***Question 1:***

a) In most cases, using geometric operations to scale pixel coordinates with interpolation is the preferred method for scaling an image. There are several reasons for this:

**Efficiency**:

Geometric operations with interpolation are generally faster compared to Fourier transform methods. The geometric operations directly manipulate pixel coordinates and values without requiring complex mathematical transformations like Fourier analysis.

**Accuracy**:

Geometric operations with interpolation tend to preserve image details more accurately compared to Fourier transform methods. Interpolation methods such as bilinear or bicubic interpolation can effectively estimate pixel values in the scaled image based on neighboring pixel values, resulting in smoother and more visually appealing results.

**Ease of implementation**:

Geometric operations with interpolation are conceptually simpler and easier to implement compared to Fourier transform methods. Algorithms for geometric scaling are widely available and well-understood, making them more accessible for practical image processing tasks.

**Robustness**:

Geometric operations with interpolation are generally more robust to variations in image content and characteristics. Fourier transform methods may encounter challenges with images containing sharp edges, high-frequency patterns, or irregular textures, leading to artifacts or distortions in the scaled image.

While Fourier transform methods have their applications in specific scenarios, such as certain types of image analysis or manipulation, for general image scaling tasks, geometric operations with interpolation are usually preferred due to their efficiency, accuracy, ease of implementation, and robustness.

b)

The uniqueness of the Fourier transform for each image stems from the fundamental properties of the Fourier transform and its relationship with the spatial domain (the original image) and the frequency domain (the Fourier transformed image).

The Fourier transform is a mathematical operation that decomposes a function or signal (such as an image) into its constituent frequencies. It expresses the function in terms of sinusoidal waves with different frequencies and amplitudes.

Here are a few reasons why the Fourier transform is unique for each image:

**Representation of Spatial Information**:

Each image consists of a unique arrangement of pixels, each with its own intensity or color value. The spatial distribution of these pixel values forms a unique pattern that represents the image content.

**Frequency Composition**:

The Fourier transform decomposes the image into its frequency components, representing how much each frequency contributes to the overall image. Since the spatial arrangement of pixel values in each image is unique, the frequency composition of each image is also unique.

**Mathematical Definition**:

The Fourier transform is defined as an integral over the entire spatial domain. This integral considers the entire image function, accounting for all the spatial variations in intensity or color. Therefore, the Fourier transform captures the complete information present in the image.

**Orthogonality of Basis Functions**:

The sinusoidal basis functions used in the Fourier transform are orthogonal to each other. This orthogonality property ensures that each frequency component contributes independently to the overall representation of the image.

As a result of these factors, the Fourier transform uniquely characterizes each image based on its spatial distribution of pixel values and their corresponding frequency components. If two images have the same Fourier transform, it implies that they share the same spatial distribution of pixel values, making them essentially identical in terms of their visual appearance and content.

***Question-2:***

As we can see here in the image the values of the restored image are 2 times bigger horizontally and also 2 times bigger vertically.

*Double Size Image:*

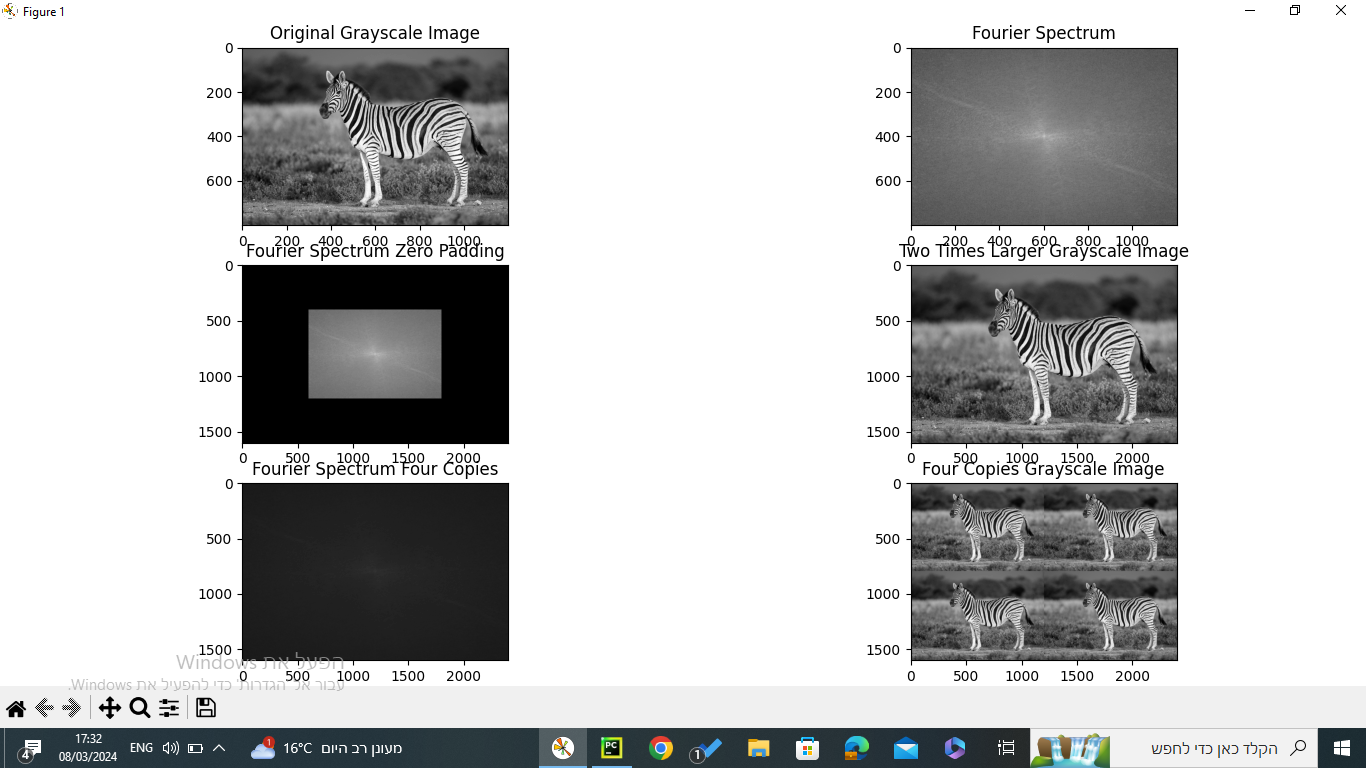
A screenshot of a computer

Description automatically generated

*4 Copies Image:*

A screenshot of a computer

Description automatically generated

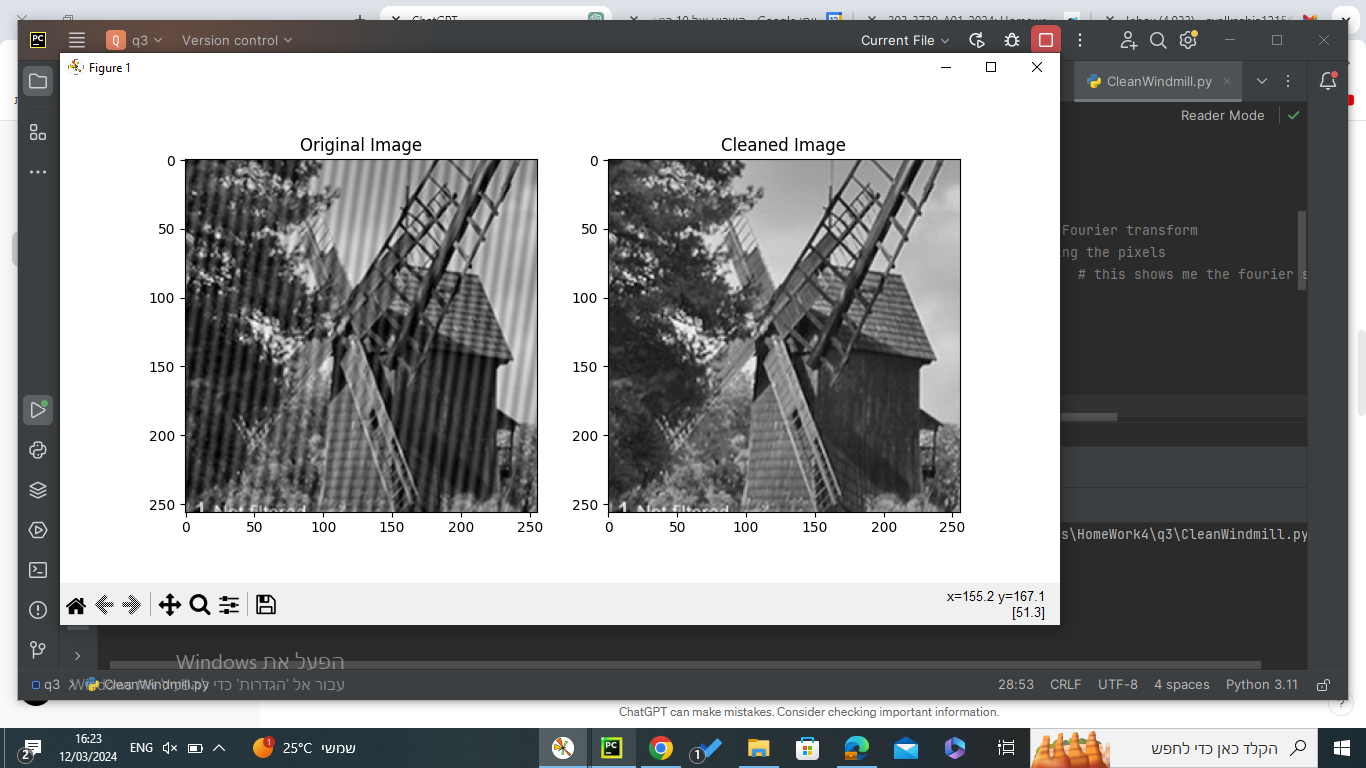


As we can see the difference between the 2 images is how we did the zero padding, in the double sized picture we had zero padding all around the fourier transform and that was what caused the image to get 2 times bigger meanwhile in the 4 copies we added the zeros between every 2 coefficients that were originally neighbors in the original fourier transform.

***Question 3:***

Windmill:

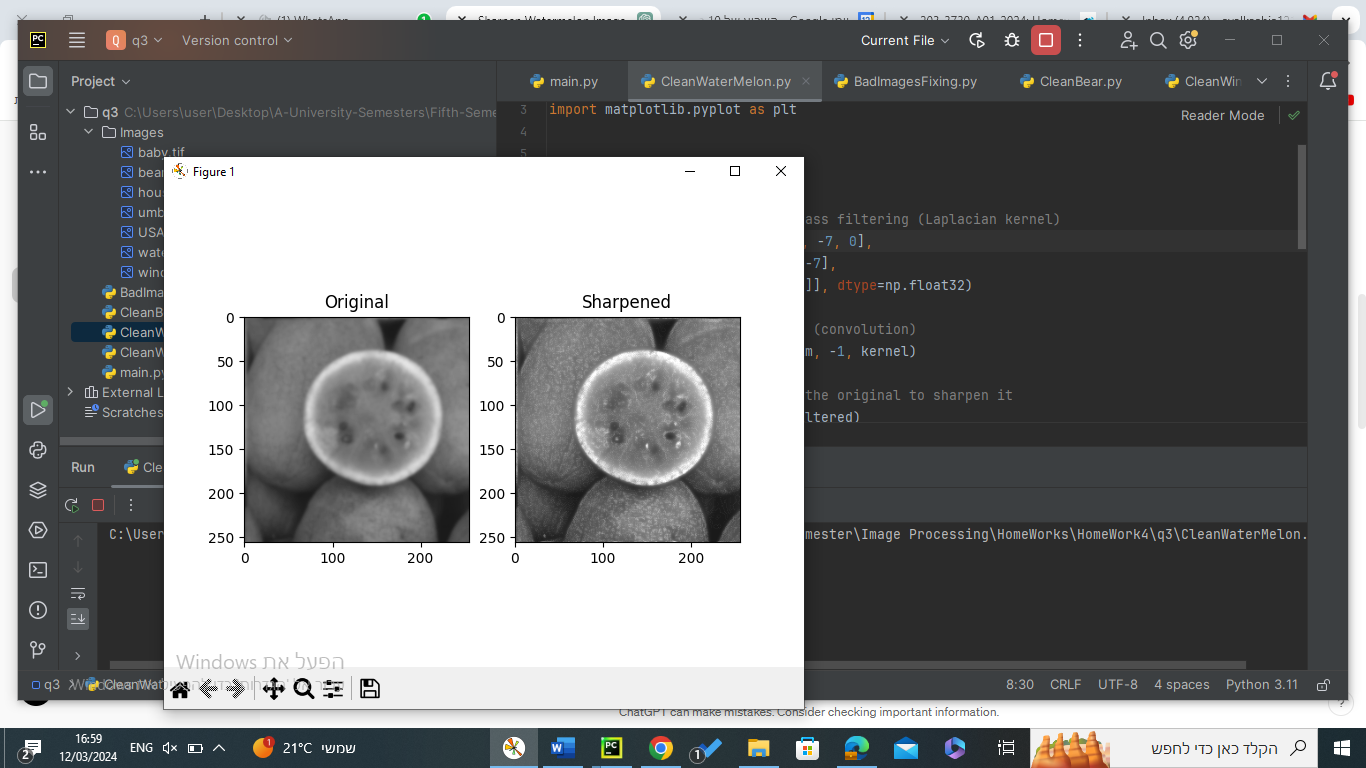
We got the fourier transform of the original image and then we identified the 2 symmetric peak pixels that represent higher than usual frequencies that are causing the noise using paint and then after it we zeroes out those pixels to fix the noise. Exactly as described in lecture 6 and tutorial 7.



Watermelon:

We fixed the watermelon using Laplacian sharpening filter to get the edges more defined and to look properly. We used the kernel=[[0,-7,0]

[-7,28,-7][0,-7,0]] to do the sharpening effect.



Bears:

We will normalize the brightness and increase the contrast to fix the dark image since the original dark image seemed to have all of the pixels kind of darker than usual so using   
clean\_im\_cv2 = cv2.normalize(im, None, alpha=0, beta=255, norm\_type=cv2.NORM\_MINMAX)

we can equalize the grayscale values but also keep the picture in the same way.

A collage of images of bears fighting

Description automatically generated

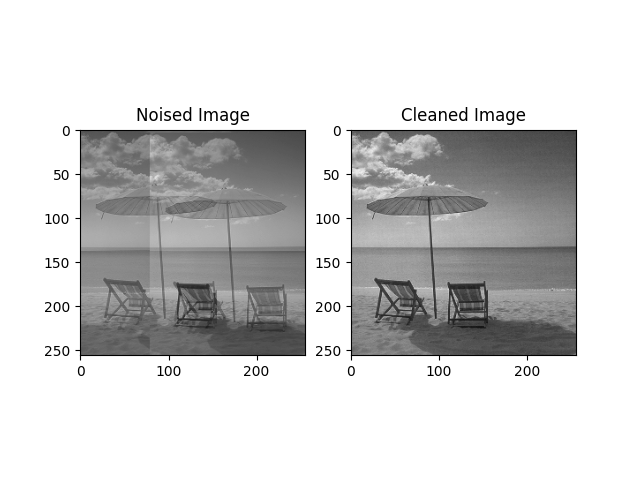
Umbrella:

We will use in this question the way we use in tutorial question 8

for the given image:

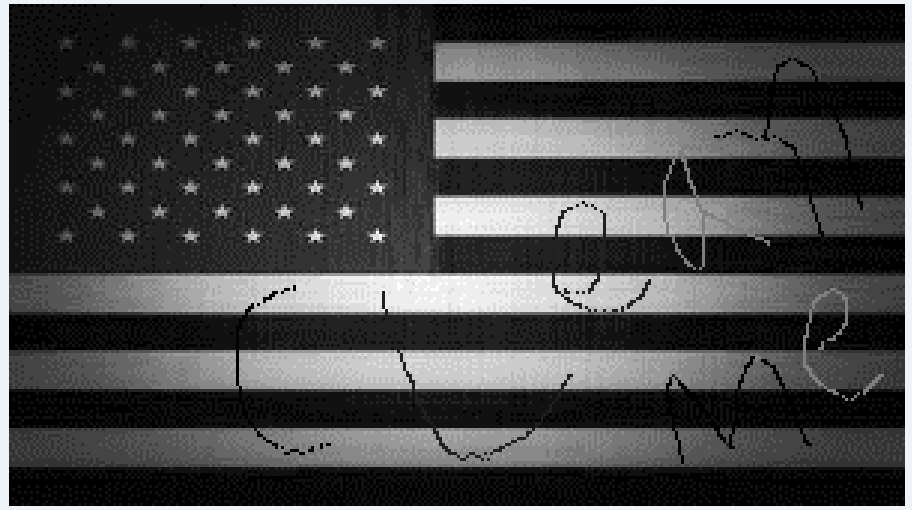
We calculate x and y by the distance between two points in the image

And then do



Usa flag:

As we see in the picture places that need no changes

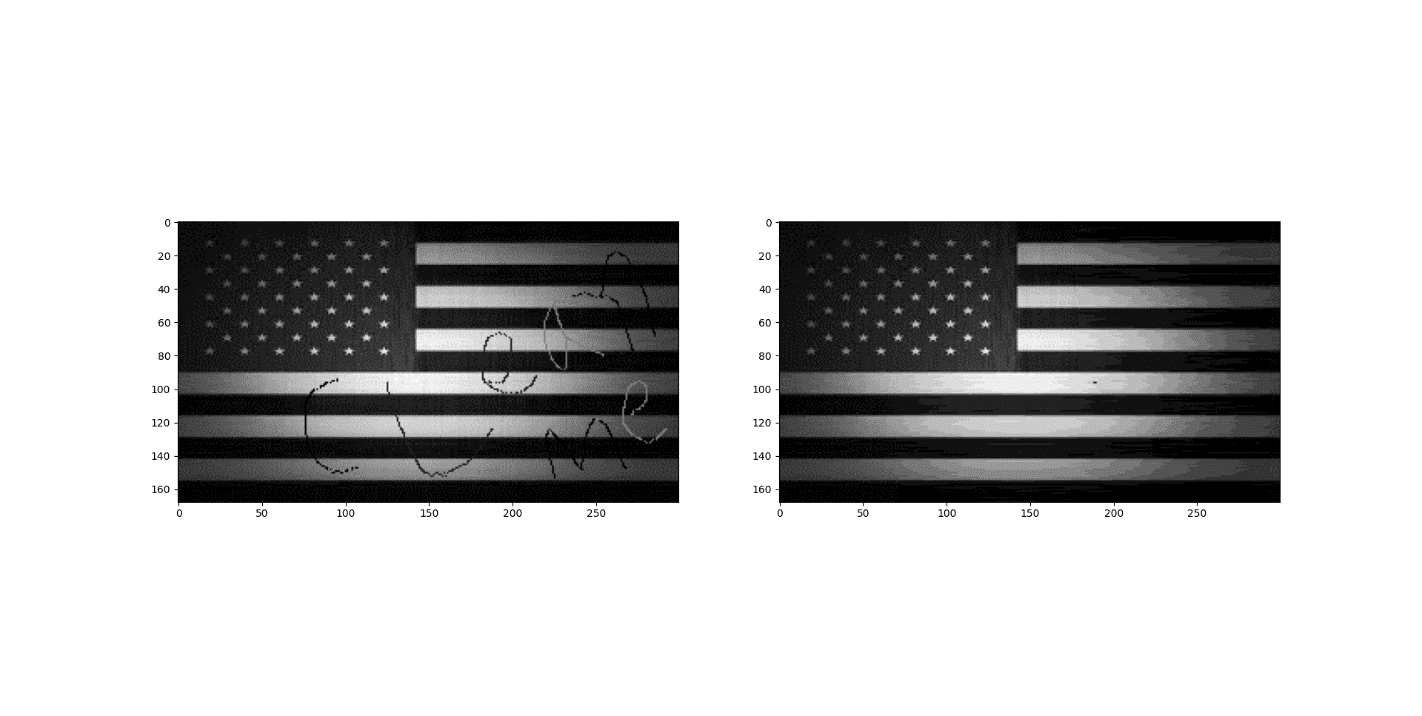




So we will copy them the same for the result image and then we will do something like cv2.medianblur() but in more specific our window will function in rows :

clean\_im[i][j] = np.median(im[i, j - r : j + r + 1])

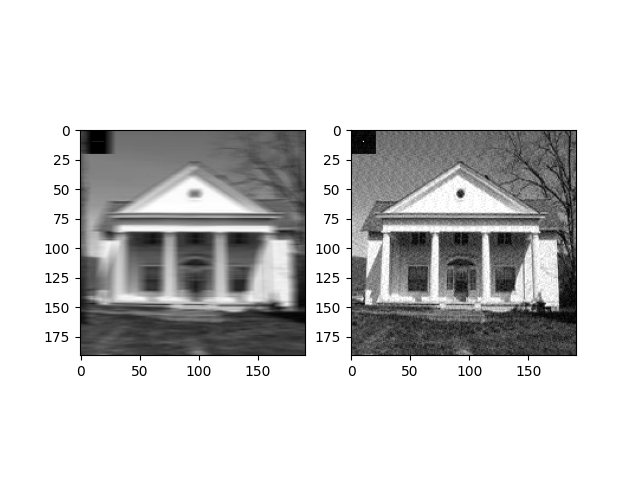
the cleaned image :



House:

We will do horizontal motion blur effect to the given image using Fourier transformation doing this by defining a blur mask that contain in the first

Row we put 1/number\_of\_images which is 0.1 (the averaging effect across these images) and then we apply the Fourier and this going to simulate the frequency response of the horizontal motion blur and then:

F(image)/F(blur\_mask) = F(clean\_image)

Baby:

In this first we extracted the three picture of the baby in the image and put each one of them in full picture (we did warpPerspective)

And then we use a bilateralFilter and medianBlur filter to clean each image at the end we used the median for all three images

