Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 7 of DIP

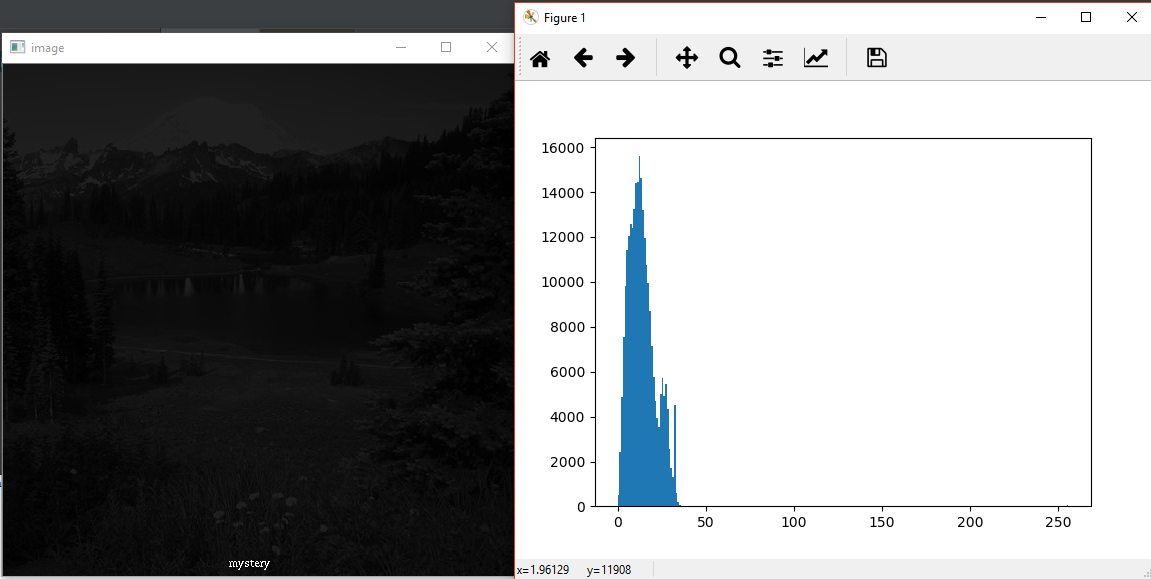
**Task 1:**

**Code:**

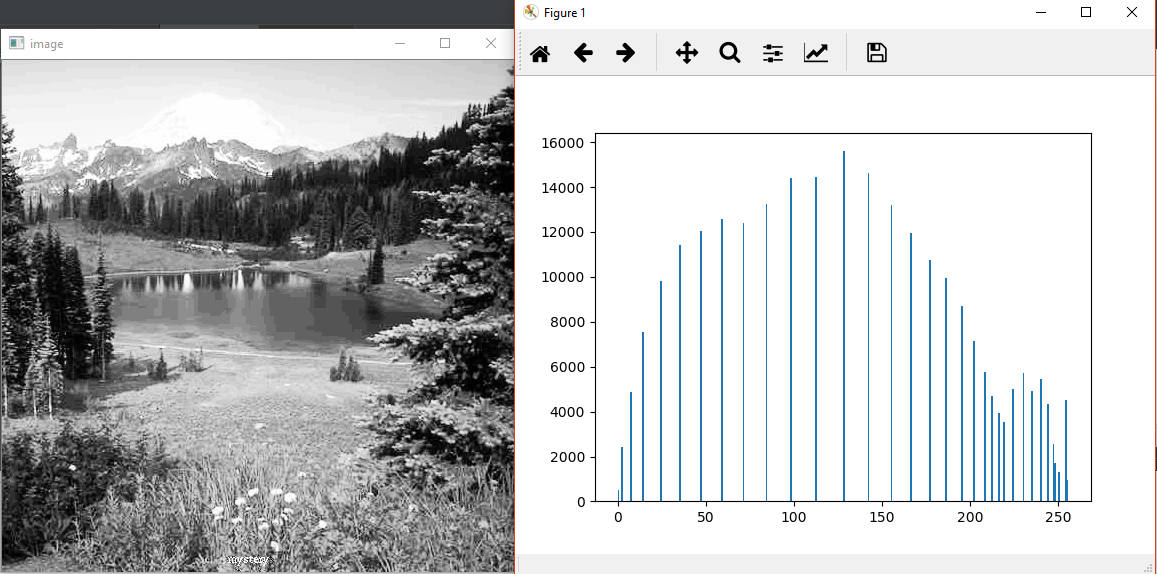
**from** matplotlib **import** pyplot **as** plt  
**import** cv2  
  
**def** show\_hist(img): *#to show histograms* plt.hist(img.ravel(), 256, [0, 256])  
 plt.show()  
  
img = cv2.imread(**'lab07.jpg'**,0)  
cv2.imshow(**'image'**,img)  
show\_hist(img)  
equ = cv2.equalizeHist(img)  
cv2.imshow(**'image'**,equ)  
show\_hist(equ)

**Screenshot:**

Before equalization



After equalization:



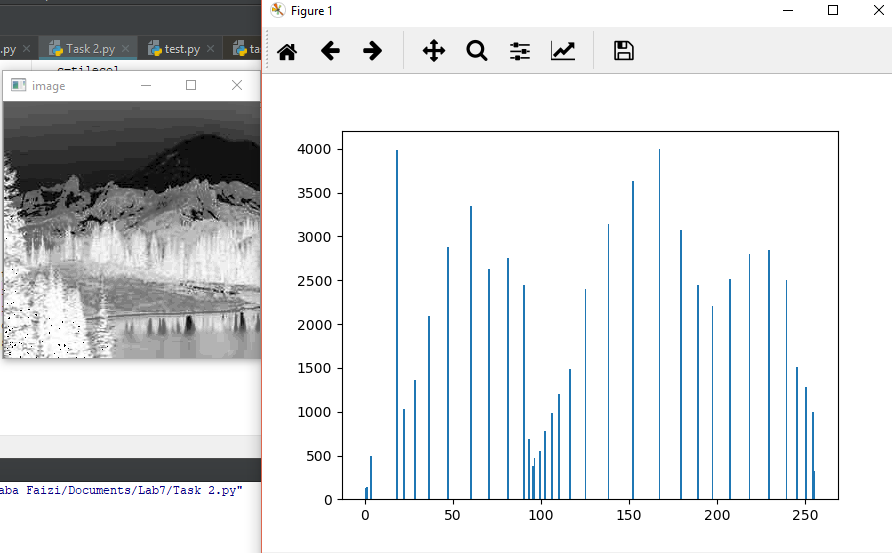
**Task 2:**

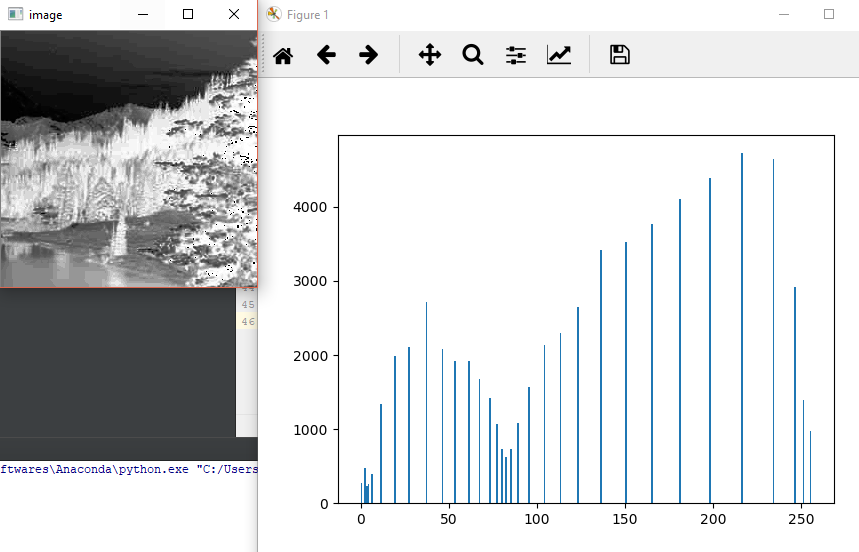
**Code:**

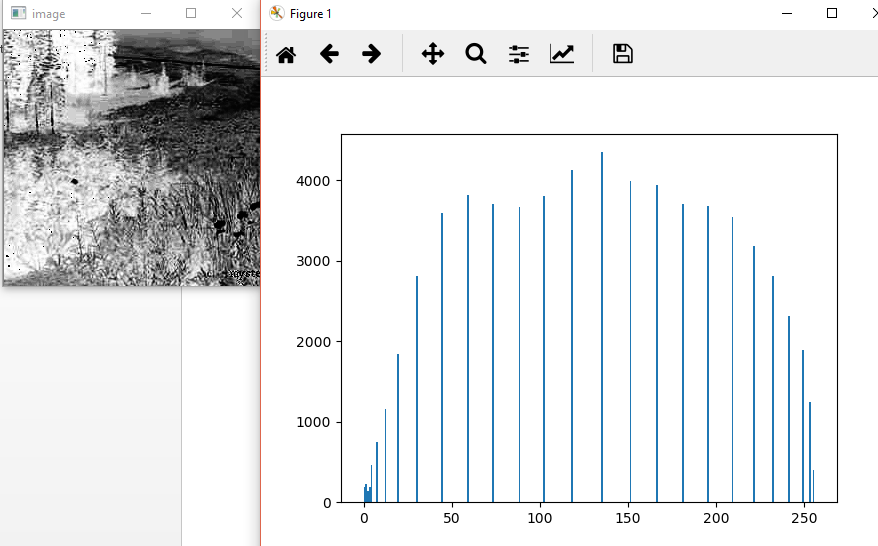
**from** matplotlib **import** pyplot **as** plt  
**import** numpy **as** np  
**import** cv2  
  
**def** show\_hist(img): *#to show histograms* plt.hist(img.ravel(), 256, [0, 256])  
 plt.show()  
  
**def** equalize(img):  
 **return** cv2.equalizeHist(img)  
  
**def** tiling(img,tiles):  
 arr=np.asarray(img)  
 rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 list=[] *#containing arrays of the tiles of image* tilecol=int(cols/2)  
 tilerow=int(rows/2)  
 c=0  
 d=0  
 **for** a **in** range(0,tiles):  
 **if** tilecol>cols: *#moving the tile to the lower side* d=tilerow  
 tilerow += rows/2  
 tilecol=cols/2  
 c=0  
 n = arr[int(d):int(tilerow),int(c):int(tilecol)] *#slicing the original array to get individual tiles* c=tilecol  
 tilecol+=cols/2  
 im = np.array(n \* 255, dtype=np.uint8)  
 img=equalize(im)  
 cv2.imshow(**'image'**, img)  
 show\_hist(img)  
 n = np.asarray(img)  
 list.append(n)  
  
 arr1 = np.concatenate([list[0], list[1]], axis=1) *#compiling the individual equalized arrays* arr2 = np.concatenate([list[2], list[3]], axis=1)  
 arr = np.concatenate([arr1, arr2], axis=0)  
 cv2.imwrite(**'color\_img.jpg'**, arr)  
 cv2.imshow(**'Color image'**, arr)  
 **return** im  
  
**def** slidingwindow(img,col,row):  
 g=col  
 arr = np.asarray(img)  
 rows = arr.shape[0] *# width & height of the image i.e. no. of pixels* cols = arr.shape[1]  
 c=0  
 d=0  
 **for** a **in** range(0,(rows\*cols)):  
 **if** col>cols: *#for last column pixel, move the window to the next row* d+=1  
 row+=1  
 c=0  
 col=g  
 **if** row>rows: *#after last row limit of window, exit* **break** b=arr[int(d):int(row), int(c):int(col)]  
 im = np.array(b \* 255, dtype=np.uint8)  
 arr[int(d):int(row), int(c):int(col)]=np.asarray(equalize(im))  
 c+=1 *#for moving the window ahead one pixel at a time (same for rows also)* col+=1  
 cv2.imwrite(**'color\_img2.jpg'**, arr)  
 cv2.imshow(**'Color image2'**, arr)  
 **return** im  
  
  
img = cv2.imread(**'lab07.jpg'**,0)  
  
 *#-------tiling approach---------*tiles=4  
img1=tiling(img,tiles)  
show\_hist(img1)  
  
 *#-------sliding window approach--------*col=500 *#size of the window i.e. 500 x 500*row=500  
img2=slidingwindow(img,col,row)  
show\_hist(img2)

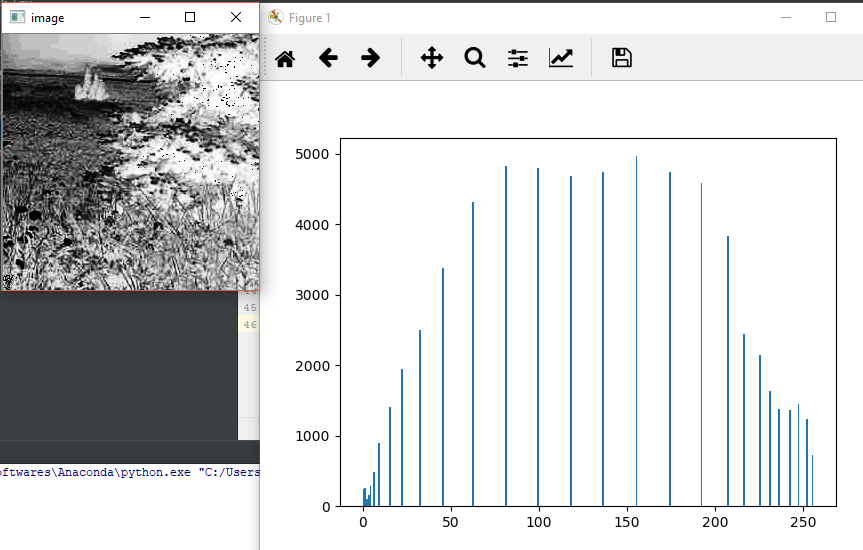
**Screenshot**

Part a (Tiles = 4):

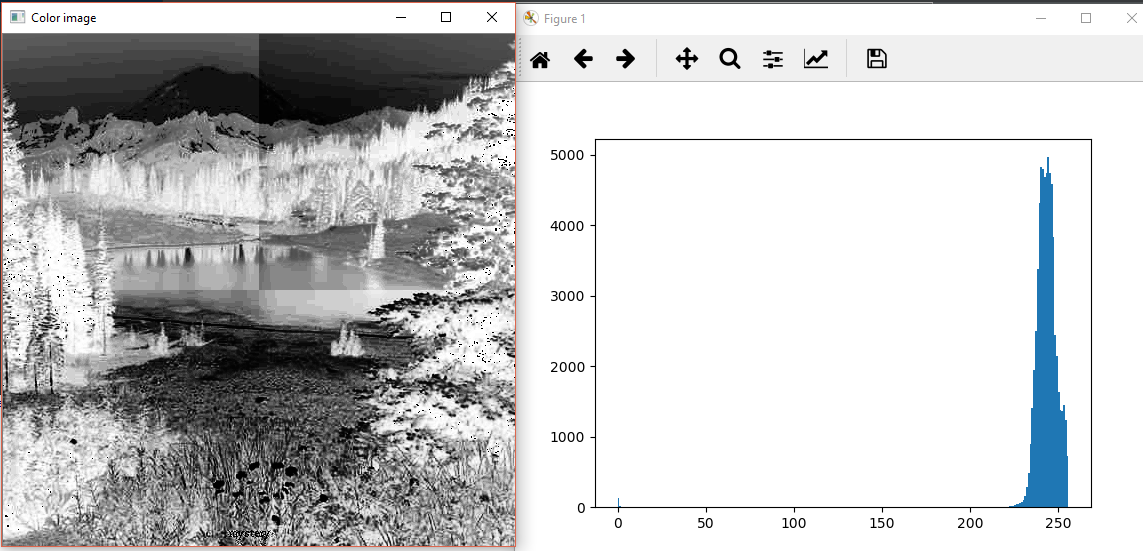




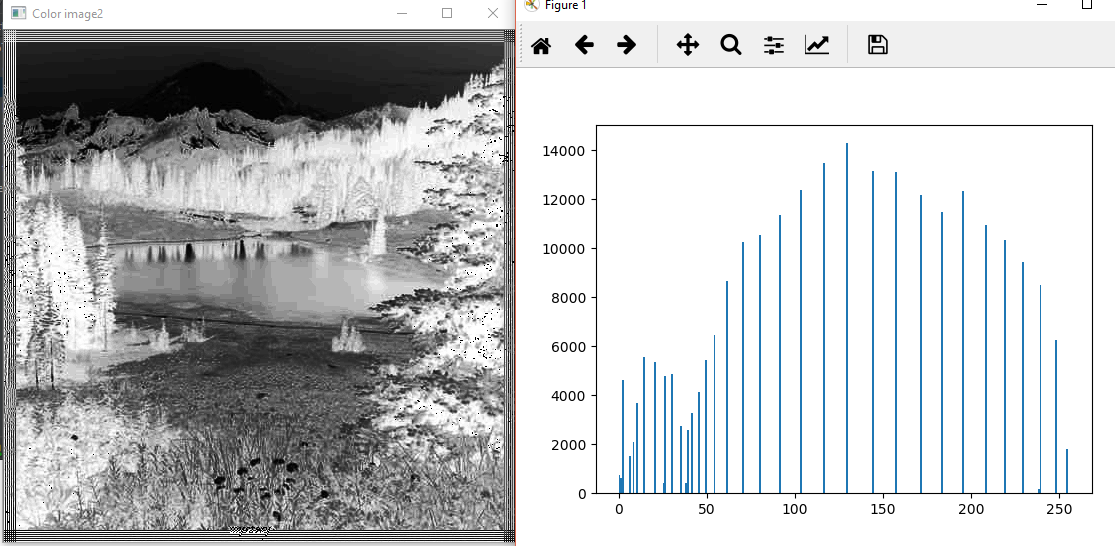




After compilation:



Part b (Window size = 500 x 500)



**Task 3:**

**Global Histogram Equalization:** The contrast of the overall image has increased a lot, in a smooth manner. The image has been enhanced giving the best result.

**Tiling Approach:** This gives a more blocky effect i.e. the image divided into tiles are equalized separately and not relative, so the difference in different tiles of the image can be seen through the blocky effect.

**Sliding Window Approach:** Brightness has increased and, decreasing the window size adds some disturbing lines on the sides of the image but the image has a smooth blending with the neighboring tiles, in the middle.