Contrast Adjustment

Histogram Equalization

```
Write a program that can compute the histogram of a grayscale image (ass
uming 256 levels of gray). In a separate
main program, apply the program to Camera Man image, and illustrate the
 histogram as a stem plot besides the
test image (using “subplot” function)
```

```
    in this exercise we just used opencv library for reading image file,
you can see out test images in test directory
```

In [74]:

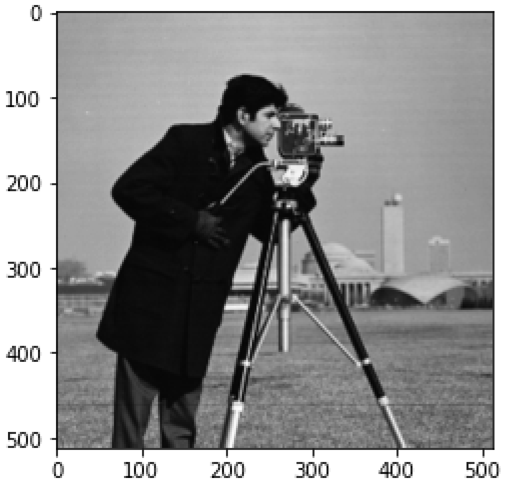
```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```

first we should read our image and for doing that we used opencv library,make sure that you installed this library before

In [75]:
```
def read_input(image_path):
    return cv.imread(image_path,cv.IMREAD_GRAYSCALE)
img=read_input('./test/Camera Man.bmp')
print(img)
plt.imshow(img,cmap='gray', vmin=0, vmax=255)
```

```
[[156 157 160 ... 152 152 152]
 [156 157 159 ... 152 152 152]
 [158 157 156 ... 152 152 152]
 ...
 [121 123 126 ... 121 113 111]
 [121 123 126 ... 121 113 111]
 [121 123 126 ... 121 113 111]]
```

Out[75]:
```
<matplotlib.image.AxesImage at 0x2157f567a58>
```



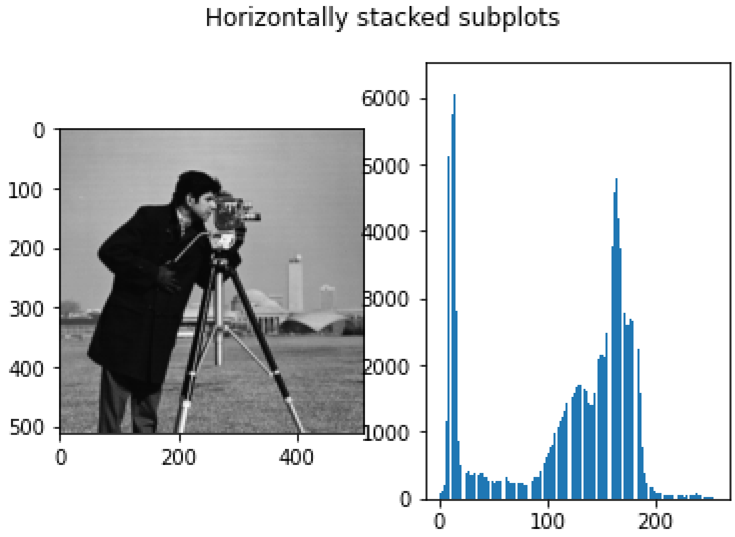
these some few step to take for creating histogram

```
    1.becuse our image pixel range are reperesnted in range between 0 to
256 we need an empty array in size of 256.


    2.just count the number of ferequent of each pixel range
```

In [76]:
```
def create_histogram(img):
    bins = np.zeros(256, np.int32)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            if img[i][j]!=-1:
                bins[img[i][j]]+=1
    return bins
histogram=create_histogram(img)
```

for plotting our histogram we should use matplotlib.

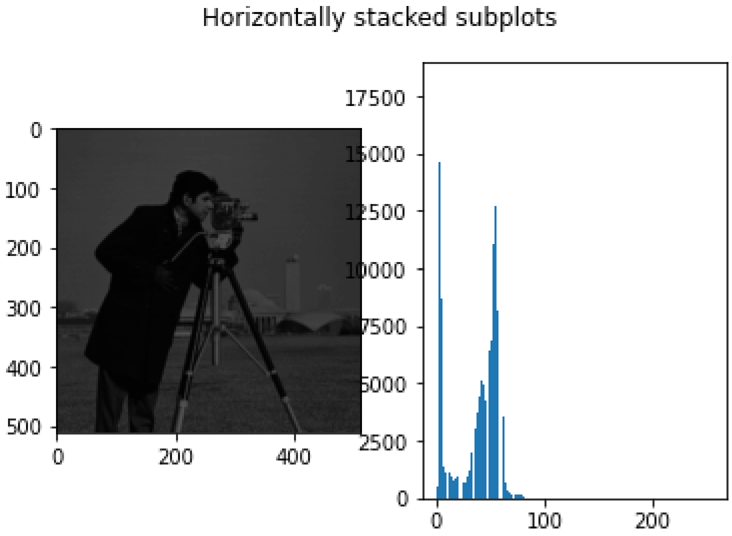
```
In [40]: def plot_histogram(img,histogram):
             fig, (ax1, ax2) = plt.subplots(1, 2)
             x=[]
             for i in range(len(histogram)):
                 x.append(i)
             fig.suptitle('Horizontally stacked subplots')
             ax1.imshow(img,cmap='gray', vmin=0, vmax=255)
             ax2.bar(x,histogram)
         plot_histogram(img,histogram)
```



Decrease the brightness of Camera Man by dividing the intensity values by 3 and named output as D

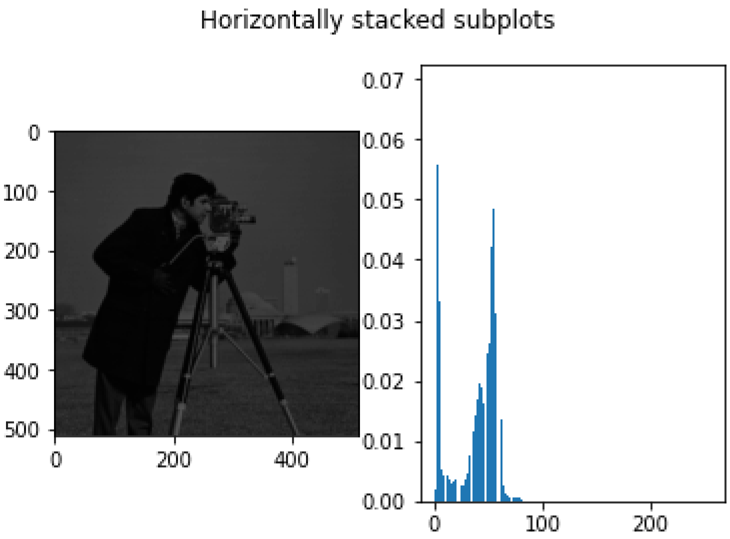
In [42]:

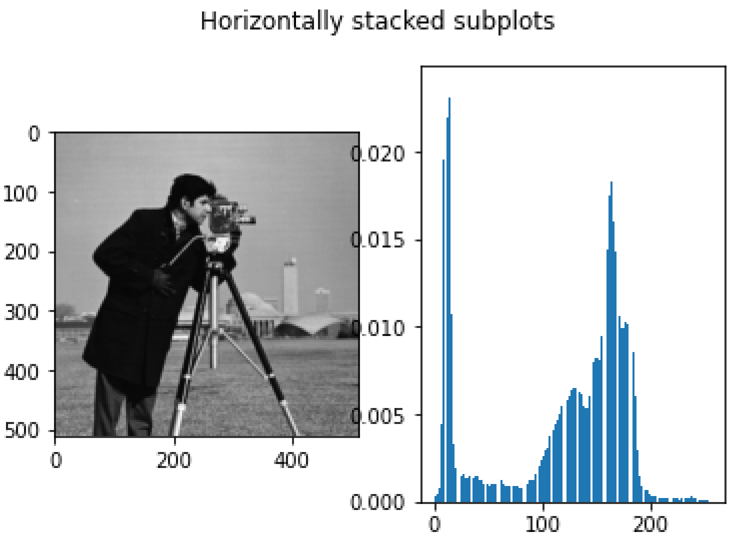
```
def decrease_brightness(img):
    new_image = np.zeros((img.shape[0],img.shape[1]),dtype=int)
    new_image = np.array(new_image)
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            new_image[i][j]=int(img[i][j]/3)
#     plt.imshow(new_image,cmap='gray', vmin=0, vmax=255)
    return new_image
D=decrease_brightness(img)
D_histogram=create_histogram(D)
plot_histogram(D,D_histogram)
```



Normalize an histogram is a technique consisting into transforming the discrete distribution of intensities into a discrete distribution of probabilities. To do so, we need to divide each value of the histogram by the number of pixel. Because a digital image is a discrete set of values that could be seen as a matrix and it's equivalent to divide each nk by the dimension of the array which is the product of the width by the length of the image.

```
In [62]: def create_pdf(histogram,img_w,img_h):
             pdf=[]
             size=img_w*img_h
             for i in histogram:
                 pdf.append(i/size)
             return pdf
         D_pdf=create_pdf(D_histogram,D.shape[0],D.shape[1])
         plot_histogram(D,D_pdf)
         img_pdf=create_pdf(histogram,img.shape[0],img.shape[1])
         plot_histogram(img,img_pdf)
```





The processing of histogram equalization relies on the use of the cumulative probability function (cdf). The cdf is a cumulative sum of all the probabilities lying in its domain and defined by:

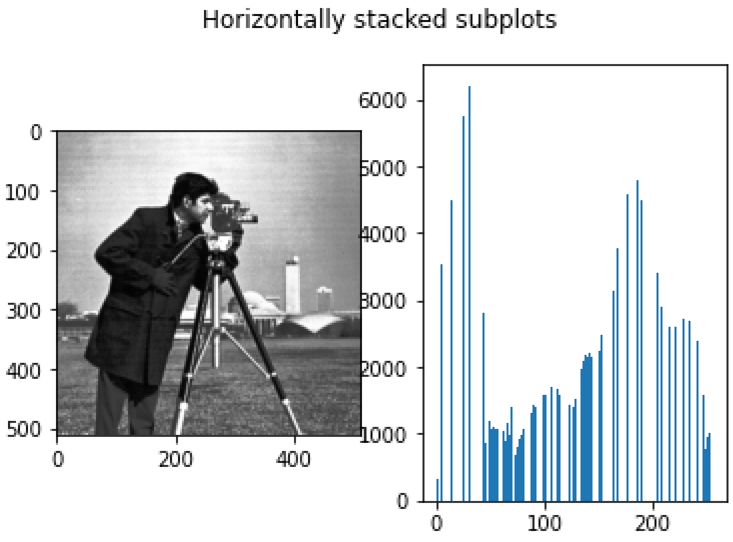
```
cdf[i]=sum(pdf[0:i])
```

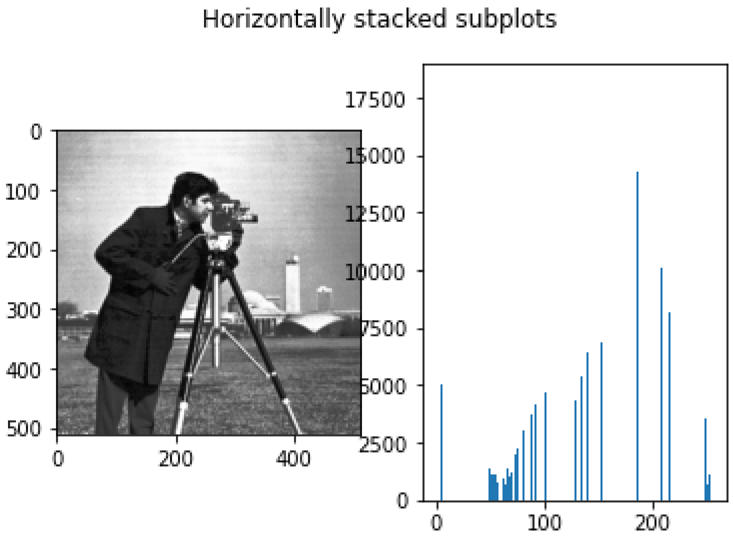
The idea of this processing is to give to the resulting image a linear cumulative distribution function. Indeed, a linear cdf is associated to the uniform histogram that we want the resulting image to have.

In [63]:

```
def calculate_cdf(pdf):
    sum=0
    cdf=[]
    for i in pdf:
        sum+=i
        cdf.append(sum)
    return cdf
D_cdf=calculate_cdf(D_pdf)
cdf=calculate_cdf(img_pdf)
```

```
In [176]: def global_histogram_equalization(cdf,pixel_range,img):
              new_pd=[]
              for i in cdf:
                  new_pd.append(i*pixel_range)
              new_image = np.zeros((img.shape[0],img.shape[1]),dtype=int)
              for i in range(0, img.shape[0]):
                  for j in range(0, img.shape[1]):
                      if img[i][j]!=-1:
                          new_image[i][j]=int(new_pd[img[i][j]])
                      else:
                          new_image[i][j]=-1
          #     plt.imshow(new_image,cmap='gray', vmin=0, vmax=255)
              return new_pd,new_image
          D_new_pdf,H=global_histogram_equalization(D_cdf,255,D)
          img_new_pdf,new_img=global_histogram_equalization(cdf,255,img)
          H_histogram=create_histogram(H)
          img_histogram=create_histogram(new_img)
          plot_histogram(new_img,img_histogram)
          plot_histogram(H,H_histogram)
```





In [72]:

```
def region_contranst_sampleing(img,sampling_rate):
    new_img=np.full((img.shape[0], img.shape[1]),-1)
    for i in range(0,img.shape[0],sampling_rate):
        for j in range(0,img.shape[1],sampling_rate):
            new_image[i][j]=img[i][j]
    return new_img
```

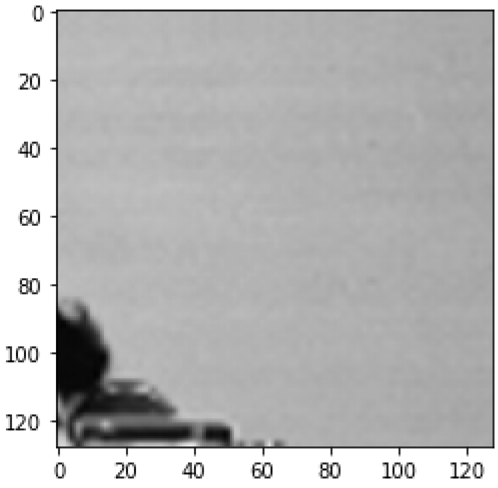
In [240]:

```
def contrast_region_devision(img,region_size):
    arr=img

    """
    Return an array of shape (n, nrows, ncols) where
    n * nrows * ncols = arr.size

    If arr is a 2D array, the returned array should look like n subblocks with
    each subblock preserving the "physical" layout of arr.
    """
    h, w = arr.shape
    nrows=int(h/region_size)
    ncols=int(w/region_size)
    assert h % nrows == 0, "{} rows is not evenly divisble by {}".format(h, nrows
    assert w % ncols == 0, "{} cols is not evenly divisble by {}".format(w, ncols
    return (arr.reshape(h//nrows, nrows, -1, ncols)
               .swapaxes(1,2)
               .reshape(-1, nrows, ncols))
images=contrast_region_devision(img,4)
print(images.shape)
plt.imshow(images[2],cmap='gray', vmin=0, vmax=255)
```

```
(16, 128, 128)
```

Out[240]:

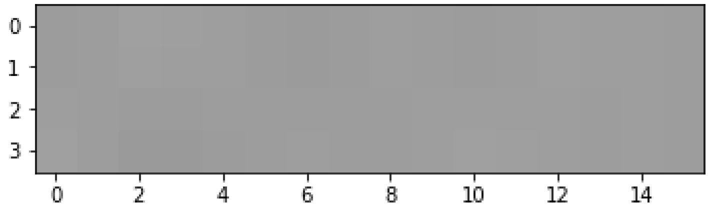
```
<matplotlib.image.AxesImage at 0x2153d039e48>
```



In [232]:
```
def reverse_sampleing(img,sampling_rate):
    for i in range(0,img.shape[0],sampling_rate):
        for j in range(0,img.shape[1],sampling_rate):
            new_image[i][j]=img[i][j]
    return new_img
```

In [233]:
```
def concat_images_horizontali(images):
    return np.concatenate(images,axis=1)
new_image=concat_images_horizontali(images[0:4])
plt.imshow(new_image,cmap='gray', vmin=0, vmax=255)
```

Out[233]:
```
<matplotlib.image.AxesImage at 0x2153d3b1390>
```

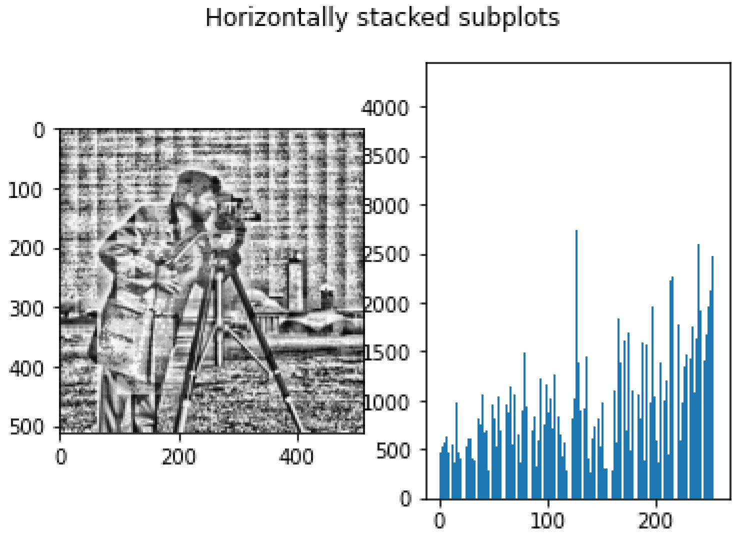


In [234]:
```
def concat_images_verticali(images):
    return np.concatenate(images,axis=0)
```

In [244]:

```
def LHE(img,window_size):
    img=np.array(img)
    print(img.shape)
    bw=window_size-(img.shape[0]%window_size) if (img.shape[0]%window_size) !=0 e
    bh=window_size-(img.shape[1]%window_size) if (img.shape[1]%window_size) !=0 e
    resized_img=np.zeros((img.shape[0]+bw,img.shape[1]+bh),dtype=int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            resized_img[i][j]=img[i][j]
    print(window_size)
    divided_images=contrast_region_devision(resized_img,window_size)
    new_images=[]
    for image in divided_images:
        histogram=create_histogram(image)
        pdf=create_pdf(histogram,image.shape[0],image.shape[1])
        cdf=calculate_cdf(pdf)
        new_pdf,ghe=global_histogram_equalization(cdf,255,image)
        new_images.append(ghe)
    rows=[]
    for i in range(0,len(new_images),window_size):

        rows.append(concat_images_horizontali(new_images[i:window_size+i]))
    new_image=concat_images_verticali(rows)
#     plt.imshow(new_image,cmap='gray', vmin=0, vmax=255)
    return new_image
L=LHE(D,16)
l_histogram=create_histogram(L)
plot_histogram(L,l_histogram)
```

```
(512, 512)
16
```

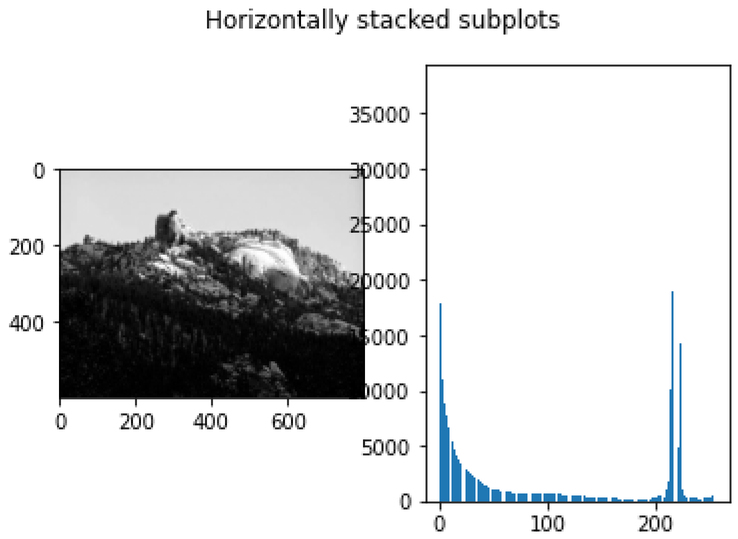


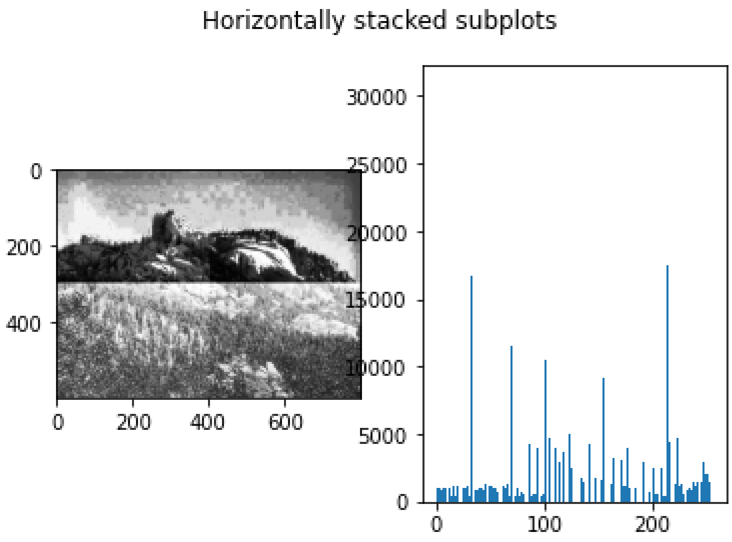
2.2.1. Implement a local histogram equalization with different windows size for the HE1,2,3, and 4 images. Explain and display the results. Discuss the effects of increasing window size and compare it with global histogram equalization in detail.

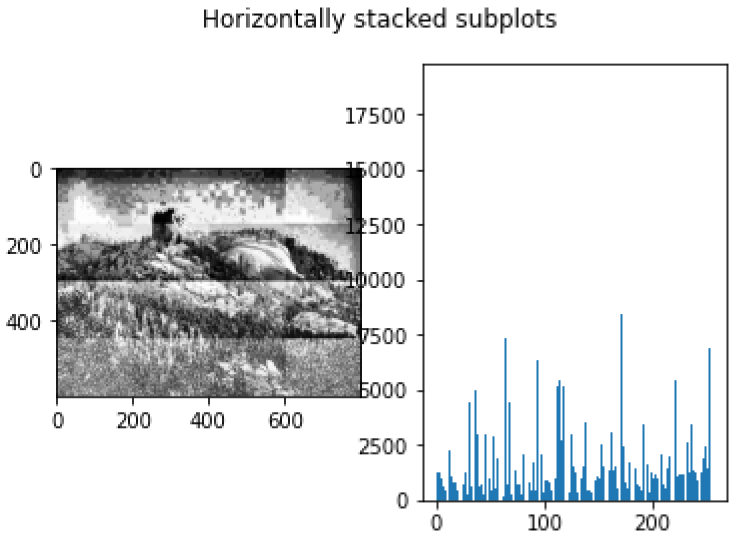
In [250]:

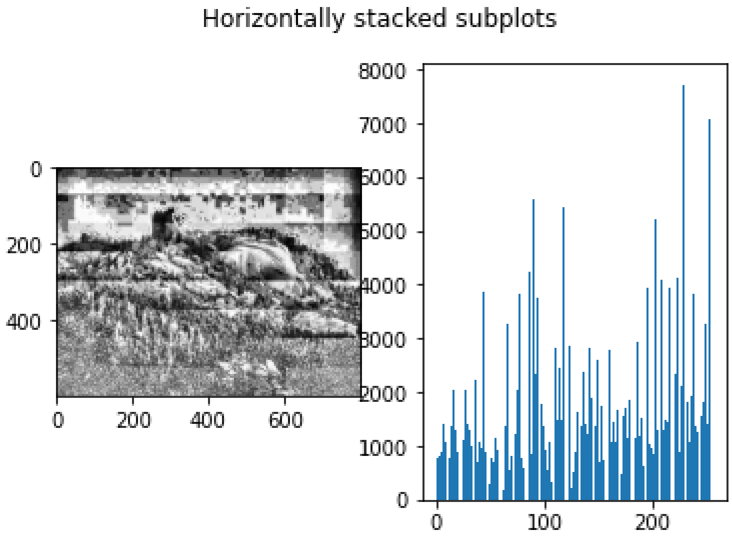
```
HE1=read_input('./test/HE1.jpg')
l_histogram=create_histogram(HE1)
plot_histogram(HE1,l_histogram)
LHE1=LHE(HE1,2)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,4)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,8)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,25)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,100)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
```

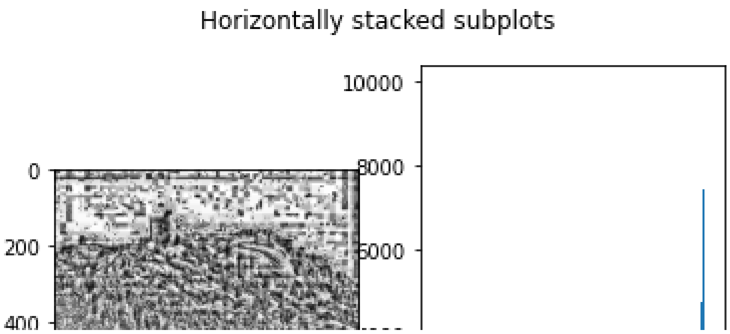
```
(599, 799)
2
(599, 799)
4
(599, 799)
8
(599, 799)
25
(599, 799)
100
```

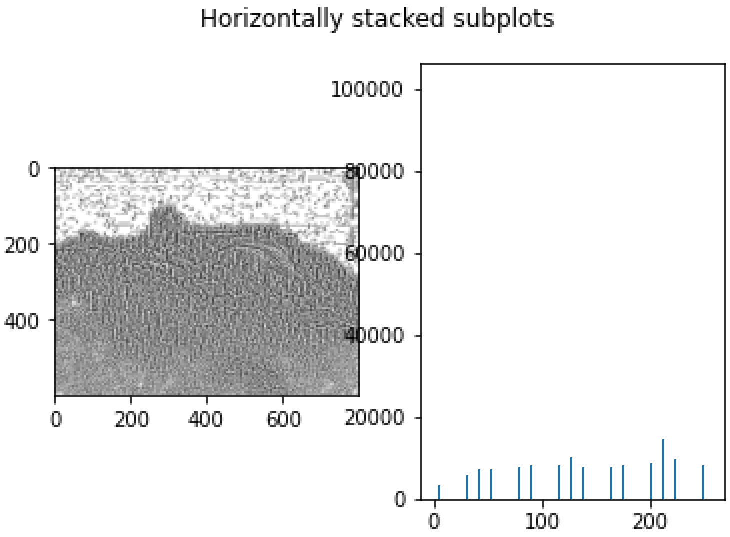



Horizontally stacked subplots
0
200
400
0
200
400
600
30000
25000
20000
15000
10000
5000
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
17500
15000
12500
10000
7500
5000
2500
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
8000
7000
6000
5000
4000
3000
2000
1000
0
0
100
200

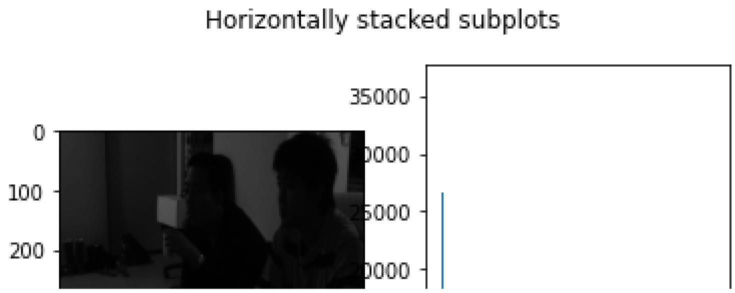

Horizontally stacked subplots
0
200
400
10000
8000
6000


Horizontally stacked subplots
0
200
400
0
200
400
600
100000
80000
60000
40000
20000
0
0
100
200

In [251]:

```
HE1=read_input('./test/HE2.jpg')
l_histogram=create_histogram(HE1)
plot_histogram(HE1,l_histogram)
LHE1=LHE(HE1,2)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,4)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,8)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,25)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,100)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
```

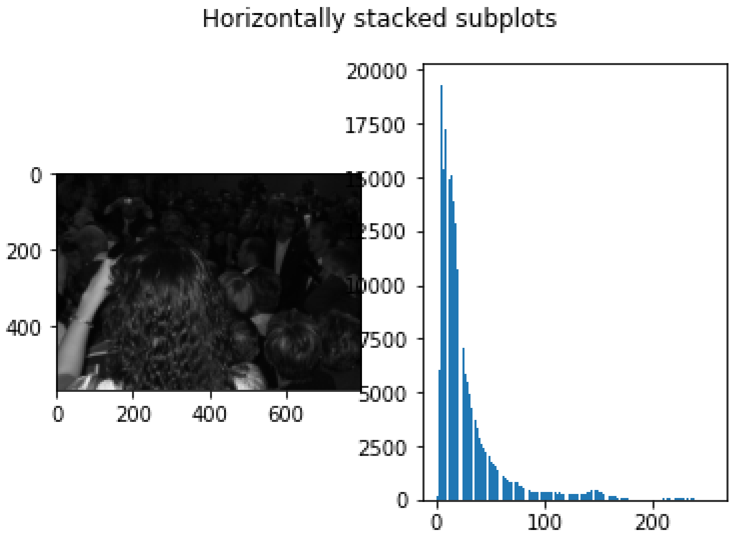
```
(512, 512)
2
(512, 512)
4
(512, 512)
8
(512, 512)
25
(512, 512)
100
```

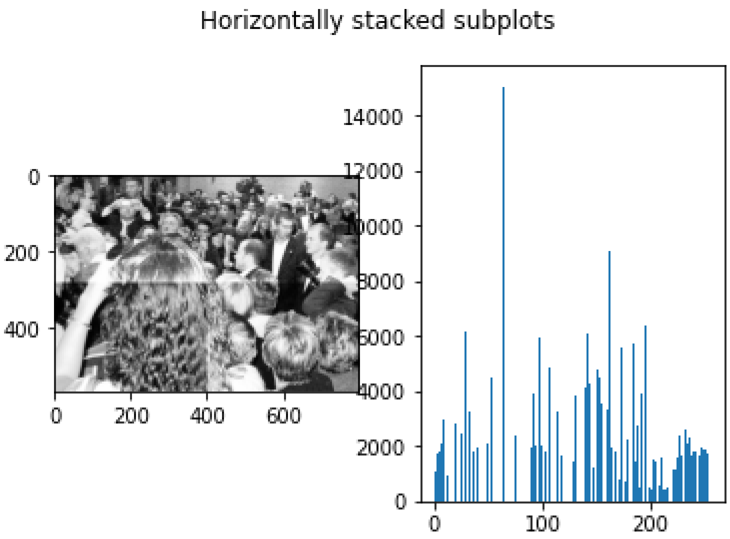


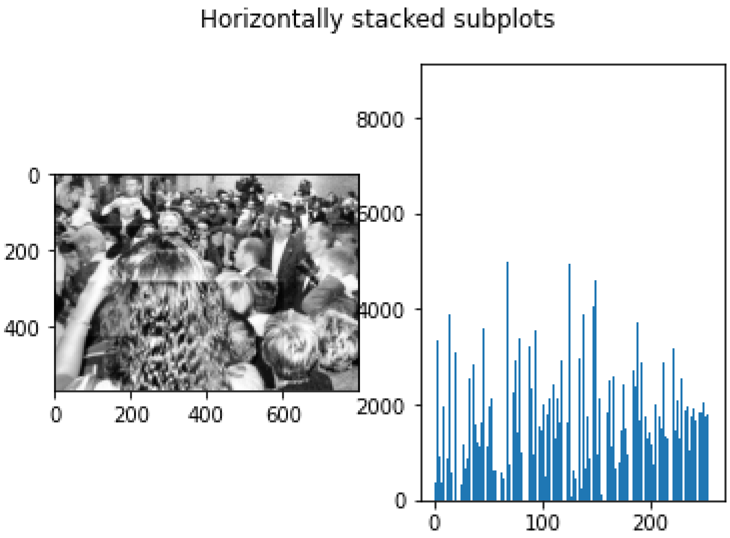
In [252]:

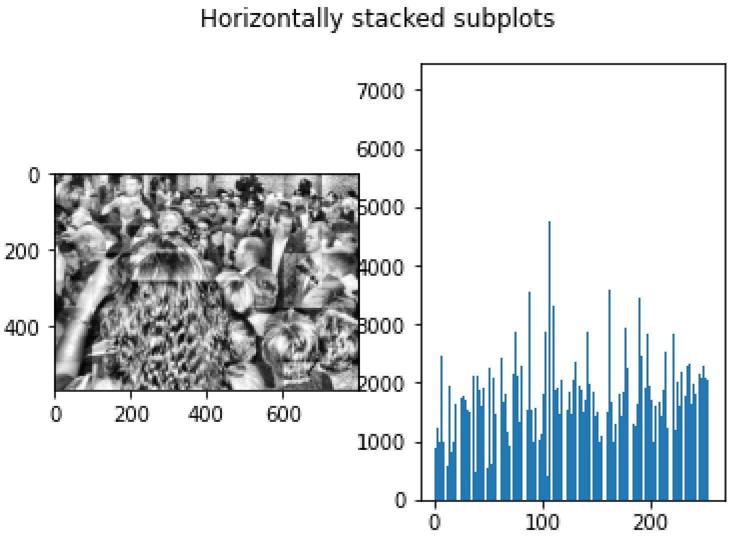
```
HE1=read_input('./test/HE3.jpg')
l_histogram=create_histogram(HE1)
plot_histogram(HE1,l_histogram)
LHE1=LHE(HE1,2)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,4)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,8)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,25)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
LHE1=LHE(HE1,100)
LHE1_histogram=create_histogram(LHE1)
plot_histogram(LHE1,LHE1_histogram)
```

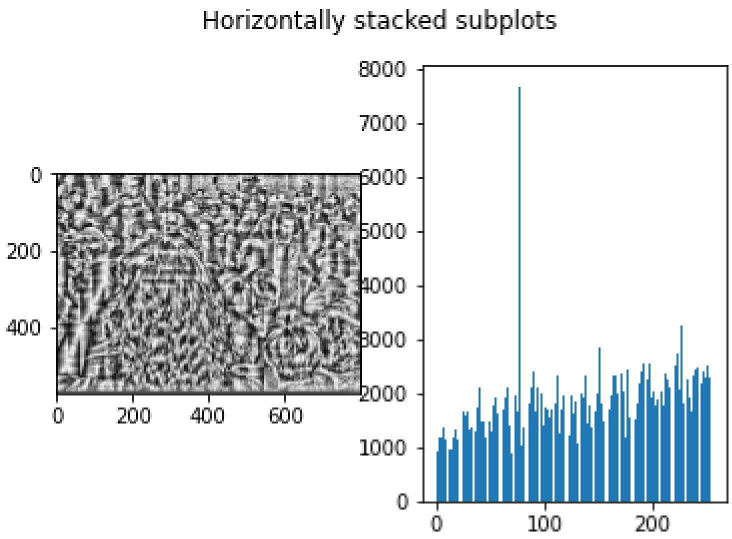
```
(568, 797)
2
(568, 797)
4
(568, 797)
8
(568, 797)
25
(568, 797)
100
```

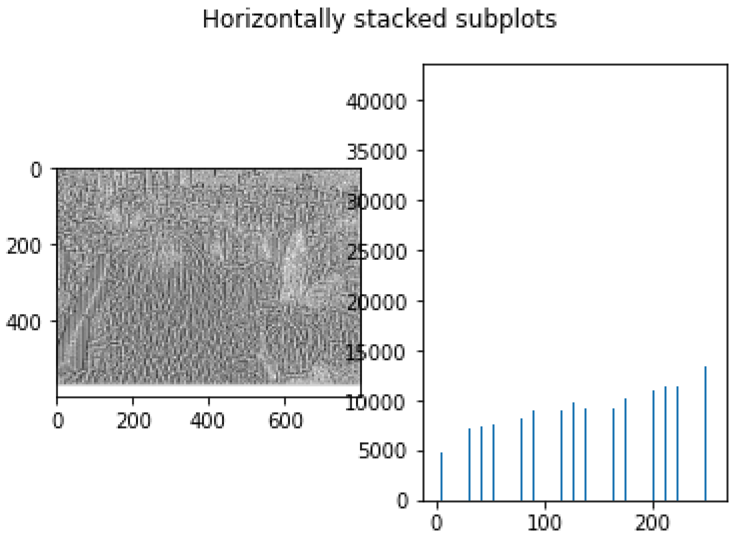



Horizontally stacked subplots
0
200
400
0
200
400
600
14000
12000
10000
8000
6000
4000
2000
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
8000
6000
4000
2000
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
7000
6000
5000
4000
3000
2000
1000
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
8000
7000
6000
5000
4000
3000
2000
1000
0
0
100
200


Horizontally stacked subplots
0
200
400
0
200
400
600
40000
35000
30000
25000
20000
15000
10000
5000
0
0
100
200

```
In [ ]:
```