Assignment 4

Samih Warwar, 324888155 – Merry Shalabi, 324007202

Problem 1:

A) When comparing the two methods for scaling an image, geometric operations with interpolation and modifying the Fourier transform followed by an inverse transformation, the choice of the better method depends on the specific requirements of the application, including factors like image quality, computational efficiency, and the presence of noise.

Geometric Operations with Interpolation

Geometric scaling involves transforming the coordinates of the original image to scale it up or down. This can be achieved through methods like pixel replication or interpolation (e.g., nearest neighbor, bilinear, bicubic). Interpolation is necessary for non-integer scale factors and aims to estimate the values of new pixels based on the values of surrounding pixels. This method is straightforward and can be implemented efficiently in both hardware and software. However, it may introduce artifacts, especially with non-integer scaling factors, leading to a slightly smoothed image appearance.

Fourier Transform Scaling

Scaling an image using its Fourier transform involves changing the frequency domain representation of the image and then applying an inverse Fourier transform to convert it back to the spatial domain. This method can handle both upscaling and downscaling efficiently and is theoretically capable of preserving the image's frequency components during scaling. However, the computational cost of Fourier transforms, especially for large images, can be significant. The process of transforming to and from the frequency domain is computationally intensive and may not be suitable for real-time applications.

Comparison and Considerations

Image Quality: Geometric operations with interpolation can introduce artifacts and smoothing, especially noticeable in non-integer scaling. Fourier transform scaling can preserve frequency components better but is sensitive to aliasing and may introduce its own set of artifacts due to the nature of the transform and inverse transform process.

Computational Efficiency: Geometric scaling, particularly with simple interpolation methods, is generally faster and more straightforward to implement than Fourier transform scaling. The latter requires complex computations and is significantly slower for large images, making it less suitable for real-time processing.

Handling of Noise: The method of scaling can affect how noise in the original image is processed. Geometric scaling with interpolation may amplify noise, especially with high scaling factors. Fourier transform scaling can potentially filter out certain frequencies of noise but requires careful handling to avoid introducing artifacts.

Conclusion

In most practical applications, geometric operations with interpolation are preferred due to their simplicity, lower computational requirements, and ease of implementation. This method provides sufficient quality for many applications and can be easily optimized for speed. However, for applications where preserving the frequency content of the image is crucial and computational resources are not a limiting factor, Fourier transform scaling might be advantageous despite its complexity and higher computational cost.

B) The uniqueness of the Fourier transform for each function or image is a fundamental property that stems from the mathematical principles underlying the Fourier transform itself. This property ensures that for any given function (or image), there is a unique Fourier transform, and conversely, a given Fourier transform corresponds to a unique function. This is crucial for applications in image processing, signal analysis, and various fields of engineering and physics.

Mathematical Basis

The Fourier transform decomposes a function (or image) into its constituent frequencies. For images, this involves transforming the spatial domain representation of the image into the frequency domain. The Fourier transform is defined by the integral:

$$F(u, v) = \iint f(x, y)\, e^{(-j2\pi(ux + vy))}\, dx\, dy$$

where $f(x,y)$ is the image in the spatial domain, and $F(u,v)$ is its representation in the frequency domain. The inverse Fourier transform, which reconstructs the spatial domain image from its frequency domain representation, is given by:

$$f(x, y) = \iint F(u, v)\, e^{(j2\pi(ux + vy))}\, du\, dv$$

Uniqueness Explained:

Bijective Mapping: The Fourier transform, and its inverse establish a bijective mapping between the spatial domain and the frequency domain. This means that each spatial domain function corresponds to a unique frequency domain representation and vice versa.

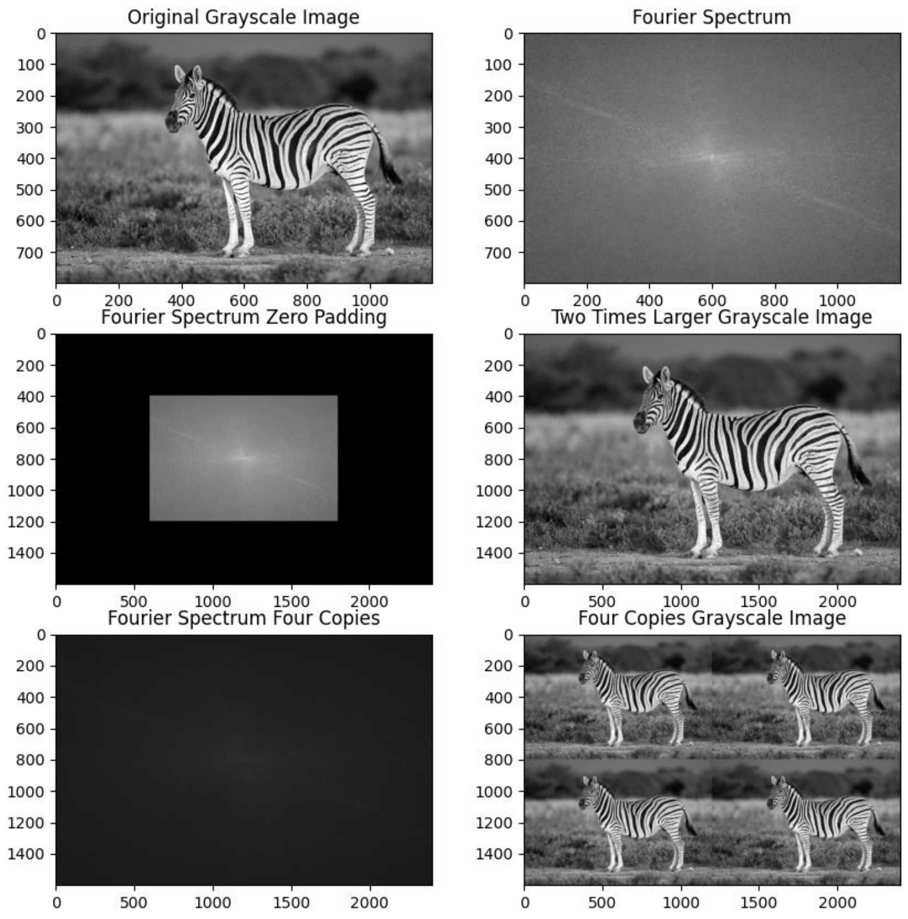
Information Preservation: The Fourier transform preserves all information contained in the original function. The magnitude and phase of the Fourier transform encode the amplitude and positional information of the constituent frequencies of the original function. This comprehensive representation ensures that no information is lost in the transformation process, allowing for the exact reconstruction of the original function from its Fourier transform.

Mathematical Proofs: The uniqueness of the Fourier transform is supported by mathematical proofs. For instance, if two functions have the same Fourier transform, their difference would have a Fourier transform that is identically zero. Mathematical analysis shows that this implies the difference between the two functions is zero almost everywhere, meaning the functions are identical almost everywhere.

Practical Implications: In the context of image processing, this uniqueness property means that every image has a distinct frequency domain representation. This allows for various operations to be performed in the frequency domain (e.g., filtering, compression) with the assurance that the original image can be accurately reconstructed.

In summary, the uniqueness of the Fourier transform for each function or image is a direct consequence of the mathematical properties of the Fourier transform. This uniqueness is crucial for the accurate analysis, processing, and reconstruction of signals and images in various applications.

Problem 2:

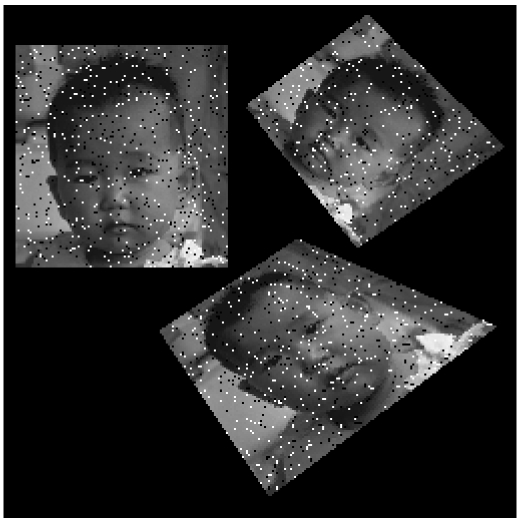
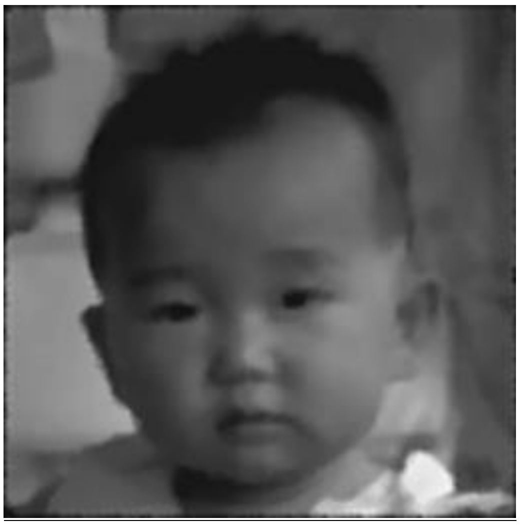


In the first scaling method we padded 0's the Fourier version of the image that means we added low frequencies where they equal to 0 therefore the image will be 4 times bigger (h*w)→(2*h*2*w) and that's why the pixels will be 4 times darker, so we decided to multiply them by 4 in order to keep the same color average as before.

In the second scaling we want to make four copies of the zebra so for each direction (x,y) we have to double the frequency and double the size of the picture itself (4*h*w) we can archive that by adding 0 between every two cells in the Fourier image of the original zebra image and then multiple by 4 for the same reason we did before.

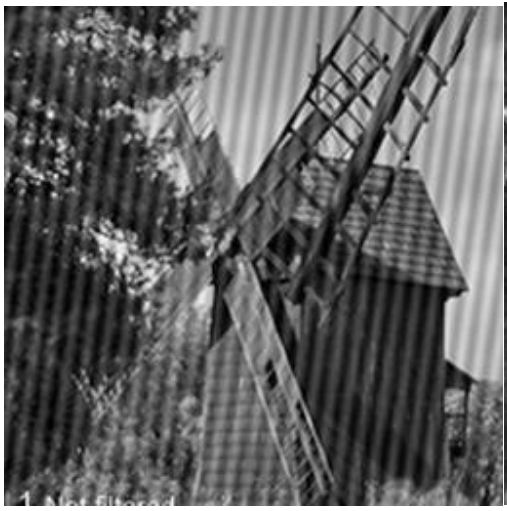
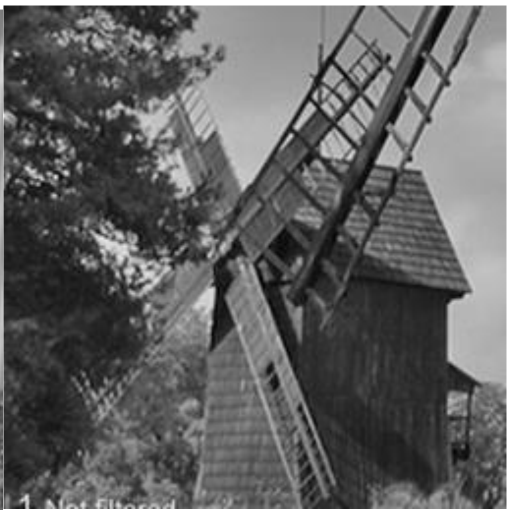
<u>Problem 3:</u>

Baby:

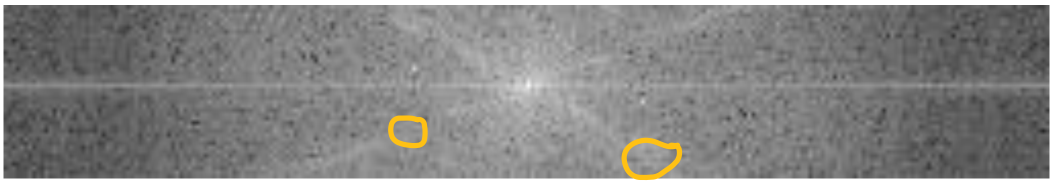



In the baby image we extracted the three pictures using geometric operations and then did median filter like we did in the third problem third assignment.

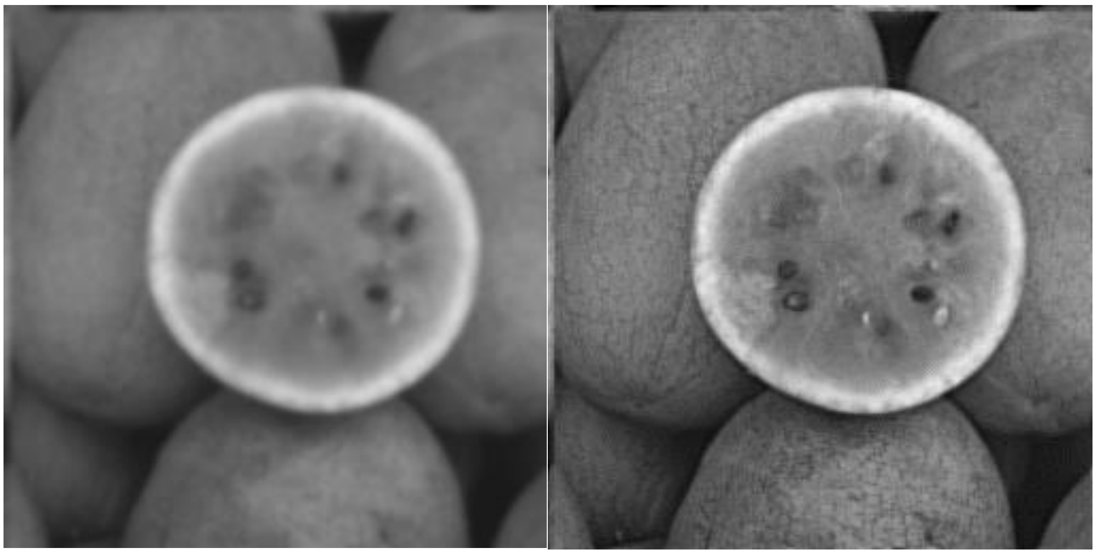
Windmill:

In this image we had to find the frequency and remove it, so we transformed the image into furrier and got two shining points that need to be removed.

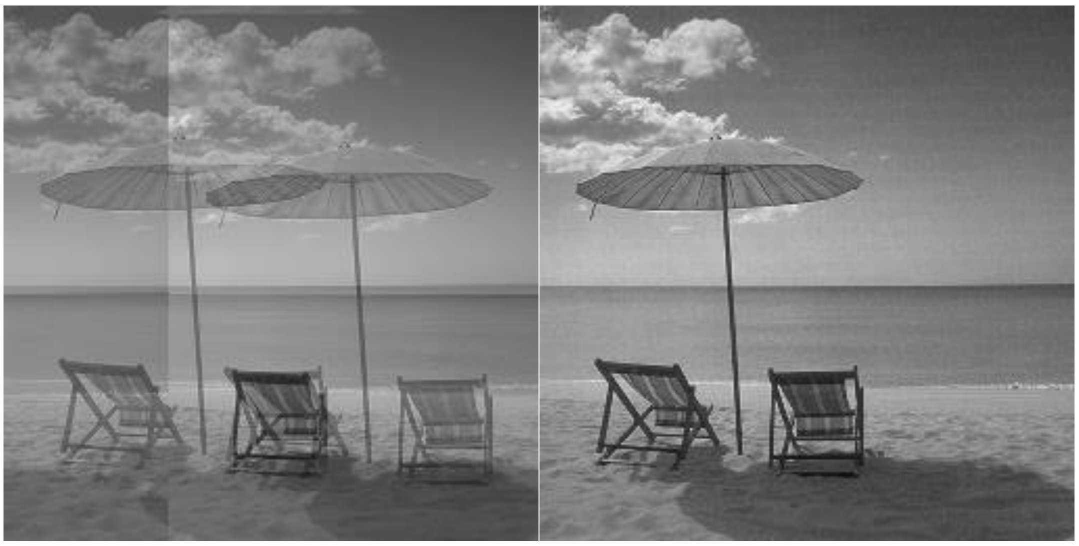


Watermelon:



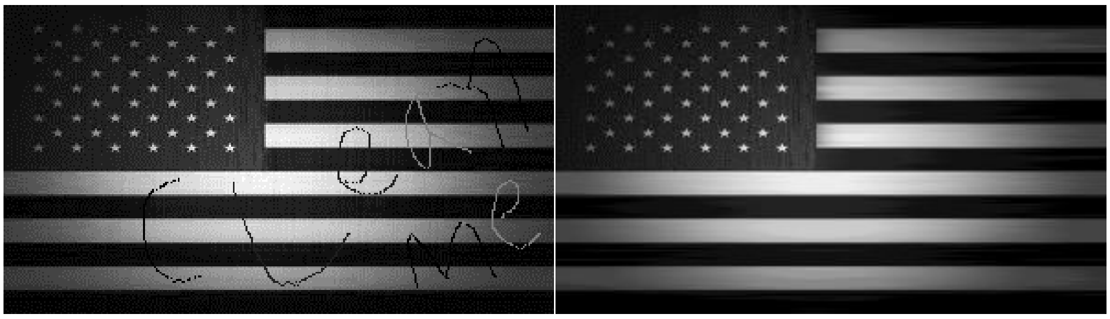
We can see that the image is blurred so to fix it we sharpened it by using a sharpening kernel [[0, -1, 0], [-1, 4, -1], [0, -1, 0]] * 3.

Umbrella:



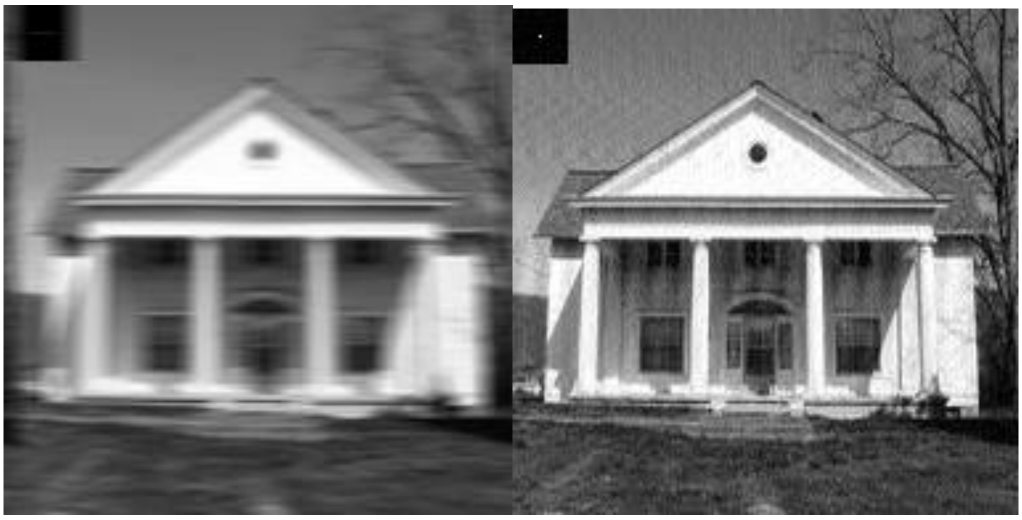
As the hint says this image is a result of an average between the original image and a shifted one. So if we want to reverse the effect we can divide the furrier transformation by a transformed kernel where kernel_value[0][0] = 0.5 kernel_value[4][79] = 0.5, we can find the second point by paying attention to the line in the top right area of the image and then we set the value to 0.5 to protect the DC, but we can see there some noise will appear because of the transformation that we can get red of by putting 1 in every |value| that is less than 0. 0000009 in order not to get high frequency values.

USAFlag:



Well, we have to clean “clean me” we can see that it doesn’t affect the right top area, so we cut that part and place it at the end, in between we applied median filter in horizontal way 50pixels to be specific and then a gaussian filter to remove the remaining noise.

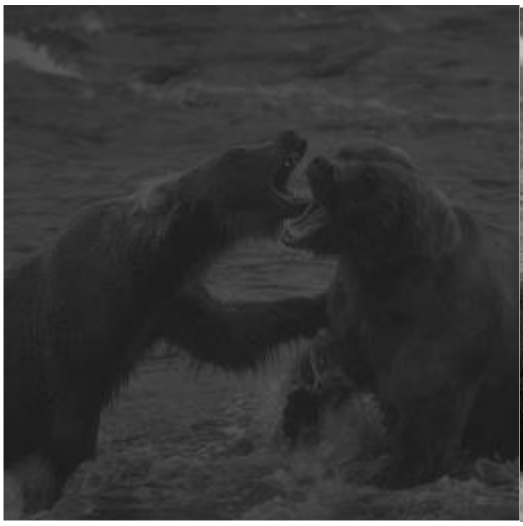
House:

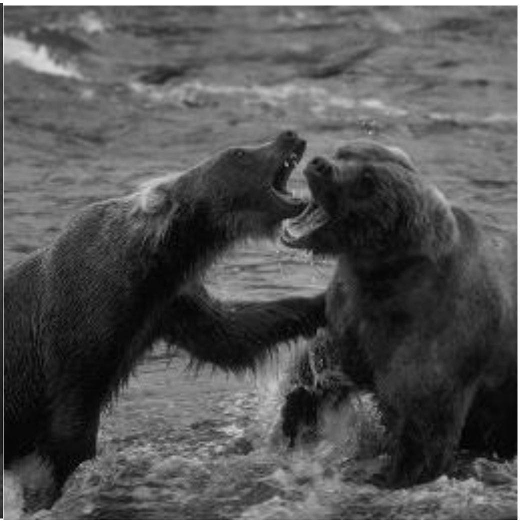


Like the umbrella image but now we have 9 shifted images of the original one. With the help of the little square we can see the top right corner of the 10 images so this time the kernel we want to divide by have the first 10 cells of the first row with value=0.1 and the rest 0, then we do the furrier transform and divide the furrier transformed image by it (as we did before to prevent high frequency values we do here by setting every value in the transformed kernel that [|value|<0.01]=1)

Note: we put 0.1 into the 10 cells because we want to protect the DC

Bears:





We have a dark image of bears so like we did in assignment 2 we changed the contrast and the brightness by using the same function we applied before we chose to increase the contrast and making thee darker areas darker the bright ones brighter and then decreasing the brightness to make it darker.