motion blur we used FFT **. to fix this we used FFT division with a mask similar to image 5 to remove the unwanted frequency caused by motion.** The blur result of 10 avg images so we create mask with small values '0.1' at the estimated shift locations to counteract the blur effect.



7- Bears - Pretty dark in here, like all the gray values are low:

From the hint all the gray values are low. to fix this we normalize the gray values using min-max normalization , which scales the pixel values to the range [0,255] and that improves brightness and contrast.



** we tried histogram equalization and the output not good as the normalization min max).