Image Processing: Assignment #4

Problem 1 – Understanding Fourier

a)

Both methods have their advantages and disadvantages, and the "better" method can vary depending on the context:

1. Geometric Operations with Interpolation

Advantages:

- Offers various interpolation methods (nearest-neighbour, bilinear, etc.), each with its own trade-offs in terms of quality and computational efficiency. This allows for more control over the balance between quality and performance.
- Directly scaling pixel coordinates is conceptually straightforward and widely supported by many libraries and software, making it easier to implement and use.
- Typically, faster, and more suitable for real-time applications or when scaling needs to be performed on-the-fly, such as in graphical user interfaces or video processing.

Disadvantages:

- Scaling down significantly or scaling up can introduce artifacts such as blurring, aliasing, or loss of detail.
- Higher-quality interpolation methods are computationally more intensive, which may not be ideal for all applications.

2. Fourier Transform Method

Advantages:

- Scaling in the frequency domain can be particularly effective for certain types of images or signals, where manipulating the spatial frequency components directly can offer advantages.
- Under certain conditions, can provide high-quality scaling with potentially fewer artifacts, especially for downscaling.

Disadvantages:

- Generally, more computationally intensive, especially for large images, due to the need to perform forward and inverse Fourier transforms.
- Can introduce edge artifacts or ringing effects due to the periodic assumption inherent in the Fourier transform. This requires careful handling or post-processing.
- Requires a deeper understanding of signal processing principles.

Conclusion:

For general use, particularly in applications requiring quick and efficient scaling with good quality, geometric operations with interpolation are often preferred due to their simplicity, speed, and the control they offer over the scaling process. They are versatile and can be easily adjusted to meet the needs of different scenarios.

The Fourier transform method, while offering unique advantages in certain specialized applications, is more complex and computationally demanding. It's typically used in more specialized fields or when the specific benefits it offers are directly relevant to the task at hand.

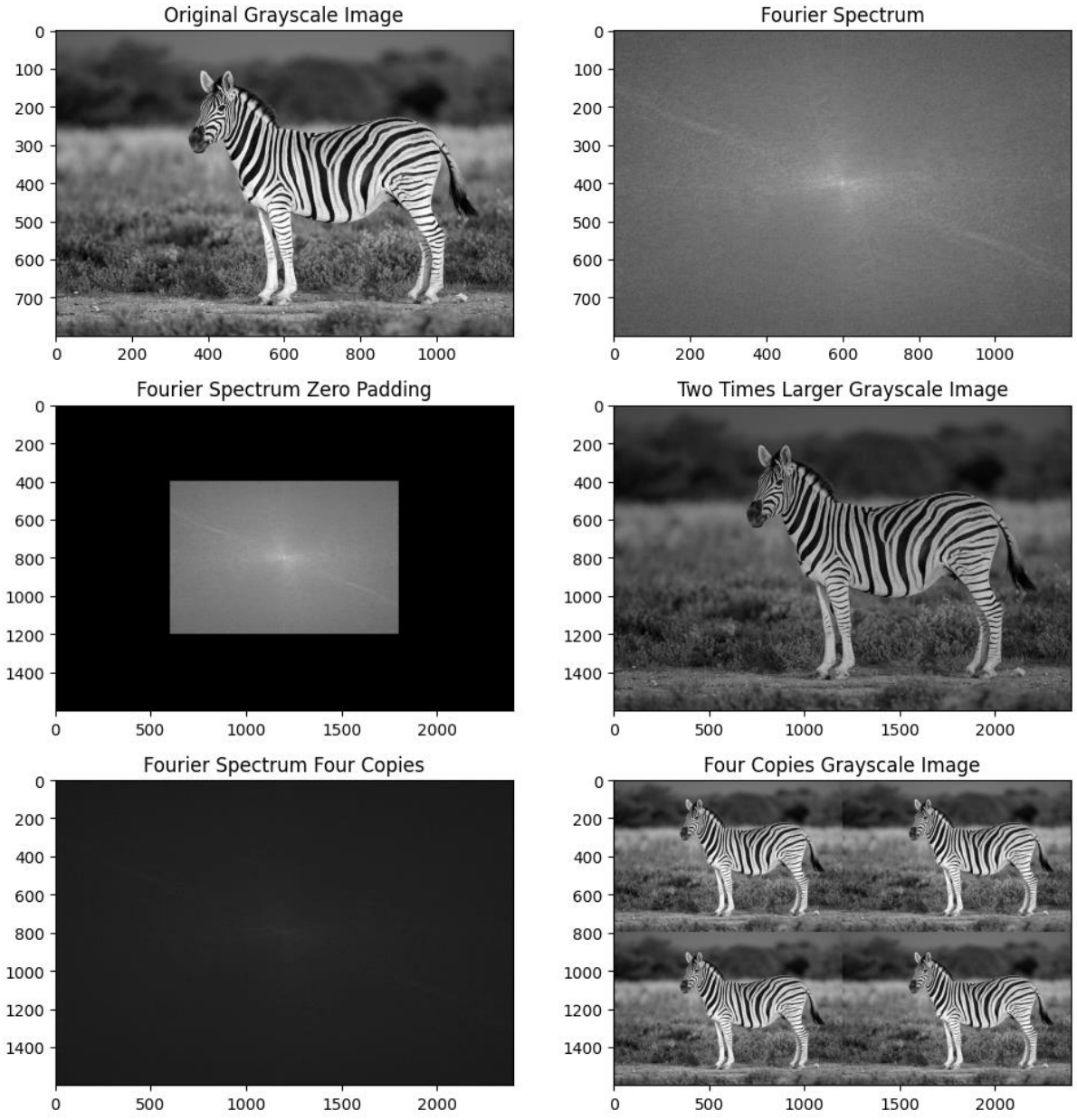
In most practical scenarios, the choice of method will depend on the specific requirements regarding image quality, computational resources, and application context.

Additionally, due to convolution theorem, convolution in one domain is multiplication in the other and vice versa. It can be very useful for example when image reconstruction is involving division or multiplication between the image and some kernel, it is not an operation that can be performed in the domain that is used convolution, but by transferring to the second domain, convolution becomes multiplication, and there, multiplication and division is possible.

b)

The Fourier transform's ability to uniquely represent each image is anchored in its comprehensive conversion of an image's spatial details into a distinct pattern of frequencies. This conversion is not arbitrary; it's underpinned by the Fourier Inversion Theorem. This theorem is a critical mathematical principle stating that the process of converting an image to its frequency representation and then back to its spatial form is lossless meaning every piece of information is retained. Therefore, if two different images resulted in identical Fourier transforms, implying their frequency patterns are the same, it would then follow, according to this theorem, that the original images must also be the same. This foundational principle ensures that the Fourier transform acts as a unique identifier, much like a fingerprint, for each image. It encodes all visual information into frequencies in such a way that no two distinct images can share the same frequency representation.

Problem 2 – Scaling the Zebra



Method 1: Zero Padding in the Frequency Domain

1. FFT Transformation: It converts the image from the spatial domain to the frequency domain using the Fourier Transform.

2. Centring the Transform: It shifts the zero-frequency component to the center of the spectrum.

3. Zero Padding: It adds zeros around the transformed image in the frequency domain, effectively increasing the dimensions of the frequency representation without altering its content.

4. Inverse FFT: It converts the padded frequency domain image back to the spatial domain, resulting in an image that appears larger due to the added zero frequencies. This method essentially interpolates the image in the spatial domain by adding additional, non-information-bearing components in the frequency domain.

Method 2: Doubling the Image Size by Replicating Frequency Components

1. Creating a New Array: A new, larger array is created to hold the expanded image.

2. Replication and Amplification: The original Fourier transform values are placed in the new array, but every other row and column is skipped. This effectively replicates the frequency domain data in a larger space. Additionally, the magnitude is multiplied by 4 to compensate for the spreading of energy across the larger array.

3. Inverse FFT of Padded Array: The inverse FFT is then applied to this modified array to obtain an image that is larger than the original.

Differences Between the Methods

- The first method increases the image size by adding zero frequencies around the original frequencies, effectively interpolating the original image when transformed back to the spatial domain.

  The second method replicates the frequency data in a larger array, intending to create a larger image with repeated content.

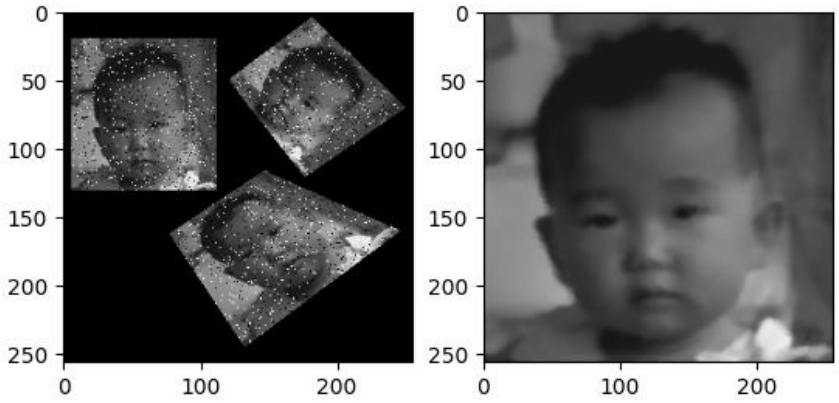
- Zero padding (Method 1) results in a smoothly interpolated image, with the added pixels blending in with the original content.

  Replicating frequency components (Method 2) attempts to enlarge the image by increasing its dimensions while keeping the original content more or less visually intact, but it may introduce artifacts or distortions due to the manipulation of the frequency components and the compensation (multiplication) step.

In summary, both methods offer ways to scale images through manipulation of their frequency domain representations, but they do so with different underlying principles and with different effects on the resulting images.

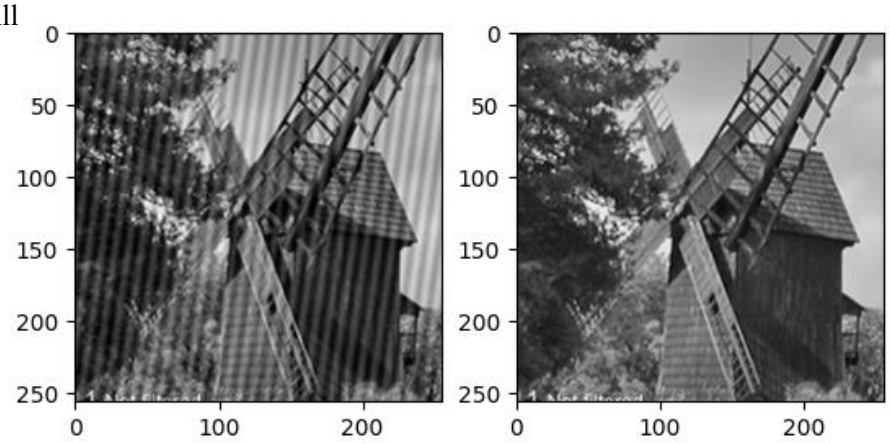
Problem 3 – Fix me!

Baby



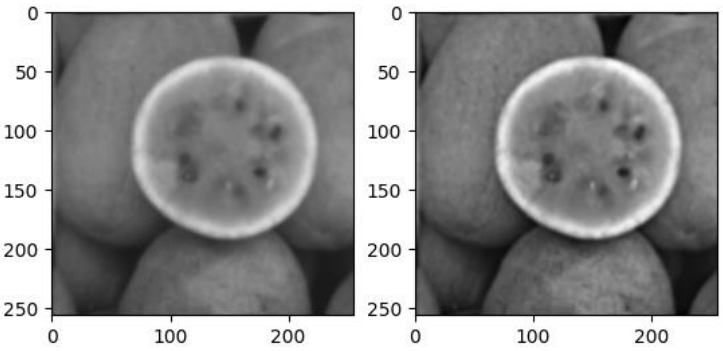
First, to remove the salt and paper noise, we apply medianBlur on the image with kernel in size 3X3. Then we apply perspective transforms on the internal 3 images. We calculate the perspective transform matrix with the corner points of each image to the corners of the 256X256 image. After we get a list of 3 transformed images, we compute the average image from them to reduce the artifacts after the blurring and the perspective transforms.

Windmill



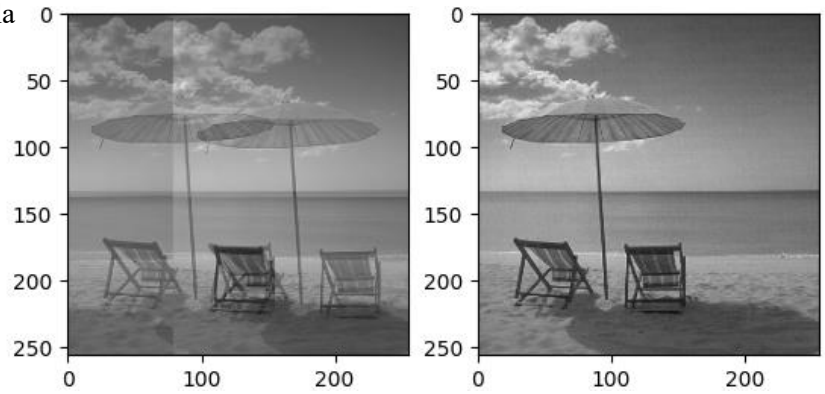
We built a Local Frequency Reject filter to remove the frequency noise. We looked at the image in frequency domain, we found 2 White dots (prominent from their neighbours) that are not the DC, which means that this is the frequency that have high amplitude that is adding the noise in the image. The dotes are in indices (132,156) (124,100) so we built filter in size of the image, with 1 in each index except for those 2 indices that are get the value 0. We apply the filter on the image in the frequency domain to remove those frequencies.

Watermelon



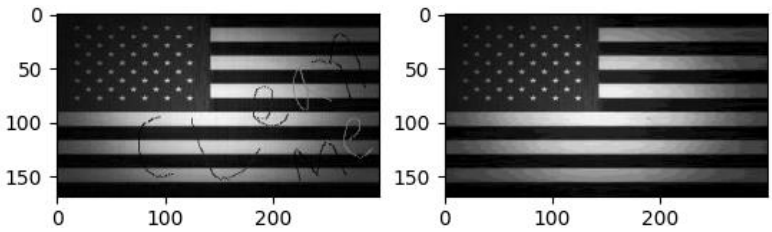
To sharpen the watermelon image, we use the High Frequency Emphasis method with Gaussian High Pass Filter. First, we built Gaussian High Pass Filter with cutoff frequency D0=10, then We gave weight of 1 to the low frequencies to keep them, and we gave weight of 1.5 to the high frequencies. We apply this filter on the image in frequency domain.

Umbrella



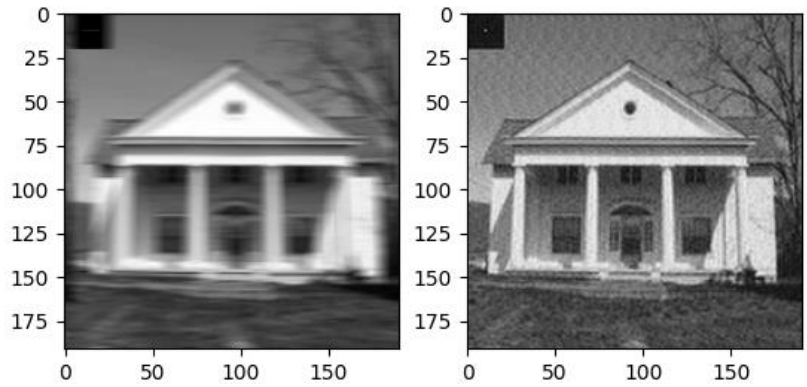
We built filter in size of 5X80 to match the artifact spacing. To build matrix for the delta function that built this image $0.5\big(\delta(x,y)+\delta(x-80,y-5)\big)$, We set 0.5 in the first index and the last index of the filter. We transfer the filter and the image to frequency domain, and we divide the image by the filter. To avoid division by zero or value that is close to zero, we sanitize the filter by replacing insignificant frequencies with a fixed threshold= 1e-5.

USA flag



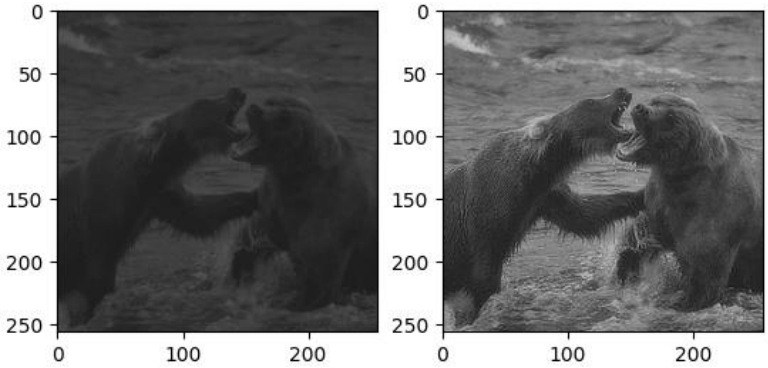
We saved the coordinates of the stars part of the flag - (0, 140, 0, 90) and a copy of the image. Then we apply median filter on x axis (kernel in size 1X10). In the end we reconstruct the image with the stars part of the flag from the copy image (without the filter application) and the stripes part after median filter application.

House



We Built horizontal blurring (averaging) filter in size of 10 and we transfer the image and the filter to frequency domain. In frequency domain we divide the image by the filter. After this fix we got artifact of frequency noise (looks like a grid) to remove it we apply low pass filter in frequency domain.

Bears



Gray values in the image are low, so to increase those values, due to Parseval's equality, we transfer the image to frequency domain. In frequency domain we initialize the mask with the scaling factor of 2 for brightening, but in order to adjust the mask for contrast enhancement we amplify by 2 the high frequencies – those with radius larger than 60 in the image in the frequency domain.