462 Homework 3

This is the third homework. I learned how to draw a histogram with this homework. I also learned to change the contrast value of an image with the values I determined. The smaller the values I enter, the fewer pixels are calculated. As a result, there is not much change in the picture. But if I increase the values I enter, the number of pixels going to the minimum and maximum also increases. This makes the picture extremely sharp. The complexity of this process depends on the number of pixels of the image we will enter. The number of different pixels in the histogram is also important. so complexity is o(n+k)

Code Work

At the beginning, we take the image input. We use functions such as image.open, which I mentioned in the previous assignment. We give the Slow and Shigh values. These values will be used when calculating how many pixels we need to take. Then we save the current pixels to the array. We find the total pixels. We find the total pixels. Here we multiply the width of the image and the height of the image We create the histogram according to these values. Using this histogram, we find aLow and aHigh values. We move through the histogram as many pixels in the S area and get the value there. To create the new image, I created an array with values of 0. When creating the histogram of the new shape, we use the values that need to be minimized and maximized. Then I display the original picture, the gray version, the original histogram, the current histogram and the current picture on the screen.

462-homework-3-2

March 27, 2024

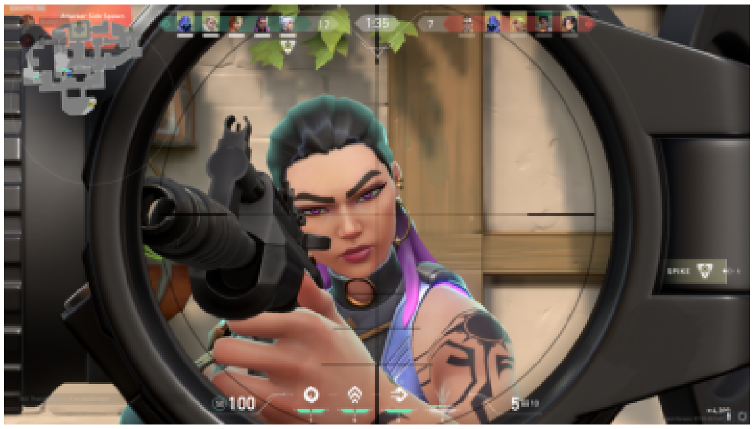
[2]:
```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def findLow(histogram,totalPixel,Slow):
    somePixel=totalPixel*Slow / 100    # What percentage of total pixels
    summation = 0          # keep sum
    value = 0          # for keep a low

    while value < len(histogram):                    #
        summation = summation+histogram[value]      #
                                                    #
        if summation >= somePixel:                  #   Proceed by adding the␣
 ↪intensity values in the histogram until you reach the value at some pixel.
                                                    #
            return value                            #
        value+=1                                    #

    return 0


def findHigh(histogram,totalPixel,Shigh):
    somePixel=totalPixel*Shigh / 100   # What percentage of total pixels
    summation=0                       # keep sum
    value=len(histogram) - 1          # for keep a high

    while value>=0:                                     #
        summation= summation+histogram[value]          #
                                                       #
        if summation>=somePixel:                       #  Proceed by adding the␣
 ↪intensity values in the histogram until you reach the value at some pixel.
                                                       #
            return 255-value                           #
        value-=1                                       #

    return 0
```

```
def fmac(a,alow,ahigh,amin,amax):
    if a<=alow:                              # reduce places up to low to min
        return amin

    elif alow<a<ahigh:
        return amin+(a-alow)*(amax-amin)/(ahigh-alow)     # update the values␣
 ↪in between

    else:
        return amax                         # increase places up to high to␣
 ↪max.

def histogram_equalization(Input,Slow,Shigh): #
    if Input.mode != 'L':                    #   to make sure the image is␣
 ↪black and white
        Input=Input.convert('L')             #



    pixelArray=np.array(Input)      # keep the intensity of image pixels in an␣
 ↪array


    totalPixel=Input.size[0]*Input.size[1]    # find total pixel
    histogram, ranges=np.histogram(pixelArray, bins=range(257)) # give array␣
 ↪and range create histogram(Each value in the range gives the number of␣
 ↪pixels at that intensity.)

    a_low=findLow(histogram,totalPixel,Slow)  # call findLow function for␣
 ↪finding alow
    a_high=findHigh(histogram,totalPixel,Shigh) # call findHigh function for␣
 ↪finding ahigh

    equalized_pixels=np.zeros_like(pixelArray)      # Give the array 0 values

    for i in range(256):
        equalized_pixels[pixelArray==i]=fmac(i,a_low,a_high,0,255)   #  check␣
 ↪values in histogram and send to update function

    equalized_image=Image.fromarray(equalized_pixels.astype(np.uint8)) #␣
 ↪Converts pixel values to 8-bit integers.

    original_histogram, r1=np.histogram(pixelArray, bins=range(257))    #␣
 ↪create original image histogram
    equalized_histogram, r2=np.histogram(equalized_pixels, bins=range(257)) #␣
 ↪create update image histogram
```

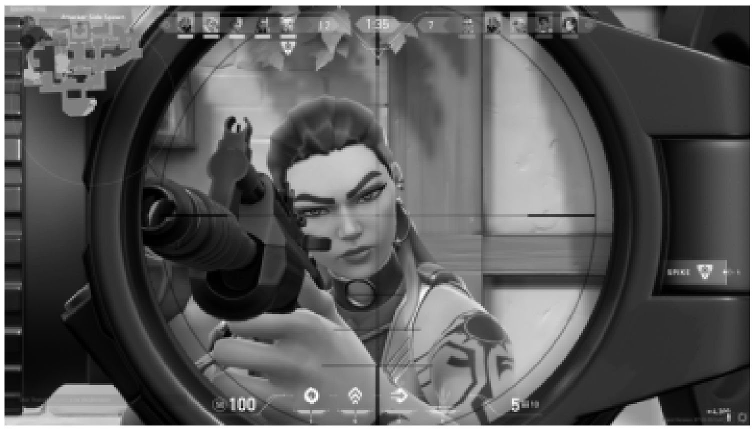
```python
    return equalized_image,original_histogram,equalized_histogram

def Show(Output,original_histogram,equalized_histogram):

    plt.figure(figsize=(10, 5))



    plt.subplot(1, 3, 1)
    plt.plot(original_histogram* np.prod(Input.size), color='blue', alpha=0.7)
    plt.title('Original Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')

    plt.subplot(1, 3, 2)
    plt.plot(equalized_histogram* np.prod(Input.size), color='red', alpha=0.7)
    plt.title(f'Equalized Histogram (slow={Slow}, shigh={Shigh})')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')

    plt.tight_layout()
    plt.show()





    plt.figure(figsize=(5, 5))
    plt.imshow(Output, cmap='gray')
    plt.title(f'Output Image (slow={Slow}, shigh={Shigh})')
    plt.axis('off')
    plt.show()


Input=Image.open("C:/Users/Nida/Pictures/Screenshots/9.png") #input image

Slow=10     # enter Slow
Shigh=90   # enter Shigh

plt.figure(figsize=(5, 5))
plt.imshow(Input, cmap='gray')
plt.title('Input Image')
plt.axis('off')
plt.show()

bw_image = Input.convert("L")
```

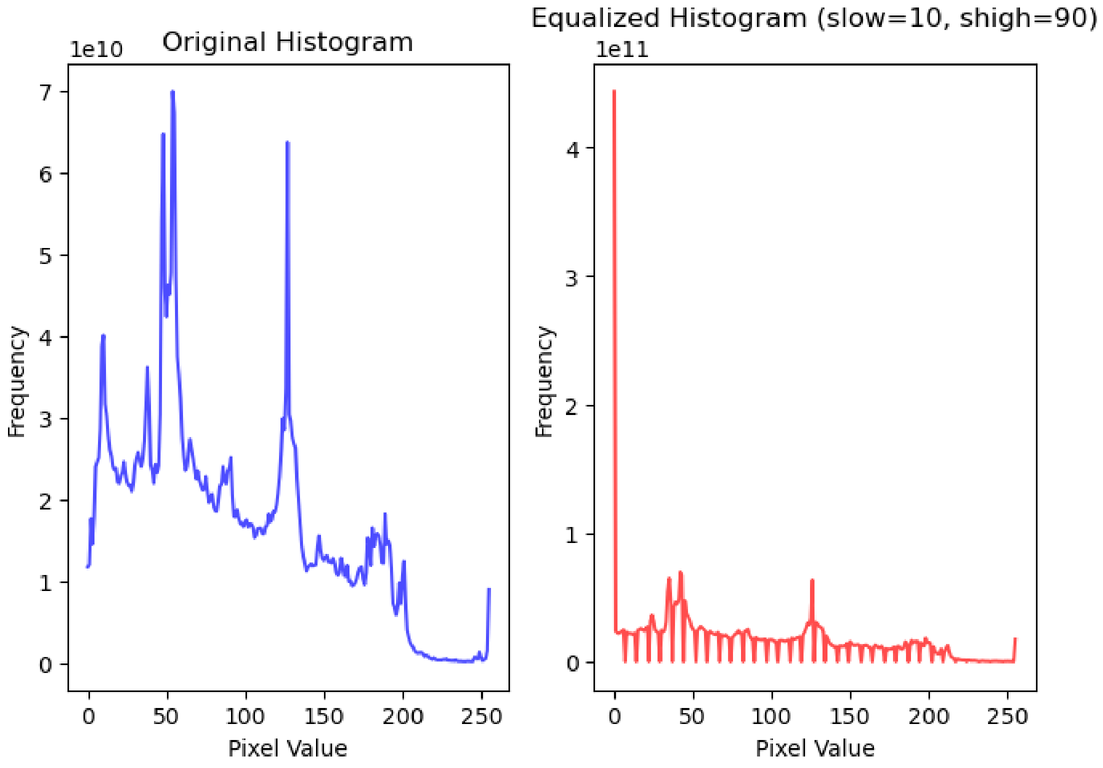
```
plt.figure(figsize=(5, 5))
plt.imshow(bw_image, cmap='gray')
plt.title('gray Input')
plt.axis('off')
plt.show()


Output,original_histogram,equalized_histogram=histogram_equalization(Input,Slow,Shigh)␣
  ↪# call histogram function
Show(Output,original_histogram,equalized_histogram)


Slow=30      # enter Slow
Shigh=70    # enter Shigh

Output,original_histogram,equalized_histogram=histogram_equalization(Input,Slow,Shigh)␣
  ↪# call histogram function
Show(Output,original_histogram,equalized_histogram)


Slow=50      # enter Slow
Shigh=50    # enter Shigh

Output,original_histogram,equalized_histogram=histogram_equalization(Input,Slow,Shigh)␣
  ↪# call histogram function
Show(Output,original_histogram,equalized_histogram)
```

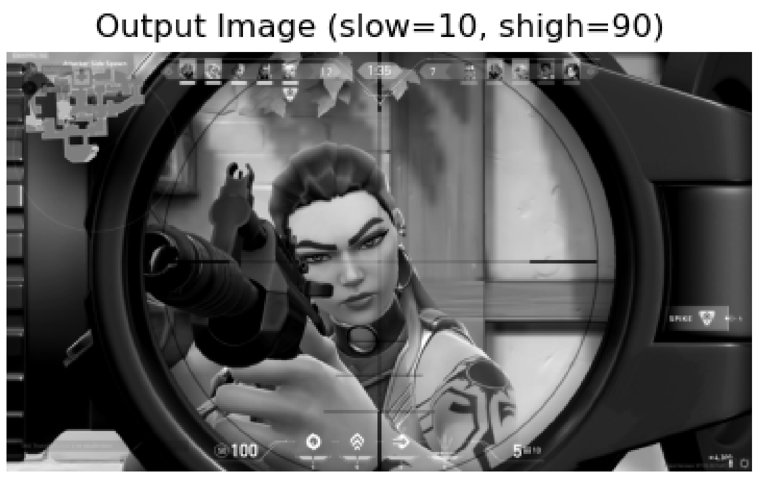
Input Image

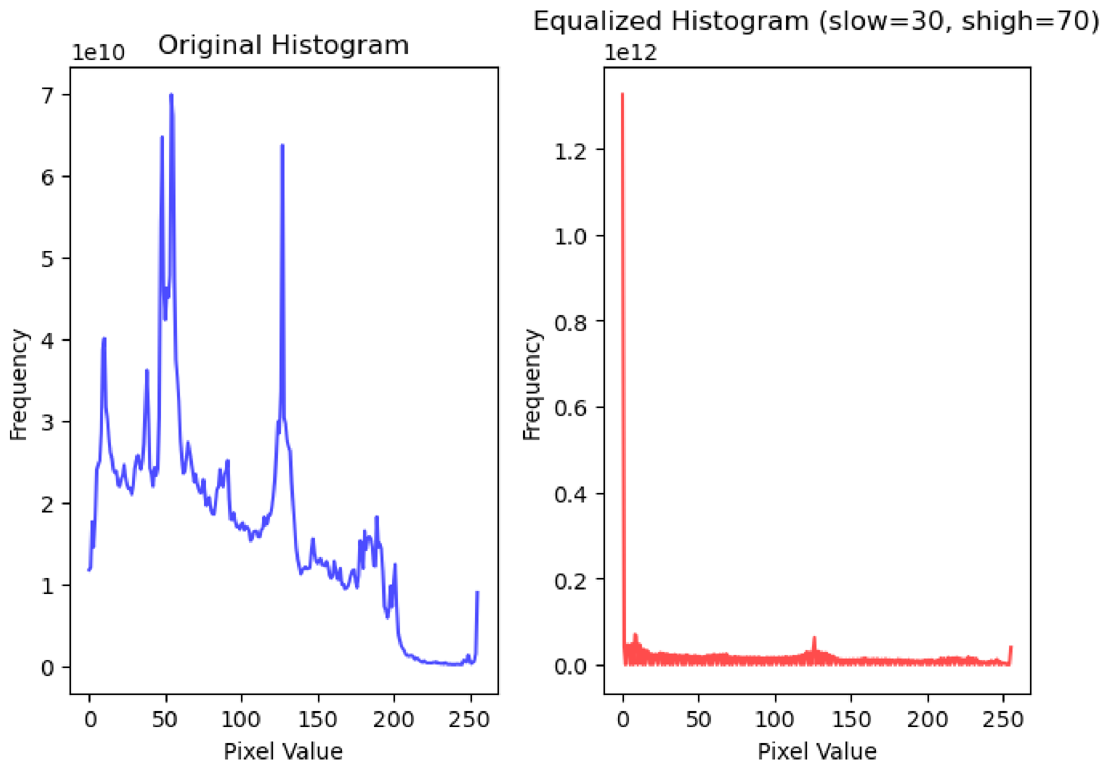


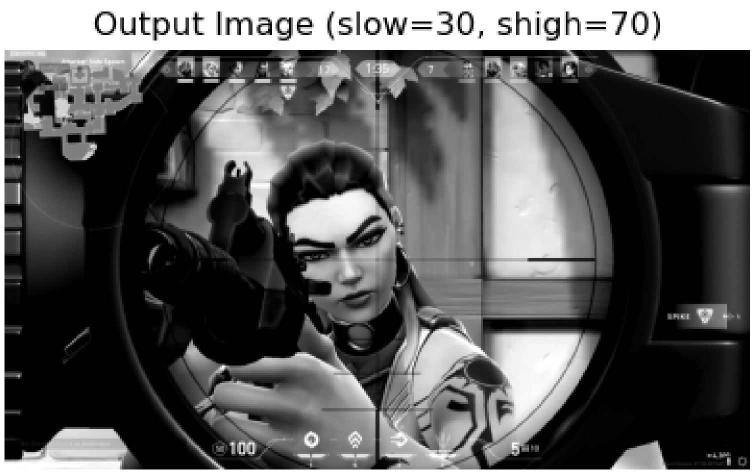
gray Input

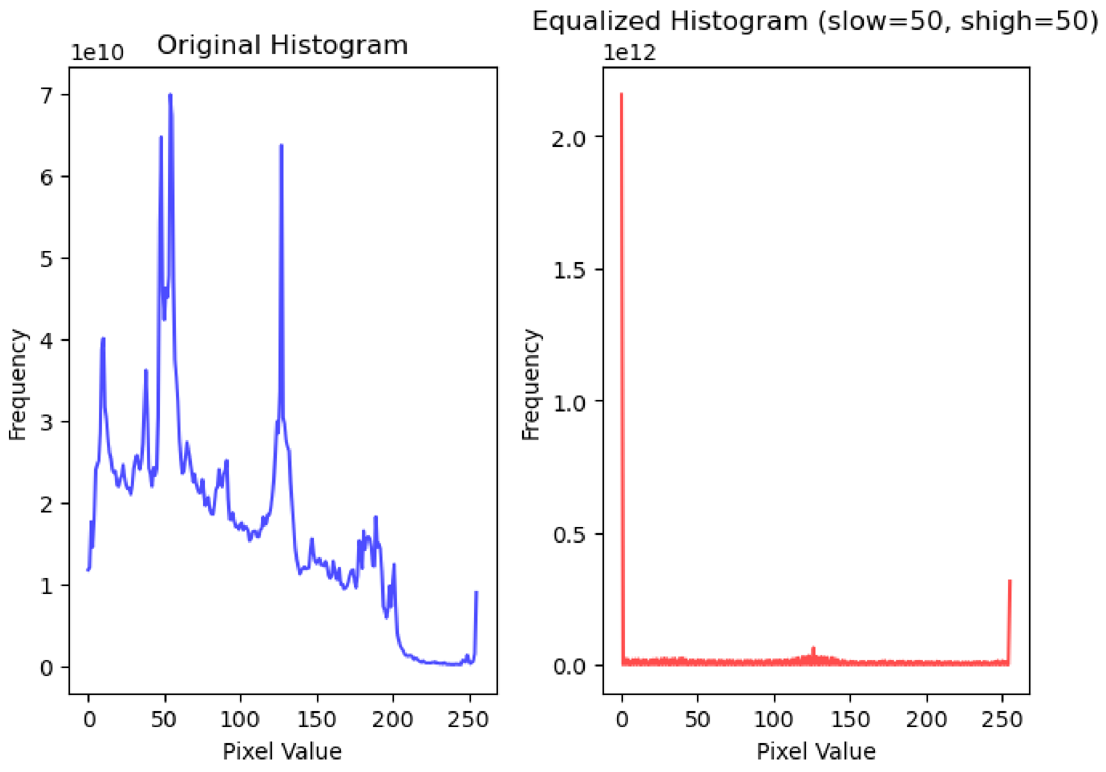



Original Histogram
1e10
Frequency
Pixel Value
0
50
100
150
200
250
0
1
2
3
4
5
6
7
Equalized Histogram (slow=10, shigh=90)
1e11
Frequency
Pixel Value
0
1
2
3
4

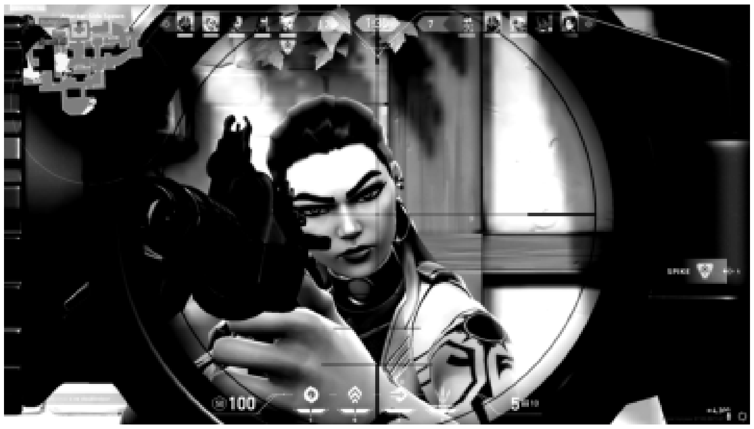

Output Image (slow=10, shigh=90)
100
5


Original Histogram
1e10
Frequency
Pixel Value
0
1
2
3
4
5
6
7
0
50
100
150
200
250
Equalized Histogram (slow=30, shigh=70)
1e12
Frequency
Pixel Value
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
50
100
150
200
250


Output Image (slow=30, shigh=70)


Original Histogram
1e10
Frequency
Pixel Value
Equalized Histogram (slow=50, shigh=50)
1e12
Frequency
Pixel Value

Output Image (slow=50, shigh=50)



```
[ ]:
```