Mateusz Drygiel 249109

Artificial Intelligence and Computer Vision Lab
Homework 4

1. Image stretching for quantized grayscale low and high frequency images

to perform the stretch of an image I first did the quantization of it by writing:

```
#TASK 1 AND 2

#defining the quantization of the image
def Quantizing_img(img, intense):

    adjust = 256 / intense
    adjust = int(adjust)
    img_quant = img.copy()
    rows, cols = img.shape[:2]
    contrast = img.std()
    for row in range(0,rows):
        for col in range(0,cols):
            index = img_quant[row,col] / adjust
            index_after_quant = int(index) * adjust
            img_quant[row, col] = int(index_after_quant)

    return img_quant
```

tin here the image and it's intensity level gets passed as "img" and "intense", then the quantization is performed by adjusting the level and doing so for each point in the image. After this we are receiving back "img_quant" which is the "img" after the quantization with desired level is performed.

When it came down to Image stretching I did:

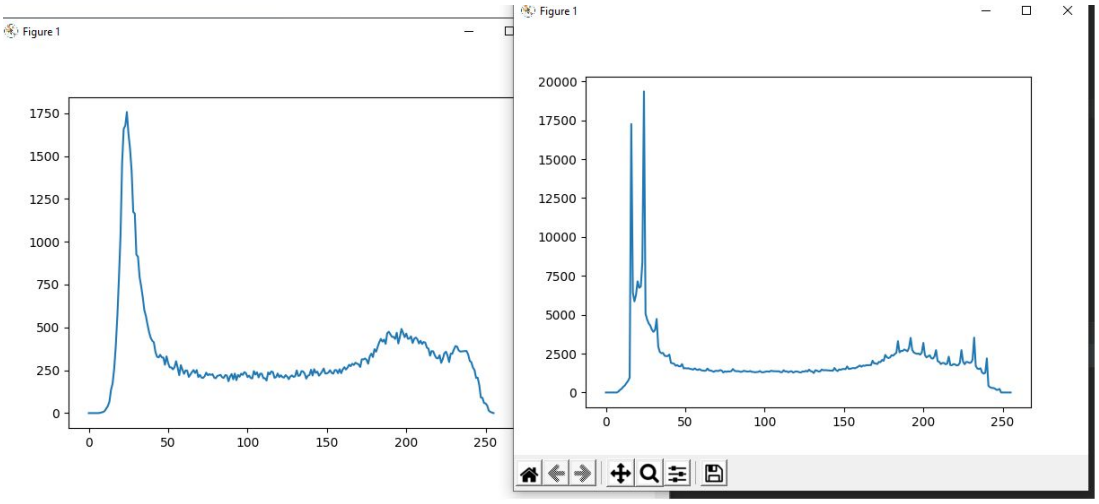
```
#defining the image stretching process
def Stretching_img(img):

    cv2.imshow('Before Stretching', img)

    img_stretch = cv2.resize(img, (700,700))
    cv2.imshow('After Stretching', img_stretch)
    return img_stretch
```

This of course just takes the “img” we chose and then performs the resize of this image, after this is done the result “after” is being shown, prior to that the user sees the image before the stretch is performed.

Because stretching the recognition if it is high or low frequency does change as it is more difficult to decipher.
Here is a brief comparison of quantization with the two images of my choice

From my point of view I can definitely say that the more quantized the image, the more vibrant and soft the colours and edges seem to be, as the number of distinct colours is reduced.
This makes the images appear a bit more flat and plain to their original counterparts.

Here is the image of before and after histogram stretching was performed:



The histogram seems to be more “stretched” across the board after the stretching was performed, the spikes are more dumbed down as a result.

When doing the low frequency image the difference with quantization was very small. My conclusion is that the quantization still does perform but due to lack of variance in the image itself.

2. Histogram equalization for quantized grayscale low and high frequency images

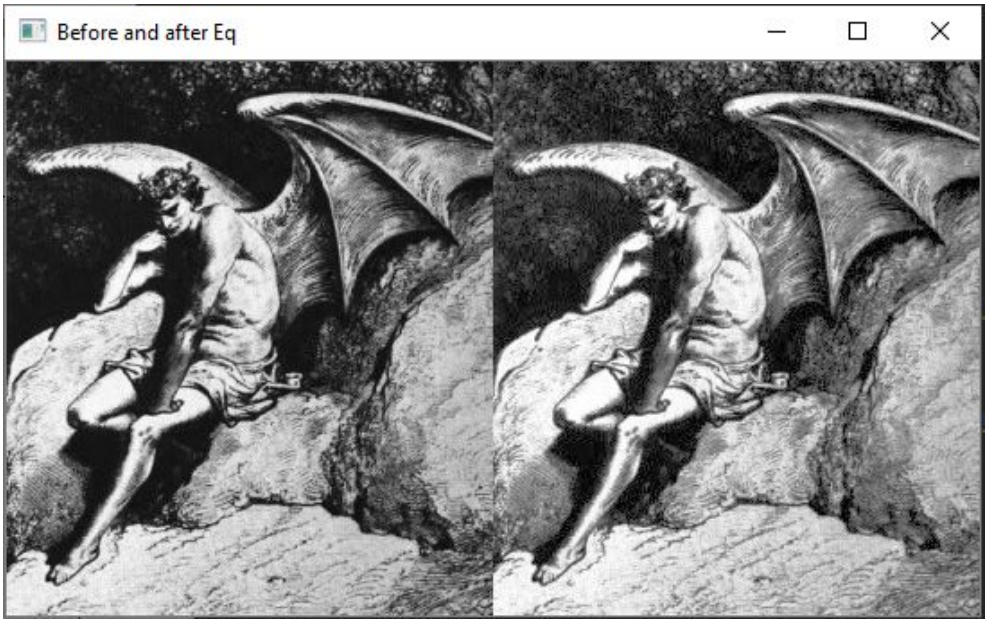
For histogram equalization the first step of quantization remains the same, however the equalization portion is the following:

```
#histogram equalization of the image
def Histogram_eq(img):

    equl = cv2.equalizeHist(img)        #equalization histogram
    eqresults = np.hstack((img, equl))  #making it so that the images appear next to eachother
    cv2.imshow('Before and after Eq', eqresults)
    return eqresults
```

In this code the “img” is passed, then it’s histogram is equalized and the before and after images of “img” are shown.
The results are the following:

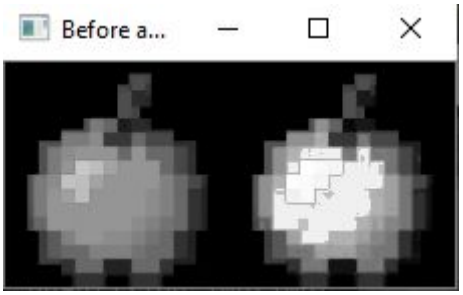
-before and after when quant is 32

-before and after when quant is 128



(it seems the 128 quant photos do not really represent the state it really looked like on the screen, the photos on the right were a bit brighter than the ones seen above)

In high frequency images (like the above examples) we can see that the contrast has been changed, mostly due to the fact that in these types of photos there are a lot more details than in a low frequency photo.
The difference is visible for both types however, with low frequency images being more noticeable. Like in this example of an apple from Minecraft:



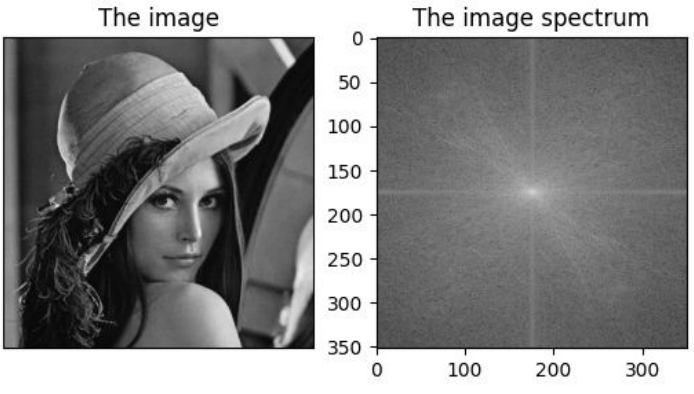
3. **THE THIRD TASK I DID NOT PERFORM UNFORTUNATELY.**

4. **Lena DFT, Inverse DFT and Fourier**

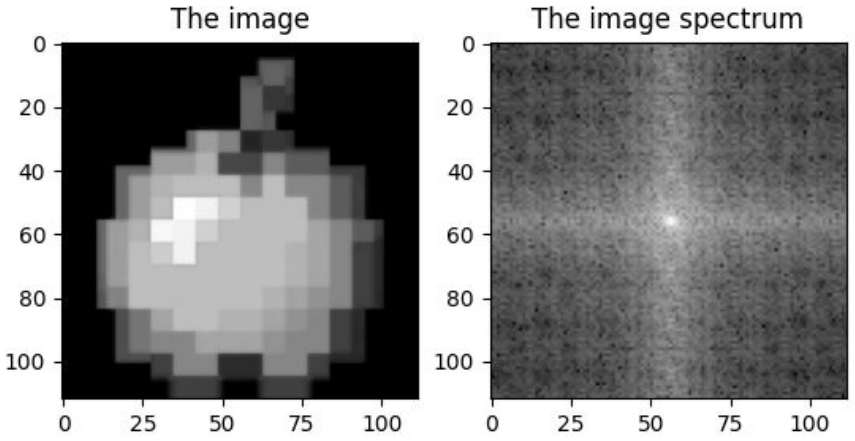
When creating the DFT for Lena's image the following was performed:

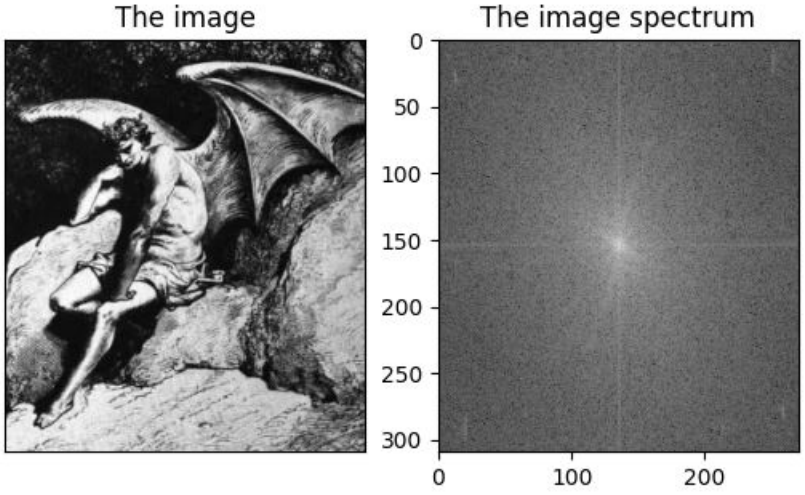
```
#TASK 4

#Lena DFT creation
DFT = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
DFT_shi = np.fft.fftshift(DFT)
mag_sp = 20 * np.log(cv2.magnitude(DFT_shi[:, :, 0], DFT_shi[:, :, 1]))
plt.subplot(121),plt.imshow(img, cmap='gray')
plt.title('The image')
plt.subplot(122),plt.imshow(mag_sp, cmap='gray')
plt.title('The image spectrum')
plt.show()
```

We obtain the image of Lena's DFT image spectrum like below:



for comparison here are the image spectrums for the other images as well:

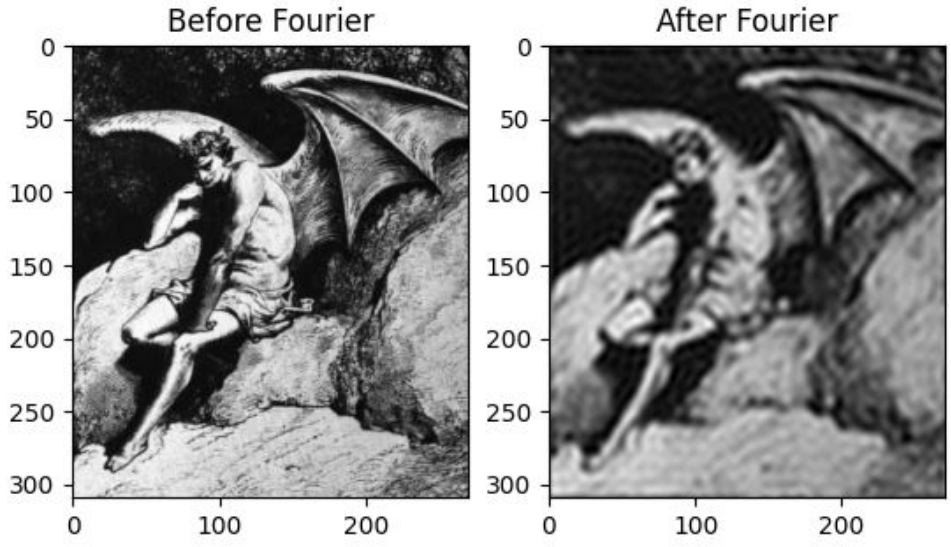




As one can see the image spectrum differs between the images and depending on if it is high or low frequency one can notice different amounts of "specs" or "noise" on the spectrum.

The Fourier Transform are the following:

and just for comparison with other image:



The first thing that is just day and night is the visibility of details after is almost nonexistent. The image becomes very watered down and in fact seems like it's elements are all blending together. It is much less sharp than its predecessor visible on the left side.