HACETTEPE UNIVERSITY

ELECTRICAL AND ELECTRONICS ENGINEERING

ELE492 – SPECIAL TOPICS II

HOMEWORK 2 REPORT

2021-2022 SPRING

EGEMEN CAN AYDUĞAN - 21728036

I have not received or given any aid in this homework. All the work presented below is my own work.

Egemen Can AYDUĞAN

Question 1:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from scipy import ndimage
from skimage.color.colorconv import rgb2gray
```

Libraries

```
Image = io.imread("OldPhoto.jpg")
Image_Gray = cv2.cvtColor(Image,cv2.COLOR_BGR2GRAY)
fig, ax = plt.subplots(1, 2, figsize=(10,10))
ax[0].imshow(Image)
ax[0].set_title('Image')
ax[1].imshow(Image_Gray,cmap="gray")
ax[1].set_title('Image in the Gray Scale')

# Magnitude Spectrum of Image in Gray Scale
FFT_Image_Gray = np.fft.fft2(Image_Gray)
FFT_Shift_Image_Gray = np.fft.fftshift(FFT_Image_Gray)
magnitude_spectrum = np.log(np.abs(FFT_Shift_Image_Gray))
plt.figure(figsize=(10,10))
plt.imshow(magnitude_spectrum,cmap="gray")
```

After ı installed the libraries, I uploaded the photo. To see the periodic noises in the photo, I turned the photo to gray scale and found the places where the noise was.

Fig. 1.1: Original Iamge and Image in Gray Scale

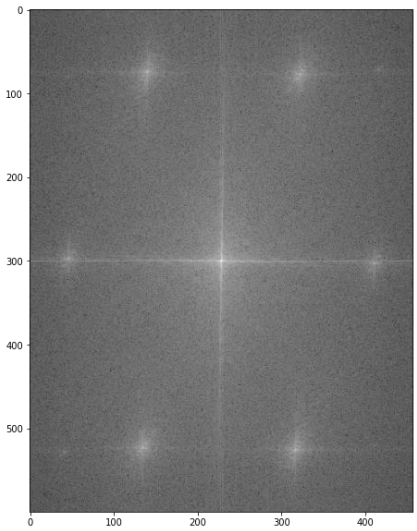

Fig. 1.2: Magnitude Spectrum of the Image in Gray Scale

```
def RGB_Transform(Image, Channel):
  RGB_FFT_Image = np.fft.fftshift(np.fft.fft2(Image[:,:,Channel]))
  RGB_FFT_Image[60:100,300:340] = 1
  RGB_FFT_Image[60:100,120:160] = 1
  RGB_FFT_Image[500:540,300:330] = 1
  RGB_FFT_Image[500:540,120:150] = 1
  RGB_FFT_Image[280:310,20:50] = 1
  RGB_FFT_Image[280:310,400:430] = 1
  RGB_FFT_Image[75:80,45:50] = 1
  RGB_FFT_Image[70:75,415:420] = 1
  RGB_FFT_Image[525:530,40:45] = 1
  RGB_FFT_Image[520:525,410:415] = 1
  RGB = abs(np.fft.ifft2(RGB_FFT_Image))
  return RGB.clip(0, 255)
```

```
# Eliminate the Noise for RGB
RED_Channel = RGB_Transform(Image, 0)
GREEN_Channel = RGB_Transform(Image, 1)
BLUE_Channel = RGB_Transform(Image, 2)

fig, ax = plt.subplots(1, 3, figsize=(15,15))
ax[0].imshow(RED_Channel)
ax[0].set_title('RED_Channel')
ax[1].imshow(GREEN_Channel)
ax[1].set_title('GREEN_Channel')
ax[2].imshow(BLUE_Channel)
ax[2].set_title('BLUE_Channel')
```

After location the noises in the photo, I needed to eliminate them. I did these eliminations not in the gray scale, but in the scale of the photo itself. I have eliminated photo the places to be eliminated in 3 cases. I took the FFT transformations of the state where it was red,green and blue. In magnitude spectrum, I remove the noises by making them ‘1’ where there ia noise, I recorded the noise removed image by IFFT.

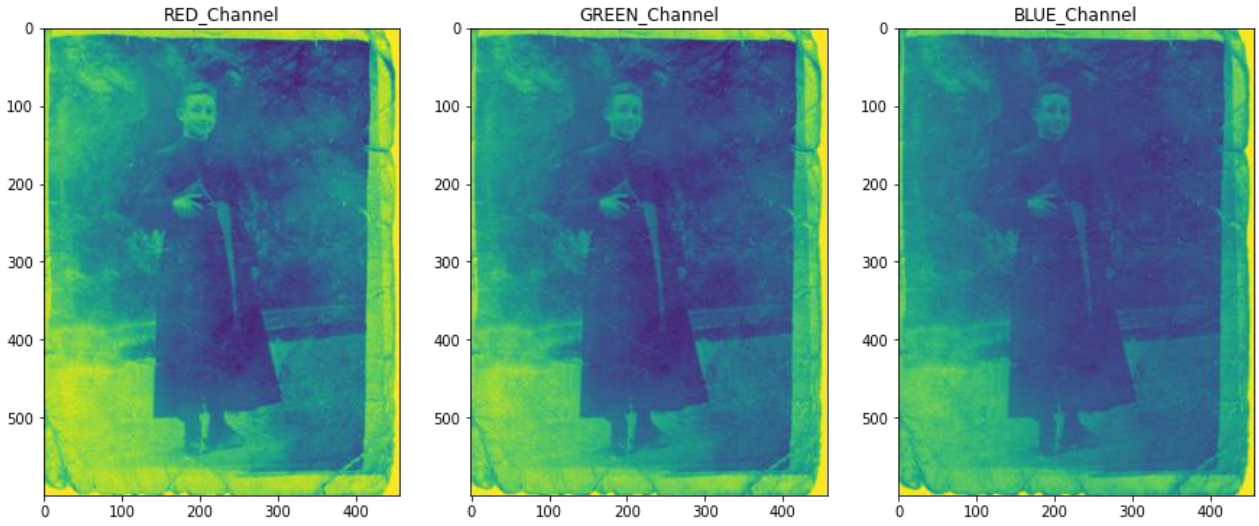

Fig. 1.3: FFT of the Image without Noises in the Red, Green and Blue Channel

```
Restored_Image = np.dstack([RED_Channel.astype(int),GREEN_Channel.astype(int),BLUE_Channel.astype(int)])
Median_Restored_Image = ndimage.median_filter(Restored_Image, size=3)

fig, ax = plt.subplots(1, 3, figsize=(15,15))
ax[0].imshow(Image)
ax[0].set_title('Image')
ax[1].imshow(Restored_Image)
ax[1].set_title('Restored_Image Image')
ax[2].imshow(Median_Restored_Image)
ax[2].set_title('Median_Restored_Image Image')
```

I stacked the Fig. 1.3 images by sequence deepth with “np.dstack”. I got the image I wanted to achieve. I also wanted to apply a median filter on this image and see what would happen. When I applied the median filtre, the periodic noises in the picture were completely removed, but it caused the picture to be blurred.

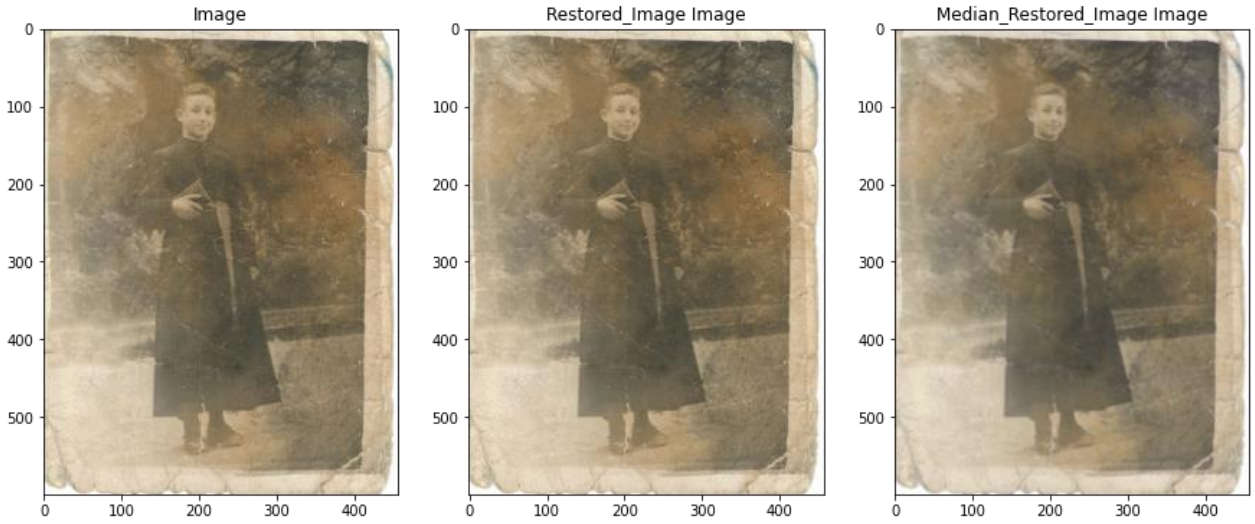

Fig. 1.4: Noisy Image, Restored Image and Image with Median Filter applied

I think the best result is in the middle because noise is removed and the sharpness of the picture is not deteriorated.

Question 2:

```
import cv2
import numpy as np
import matplotlib.image as mpimg
from skimage import io

def ImHist(Im):
  H,W = Im.shape
  M = [0.0]*256
  for i in range(H):
    for j in range(W):
      M[Im[i,j]] += 1
  return np.array(M)/(H*W)
def CumulativeSum(Hist):
  return [sum(Hist[:i+1]) for i in range(len(Hist))]
def HistEq(Im):
  OriginalHist = ImHist(Im)
  CumDistFunc  = np.array(CumulativeSum(OriginalHist))
  TraFunctVal  = np.uint8(255*CumDistFunc)
  SH,SW = Im.shape
  Y = np.zeros_like(Im)
  for i in range(0,SH):
    for j in range(0,SW):
      Y[i,j] = TraFunctVal[Im[i,j]]
  RestoredHist = ImHist(Y)
  return Y,OriginalHist,RestoredHist

# Load Image and Gray Sacle
Image = np.uint8(mpimg.imread("Findit.jpg")*255)
Image_Gray = cv2.cvtColor(Image,cv2.COLOR_BGR2GRAY)

# Histogram
Restored_Image, Original_Hist, Restored_Hist = HistEq(Image_Gray)

plt.figure(figsize=(5,5))
plt.imshow(Restored_Image,cmap = "gray")
plt.title("Restored Image")
plt.show()
plt.figure(figsize=(5,5))
plt.plot(Restored_Hist)
plt.title("Equalized Histogram ")
plt.figure(figsize=(5,5))
plt.imshow(Image_Gray,cmap = "gray")
plt.title("Image in Gray Scale")
plt.show()
plt.figure(figsize=(5,5))
plt.plot(Original_Hist)
plt.title("Histogram")
```

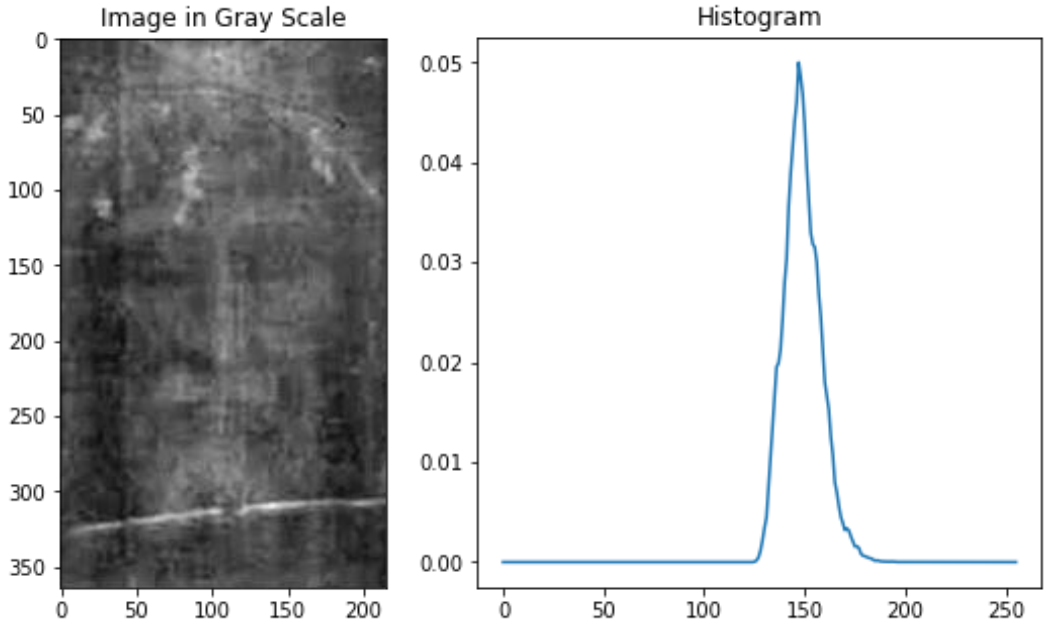

Fig. 2.1: Image in Gray Scale and Its Histogram Graph

I changed the picture to gray scale so that I can work more comfortably. I also made the histogram graph of the picture on this scale.

The color transitions in the picture are too little and the edges are not clear, causing the histogram graph to be concentrated in the narrow space.

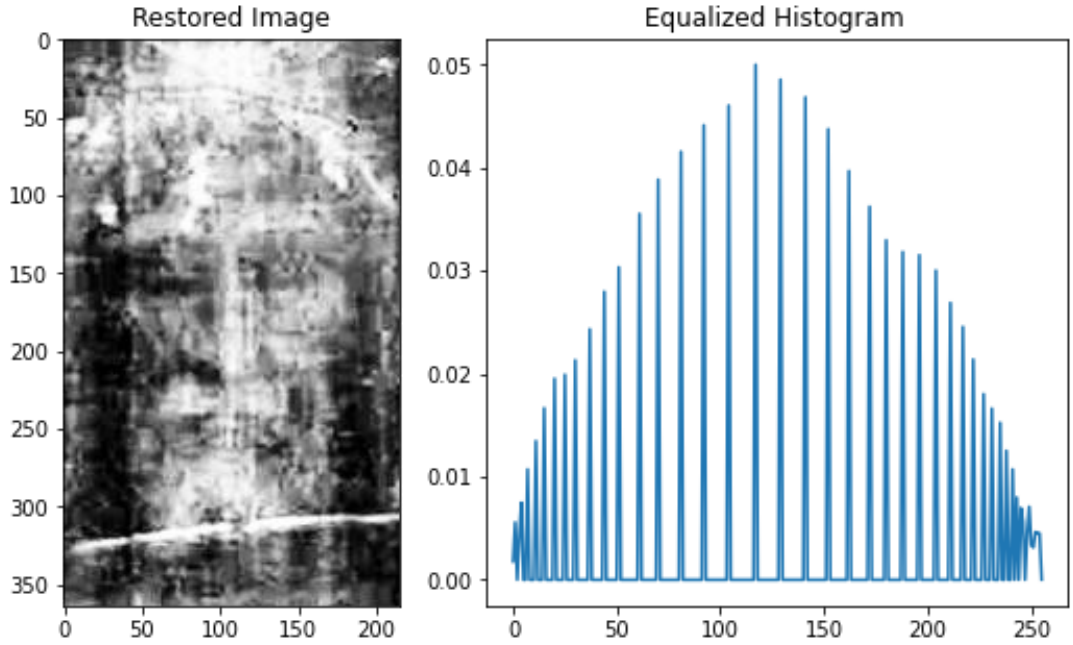

Fig. 2.2: Restored Image and Its Histogram Graph

I applied histogram equalization to see the details in the picture and to distribute the density. I did not want to use OpenCV's function for histogram equalization. In order to see the details, I searched the histogram equalization function from the web.