HACETTEPE UNIVERSITY
DEPARTMENT OF
GEOMATICS
ENGINEERING

GMT234
DIGITAL IMAGING & INTERPRETATION 2021-2022

Prof. Dr. Ali Özgün OK

HOMEWORK I REPORT

Abdulsamet TOPTAŞ

Deadline	10/04/2022
Delivery date	10/04/2022

QUESTION -1-

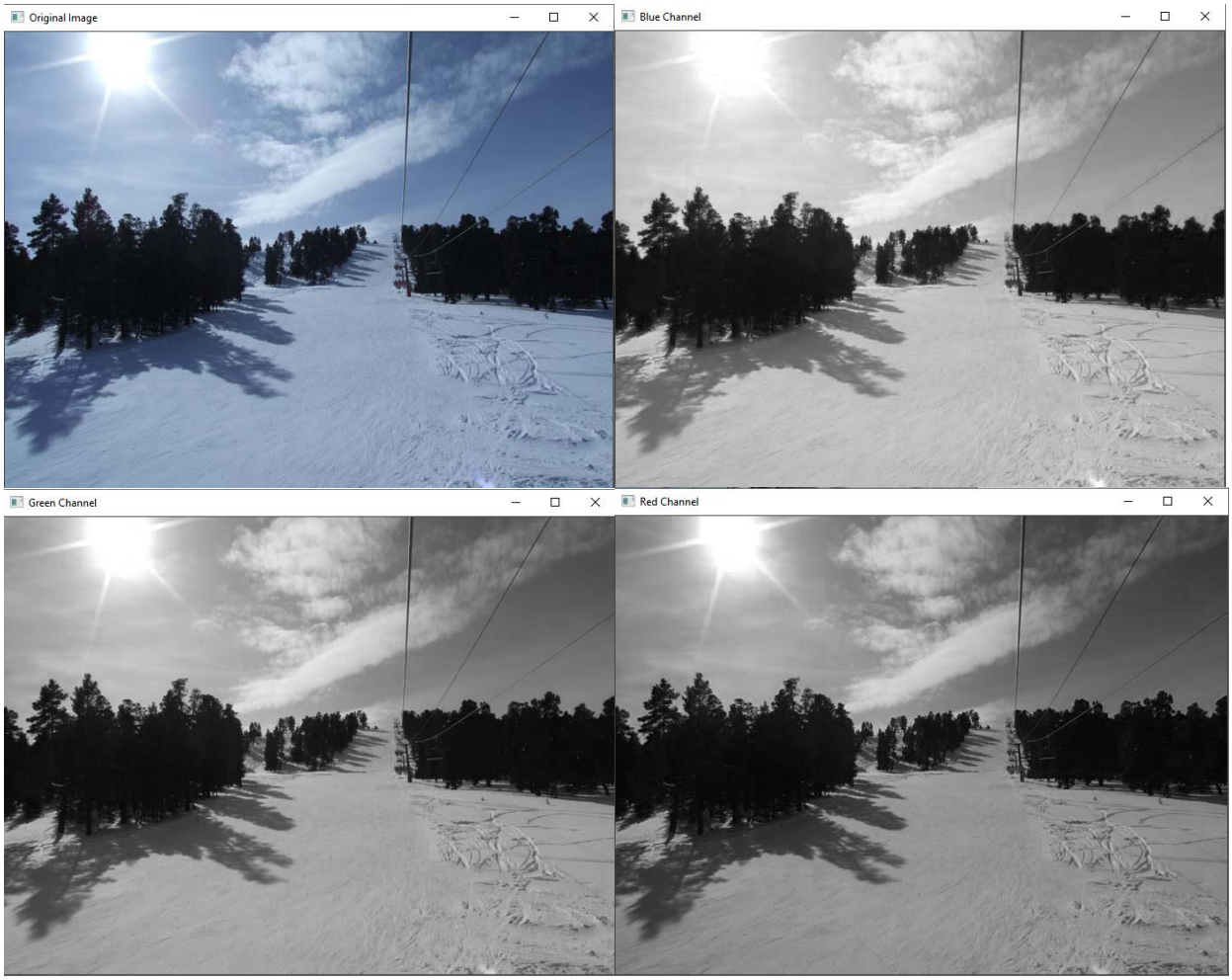
1. Sarıkamış is a fast rising resort located 2.5 kilometers from the town center, 50 kilometers from Kars International Airport, and surrounded by pine trees. Because of its elevation and distance from the shore, Sarıkamış is considered a snow-guaranteed destination. Aside from being rather short in comparison to Alpine standards, the pistes at Sarıkamış are among the longest in Turkey. In this respect, you are given an image acquired from the Sarıkamış ski resort (Sarikamis.jpg).

A) Split the given RGB image into 3 different grayscale images, and save the outputs in “tif” format. (5 points)

```python
# QUESTION - 1

# A) Split the given RGB image into 3 different grayscale images, and save the outputs in “tif” format. (5 points)

import cv2
from matplotlib import pyplot as plt

# Reading original image
img=cv2.imread("Sarikamis.jpg")

# Splitting as RGB
Blue=img[:,:,0]
Green=img[:,:,1]
Red=img[:,:,2]

# Showing Channels as RGB and original img
cv2.imshow('Original Image',img)
cv2.imshow('Blue Channel',Blue)
cv2.imshow('Green Channel',Green)
cv2.imshow('Red Channel',Red)
# Saving Channels as .tif
cv2.imwrite("New_Blue_Image.tif", Blue)
cv2.imwrite("New_Red_Image.tif", Green)
cv2.imwrite("New_Green_Image.tif", Red)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

I scanned the image given as (Sarikamis.jpg) with the help of the "cv2" library. Later, I divided the images that I will separate as RGB into three separate bands and numbered them in sequence (0,1,2). Lastly, I brought both the original and the three images that I split to the screen with the "imshow" command and saved them to the file using the "imwrite" command. I also specified that the images should stay open until the user closes them by entering 0 in the “weitKey” value.

Output A;

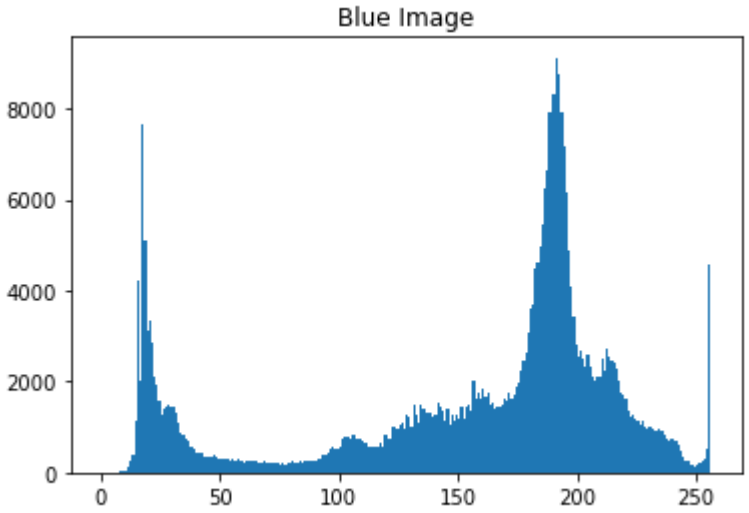


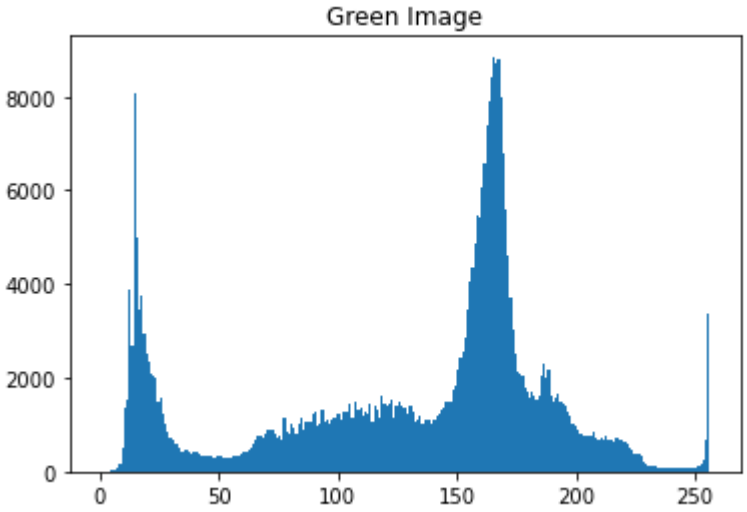
B) Create histogram of each grayscale image created in (a). (5 points)

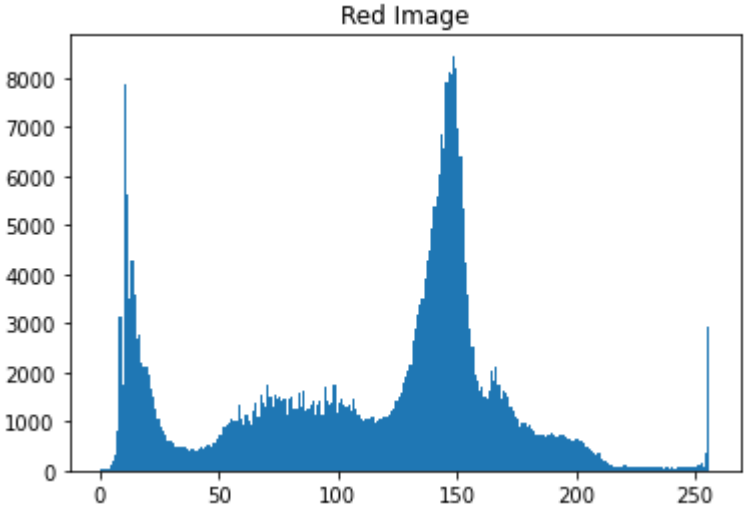
```
# B) Create histogram of each grayscale image created in (a). (5 points)

# Reading images
Blue_Image = cv2.imread('New_Blue_Image.tif',0)
Green_Image = cv2.imread('New_Red_Image.tif',0)
Red_Image = cv2.imread('New_Green_Image.tif',0)

# Finding histograms of all grayscale images in the 0-255 band
# Blue image histogram
plt.hist(Blue_Image.ravel(),256,[0,256])
plt.title('Blue Image')
plt.show()
# Green image histogram
plt.hist(Green_Image.ravel(),256,[0,256])
plt.title('Green Image')
plt.show()
# Red image histogram
plt.hist(Red_Image.ravel(),256,[0,256])
plt.title('Red Image')
plt.show()
```

I created three different histograms in RGB using the "tif" images I recorded in the first part. Then, I brought it to the screen with the help of the "plt.show()" command of the "matplotlib" library in the 0-255 band. Thus, we can easily interpret the pixel values on the histogram.

Output B;


Blue Image
8000
6000
4000
2000
0
0
50
100
150
200
250


Green Image
8000
6000
4000
2000
0
0
50
100
150
200
250


Red Image
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

C) Create a new RGB image by adding 60 to all grayscale images created in (a). (5 points) (must be done without a python library)

```
# C) Create a new RGB image by adding 60 to all grayscale images created in (a). (5 points)
# (must be done without a python library)

# Creating a new RGB by adding 60
(h, w) = Blue.shape

for i in range(h):
    for j in range(w):
        Blue[i,j]=Blue[i,j]+60

for i in range(h):
    for j in range(w):
        Red[i,j]=Red[i,j]+60

for i in range(h):
    for j in range(w):
        Green[i,j]=Green[i,j]

# Showing Channels as RGB
cv2.imshow('New_Green_Channel',Green)
cv2.imshow('New_Blue_Channel',Blue)
cv2.imshow('New_Red_Channel',Red)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

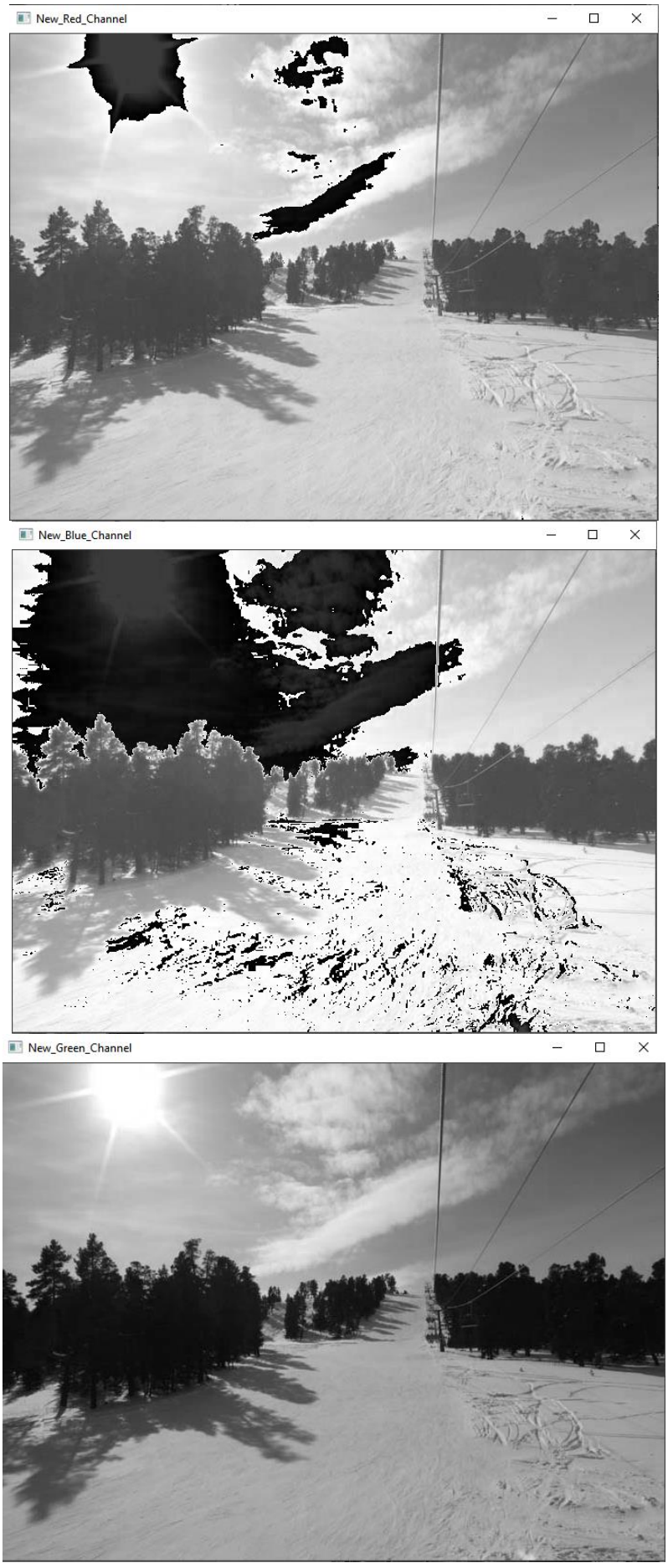
In this section, I added +60 to each of the RGB images. While doing this, I created a for loop and reached all rows and columns with the range() command and added this value to each value in the matrix as [i,j] for both rows and columns and got my new images.

İmportant Note : I did not add +60 to the Green band because the pixel values in the image in the D part are drawn from c. I worked hard for this, I created a new 0 matrix in the C part and reprinted it under a separate variable, but when I did this, the output in the c part came up blank. **Because of all this, I didn't want to add +60 to the Green band and spoil the D part.**

Below is the image of the code I tried;

```
# C) Create a new RGB image by adding 60 to all grayscale images created in (a). (5 points)
# (must be done without a python library)
import cv2
import numpy as np


# Creating a new RGB by adding 60
(h, w) = Blue.shape

empty_blue=np.zeros((h,w))
empty_green=np.zeros((h,w))
empty_red=np.zeros((h,w))

for i in range(h):
    for j in range(w):
        empty_blue[i,j]=int(Blue[i,j])+60
        if empty_blue[i,j]>255:
            empty_blue[i,j]=255

for i in range(h):
    for j in range(w):
        empty_green[i,j]=int(Green[i,j])+60
        if empty_green[i,j]>255:
            empty_green[i,j]=255
            continue
for i in range(h):
    for j in range(w):
        empty_red[i,j]=int(Red[i,j])+60
        if empty_red[i,j]>255:
            empty_red[i,j]=255

# Showing Channels as RGB
cv2.imshow('New_Blue_Channel',empty_blue)
cv2.imshow('New_Green_Channel',empty_green)
cv2.imshow('New_Red_Channel',empty_red)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output C;



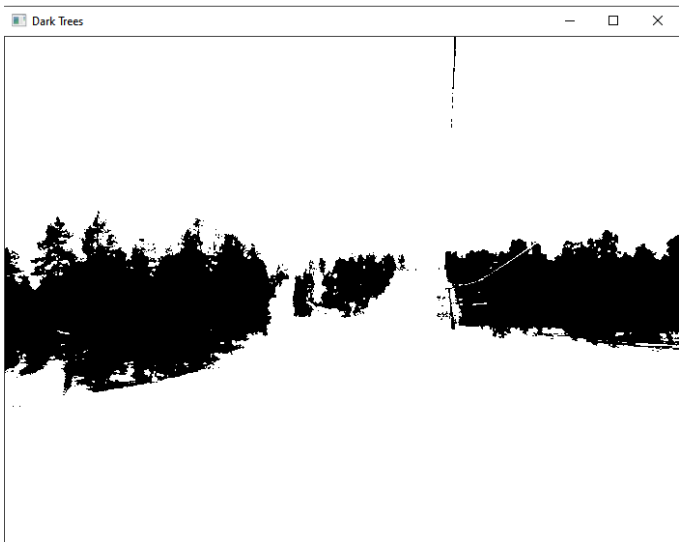
D) With the help of one the grayscale images created in (a), design and develop an algorithm (using a python code) that automatically counts the total number of pixels covered by the pine trees. (25 points) (must be done without a python library)

```python
# D) With the help of one the grayscale images created in (a), design and develop an algorithm (using a python code) that
# automatically counts total number of pixels covered by the pine trees.(25 points)(must be done without a python library)
import cv2
import numpy as np

print(Green.shape)
(h,w)=Green.shape

# A threshold range has been set
for i in range(h):
    for j in range(w):
        if Green[i,j]>50:
            Green[i,j]=255
        else:
            Green[i,j]=0
# An algorithm that counts total pixels
num = 0
for i in range(h):
    for j in range(w):
        if Green[i,j] == 0:
            num = (num +1)
            continue

print(f'the total number of pixels covered by the pine trees: {round(num)}')
cv2.imshow('Dark Trees',Green)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this section, in addition to the "cv2" module, I imported the "numpy" module for mathematical calculations. I printed the green image in section a) and returned a tuple containing the number of items in rows and columns corresponding to each index. I set the threshold by creating a for loop. The purpose of this was to whiten all but the trees in the image, convert the trees to a dark image and start with a dark color when counting the pixels. In this case, I set values greater than 50 equal to 255. So I made a bright image. I set pixel values less than 50 equal to 0. So I made a dark image. Next, I created an algorithm to count the sum of tree pixels by creating another for loop. By assigning a number variable, I designed an algorithm to scan the entire row column, detecting the sum of the pixels of the black places, i.e. the places equal to 0, and I have successfully completed this section.

Output D;

```
(523, 698)
the total number of pixels covered by the pine trees: 59881
```



QUESTION -2-

2. You are given a screenshot image of a highway scene (image1.png). Besides, an image without any foreground objects is given (image0.png). Design and develop an algorithm (using a python code) that automatically counts the total number of cars in the given highway screenshot image. *(30 points)* (must be done without a python library)

(a) Image1.png

(b) Image0.png

Code ;

```python
# QUESTION - 2
# You are given a screenshot image of a highway scene (image1.png). Besides, an image without any foreground objects
# is given (image0.png). Design and develop an algorithm (using a python code) that automatically counts the total number
# of cars in the given highway screenshot image. (30 points) (must be done without a python library)

import cv2
import numpy as np

# Reading images
img1=cv2.imread("image1.png")
img0=cv2.imread("image0.png")

# Assigned to the same bands
image1=img1[:,:,0]
image0=img0[:,:,0]

# Print the car image and create a zero matrix
print(image1.shape)
(h,w)=image1.shape
empty_img=np.zeros((h,w))

# Get the difference of pixel values and determine the change
for i in range(h):
    for j in range(w):
        empty_img[i,j]=int(image0[i,j])-int(image1[i,j])
        if empty_img[i,j]<150 and empty_img[i,j]>40:
            empty_img[i,j]=255

        else:
            empty_img[i,j]=0

# Finding the number of cars
number = 0
for i in range(h):
    for j in range(w):
        if empty_img[i,j] == 255:
            number = (number +1)
            continue
print(f'the total number of cars: {round(number/400)}')

# Showing Channels as Threshold and original img
cv2.imshow('threshold_image1',empty_img)
cv2.imshow("image1",image1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this section, I first had two different images read with the "imread" command. Next, I transferred both of our images onto the same band as seen on lines “14-15” so that I could easily create an algorithm for the difference between the two images. I created a zero matrix using the "Numpy" module. This will help me change the cars to bright color with the differences in the two images when applying my threshold values. Next, I created a for loop and subtracted the two image values from each other using the "Image Subtraction" topic to determine their degree of variation. The points where the degrees of change are high are the places where the car is and where it is not. Where the car is is where the value 255 is. I

determined my threshold value range with the “if” condition I created and when I found the cars in my remaining values in the range, I assigned these values to 255. In other words, since the values I assigned are the values of the white part, the cars will appear as white in the image. While finding the Threshold range, I created a histogram and even though I got help from him, I was able to get our cars to the white part by trial and error method. Finally, I created a for loop and created an "if" condition, counting 255 values in the display, and I got our total number of cars.

Output QUESTION 2;

```
(240, 320)
the total number of cars: 5
```

QUESTION -3-

3. You are given a Portrait of a Young Woman which is a small oil-on-oak panel painting completed in 1470 (Portrait_of_a_Young_Woman.jpg). It marks a major stylistic advance in contemporary portraiture; the girl is set in an airy, three-dimensional, realistic setting, and stares out at the viewer with a complicated expression that is reserved, yet intelligent and alert.

However, because this picture belongs to a 15th century painting, we clearly see the deformation and cracks within the picture.

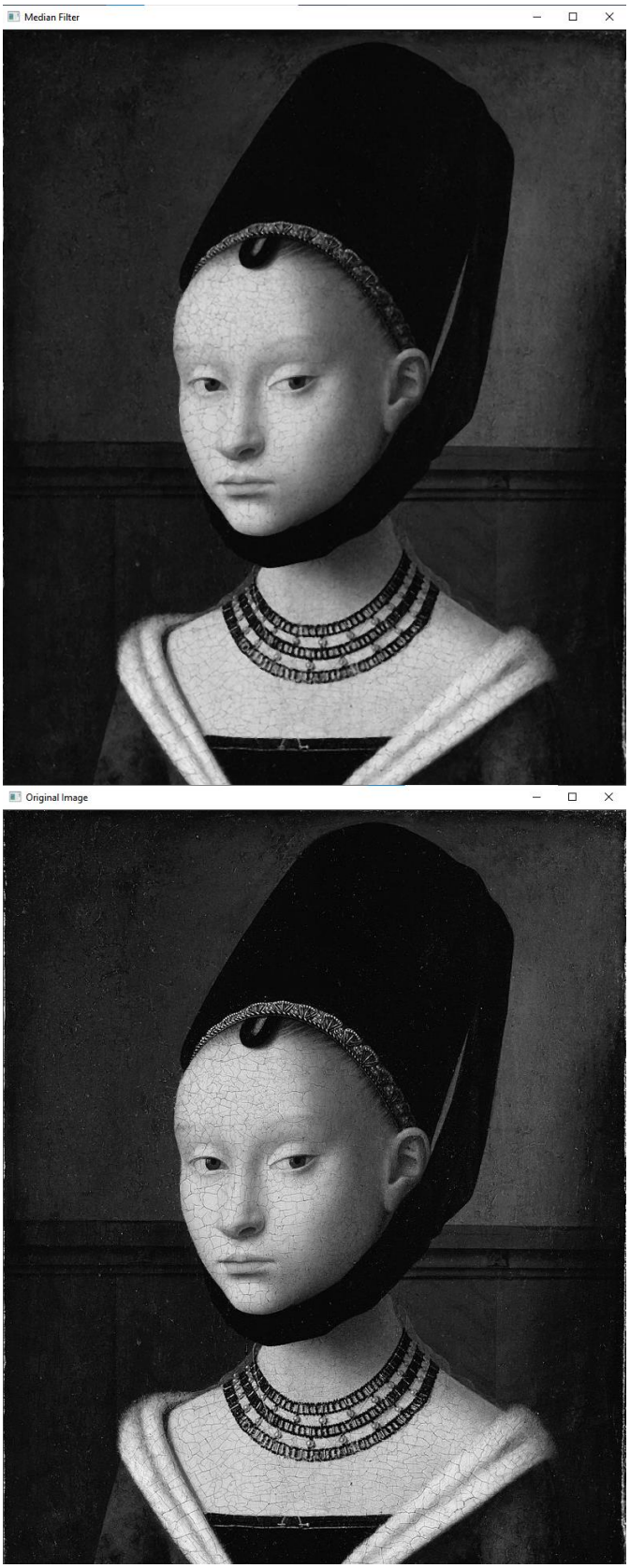
Find at least two different ways to digitally improve the quality of this picture with a Python code (30 points). (must be done without a python library)

First Way – Median Filter ;

```python
# QUESTION - 3    ((FIRST WAY)) - Median Filter
# Find at least two different ways to digitally improve the quality of this picture with a Python code
# (30 points). (must be done without a python library)

import cv2
import numpy as np

# Reading image
img = cv2.imread('Portrait_of_a_Young_Woman.jpg', 0)

# Set the number of rows and columns
h, w = img.shape

# hover image pixels by creating matrix and find median value
new_median_image = np.zeros([h, w])
for i in range(1,h-1):
    for j in range(1,w-1):
        pix = [img[i-1, j-1],
               img[i-1, j],
               img[i-1, j + 1],
               img[i, j-1],
               img[i, j],
               img[i, j + 1],
               img[i + 1, j-1],
               img[i + 1, j],
               img[i + 1, j + 1]]

        pix = sorted(pix)
        new_median_image[i, j]= pix[4]
new_median_image= new_median_image.astype(np.uint8)

# Showing Channels as Median Filter and original img
cv2.imshow('Original Image',img)
cv2.imshow('Median Filter',new_median_image)
cv2.imwrite('new_median_filtered.jpg', new_median_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this section, I first wanted to apply the median filter. This filter is one of the best ways to soften the picture. The purpose of this filter is to sort the pixel values in the matrix and assign the median value to the middle of the new matrix we will create. First, I called our image with the "imread" command, then I got the number of rows and columns of the image on the "12" line. Then I created the zero matrix with the help of the "Numpy" module. The purpose of this is to easily write our value in the middle of our new matrix while assigning the median value we will reach after the loop. Next, I created a For loop and wrote an algorithm that can reach the median value of our image. While creating this matrix, I did various researches and finally created my matrix. Using the “sorted()” command, I sorted the values from smallest to largest and scanned and printed the median value with the help of 3x3 matrices.

Output QUESTION 3 – Median Filter ;



Second Way – Mean Filter ;

```python
# QUESTION - 3   ((SECOND WAY)) - Mean Filter
# Find at least two different ways to digitally improve the quality of this picture with a Python code
# (30 points). (must be done without a python library)

import cv2
import numpy as np

# Reading image
img = cv2.imread('Portrait_of_a_Young_Woman.jpg', 0)

# Set the number of rows and columns
h, w = img.shape

# Creating a unique 3x3 matrix
uniq = np.ones([3, 3], dtype = int)
# Multiply the unit matrix by 1/9
uniq = uniq / 9

# Convolve the 3X3 mask over the image
mean_img = np.zeros([h, w])

for i in range(1, h-1):
    for j in range(1, w-1):
        pix = img[i-1, j-1]*uniq[0, 0]+img[i-1, j]*uniq[0, 1]+img[i-1, j + 1]*uniq[0, 2]+img[i, j-1]*uniq[1, 0]
        + img[i, j]*uniq[1, 1]+img[i, j + 1]*uniq[1, 2]+img[i + 1, j-1]*uniq[2, 0]+img[i + 1, j]*uniq[2, 1]
        +img[i + 1, j + 1]*uniq[2, 2]

        mean_img[i, j]= pix

mean_img = mean_img.astype(np.uint8)
cv2.imshow('original image',img)
cv2.imshow('mean filter',mean_img)
cv2.imwrite('blurred.jpg', mean_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In this section, I wanted to apply the Mean filter. This filter is one of the best ways to remove noise from the image. My purpose here is to move each value through 3x3 matrices after creating the unit matrix and multiply by 1/9 after adding each value. This gives us the average. I created a 3x3 unit matrix using the "Numpy" module. Then I multiplied this unit matrix by 1/9 because my matrix was 3x3. I created a "For" loop and moved my unit matrix, which I multiplied by 1/9, through the pixel values of our image and summed it up to find the average value and applied the mean filter.

Output QUESTION 3 – Mean Filter ;


original image