HACETTEPE UNIVERSITY

GMT234 DIGITAL IMAGING AND INTERPRETATION

2021-2022 SPRING

HOMEWORK REPORT

Mert Çetintürk

21967387

Before we start the code, we do the importing part, which is our preparation stage.

Preparation Phase

```
]: import numpy as np
   import cv2
   import matplotlib.pyplot as plt
   from PIL import Image
```

After finishing this step, let's start with part A of question 1.

In this section, we have done the process of reading our image and displaying it on the screen.

```
sarikamis = cv2.imread('Sarikamis.jpg')

cv2.imshow('Sarikamis',sarikamis)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

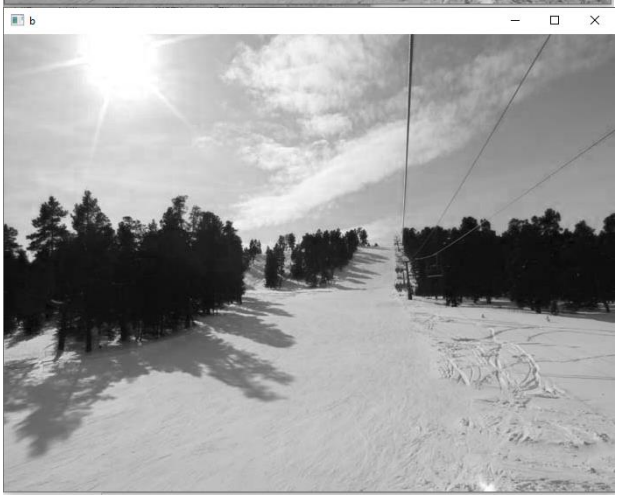
After that we split the image into bands,

```
b, g, r = cv2.split(sarikamis)
cv2.imshow('b',b)
cv2.imshow('g',g)
cv2.imshow('r',r)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This is our second way to split image into bands,

```
img_B = sarikamis[:,:,0]
img_G = sarikamis[:,:,1]
img_R = sarikamis[:,:,2]

cv2.imshow('img_B',img_B)
cv2.imshow('img_G',img_G)
cv2.imshow('img_R',img_R)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

These are RGB images,



After doing these, we were asked to convert the RGB images we obtained from jpg to tiff and save them. We can do this in two different ways. One with OpenCV,

```
cv2.imwrite('Sarikamis_Red.tiff',img_R)
cv2.imwrite('Sarikamis_Green.tiff',img_G)
cv2.imwrite('Sarikamis_Blue.tiff',img_B)
```

```
Out[5]: True
```

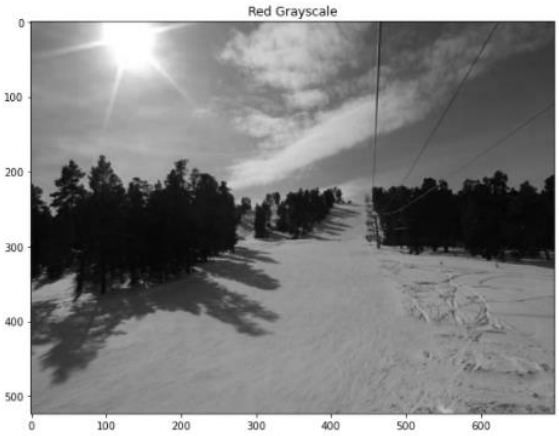
the other with Pillow,

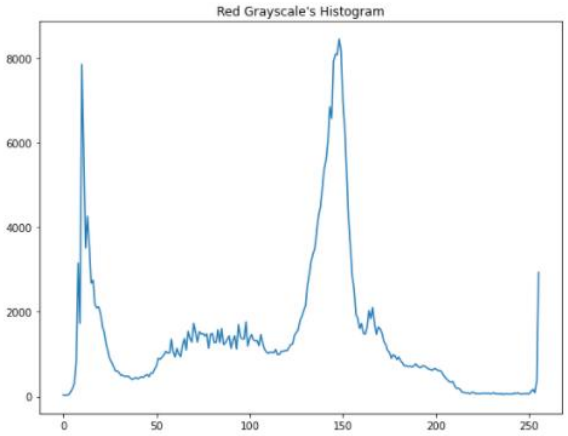
```
im = Image.open('Sarikamis.jpg')
im.save("Sarikamis.tiff",'TIFF')

image = cv2.imread('Sarikamis.tiff')

cv2.imshow('Sarikamis',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Now let's come to part B of question 1. Here we are asked to show the histograms of the RGB images we have obtained. We did this easily with the help of matlotlib and projected it to the screen. Below you just see the histogram code of an RGB image,

```
histogram_red = cv2.calcHist([sarikamis_red],[0],None,[256],[0,256])

plt.figure(figsize = (20,7))
plt.subplot(121)
plt.title("Red Grayscale")
plt.imshow(sarikamis_red)
plt.subplot(122)
plt.title("Red Grayscale's Histogram")
plt.plot(histogram_red)
```

Now you can see the output of this code below,

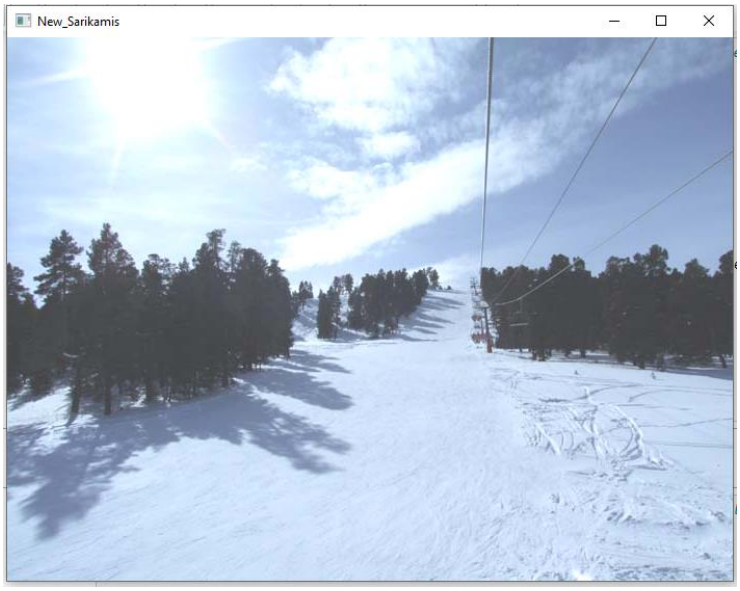




In the C part of the question, we were asked to add 60 values to each RGB layer and then combine these images. Again, we did this easily with the help of OpenCV.

```
red_tint = 60
green_tint = 60
blue_tint = 60
img = Image.open('Sarikamis.tiff')
red, green, blue = img.split()

# loop to add values to images
for y in range(img.height):
    for x in range(img.width):
        value = img.getpixel((x, y))
        new_color = (value[0] + red_tint, value[1] + green_tint, value[2] + blue_tint)
        img.putpixel((x, y), new_color)
img.save('New_Sarikamis.tiff')

new_sarikamis = cv2.imread('New_Sarikamis.tiff')

cv2.imshow('New_Sarikamis',new_sarikamis)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

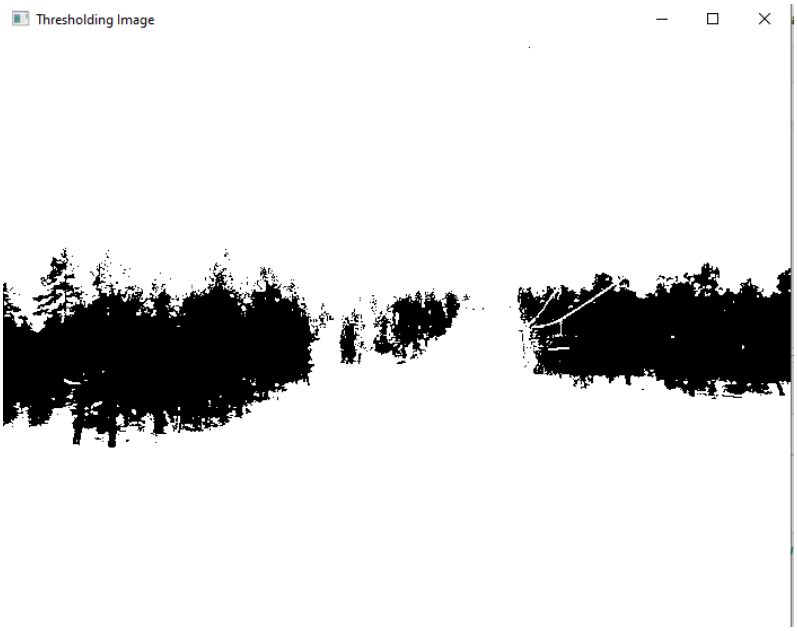
This is the output of the code,



In part D, we were asked to find the number of pixels in the pine trees in the image. Here, too, we made the image black and white with the threshold code we wrote and then counted the pixels in the trees.

```
img = cv2.imread("Sarikamis_Red.tiff",0)

height = img.shape[0]
width = img.shape[1]

img_thres= np.zeros((height,width))
n_pix = 0
T = 26 # threshold value

# threshold code
for y in range(height):
    for x in range(width):
        pixel = img[y,x]
        if pixel < T:
            n_pix = 0
        else:
            n_pix = 255

        img_thres[y,x] = n_pix

cv2.imshow("Thresholding Image", img_thres)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

And this code's output,



This is the number of pixels of trees,

```
number_of_Tree_pix = np.sum(img_thres == 0)
print('Number of Tree Pixels: ',number_of_Tree_pix)
```

```
Number of Tree Pixels:  51428
```

Now we move on to question 2.

Here we are given two images. One is a blank highway image and the other is the same highway image with cars. He asks us how many cars are here. First of all, we extract our images from each other with the help of OpenCV. This way we will only get the pixels of the cars.

```
image_highway1 = cv2.imread('image0.png',0)
image_highway2 = cv2.imread('image1.png',0)

image_highway3 = cv2.absdiff(image_highway2, image_highway1)

cv2.imshow('Cars',image_highway3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

And it's output,

Then we apply threshold to interpret the images more accurately.

```
ret, thresh = cv2.threshold(image_highway3, 50, 255, cv2.THRESH_BINARY)
cv2.imshow('Thresholded', thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

And it’s output,

Then we divide the resulting pixel values by the average vehicle pixel number and find how many cars are in the image.

```
number_of_cars = int(np.sum(thresh == 255)/400)
print("Total Cars:", number_of_cars)
```

```
Total Cars: 4
```

We move on to question 3. In this question, we were asked to correct the distortions in the image. Smoothing filters should be used for this task. What we use here are Averaging Filter and Median Filter. Both filters are used to correct image artifacts and soften the image.

Let's look at the Averaging Filter first

```
img = cv2.imread('Portrait_of_a_Young_Woman.jpg', 0)

# Obtain number of rows and columns
# of the image
m, n = img.shape

# Develop Averaging filter(3, 3) mask
mask = np.ones([3, 3], dtype = int)
mask = mask / 9

# Convolve the 3X3 mask over the image
img_new = np.zeros([m, n])

for i in range(1, m-1):
    for j in range(1, n-1):
        temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j + 1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mask[

        img_new[i, j]= temp

img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.tiff', img_new)

cv2.imshow('Blurred',img_new)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

First we assign the width and height of our original image to two different variables.

```
m, n = img.shape
```

We will use these variables later when we create the zero matrix.

```
img_new = np.zeros([m, n])
```

We create the filter that we will apply. Here we create a 3x3 unit matrix and divide this matrix by 9. Because that's the rule. If it was 4x4, we would have to divide by 16 this time.

```
mask = np.ones([3, 3], dtype = int)
mask = mask / 9
```

Then we create two loops and apply this filter to all pixels in our image and assign it to a variable.

```
for i in range(1, m-1):
    for j in range(1, n-1):
        temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j + 1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mas

        img_new[i, j]= temp

img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.tiff', img_new)
```

We process the astype command so that there is no incompatibility problem in this variable we created, and then we create our new, filtered image.

```
img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.tiff', img_new)

cv2.imshow('Blurred',img_new)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This is the output,

Now let's look at the median filter.

Here again we first read our image and then we assign the width and height to a variable. These variables will then be used to construct the zero matrix.

```
img_noisy1 = cv2.imread('Portrait_of_a_Young_Woman.jpg', 0)

# Obtain the number of rows and columns
# of the image
m, n = img_noisy1.shape

# Traverse the image. For every 3X3 area,
# find the median of the pixels and
# replace the center pixel by the median
img_new1 = np.zeros([m, n])
```

Then we try to reach all the pixels in the image by creating two loops. Then we sort these 3x3 matrices among themselves.

```
for i in range(1, m-1):
    for j in range(1, n-1):
        temp = [img_noisy1[i-1, j-1],
               img_noisy1[i-1, j],
               img_noisy1[i-1, j + 1],
               img_noisy1[i, j-1],
               img_noisy1[i, j],
               img_noisy1[i, j + 1],
               img_noisy1[i + 1, j-1],
               img_noisy1[i + 1, j],
               img_noisy1[i + 1, j + 1]]

        temp = sorted(temp)
        img_new1[i, j]= temp[4]
```

We process the astype command so that there is no incompatibility problem in this variable we created, and then we create our new, filtered image.

```
img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_median_filtered.tiff', img_new1)

cv2.imshow('Median Filtered',img_new1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

This is the output,