Mansoura University
Faculty of Computers and Information
Department of Information Technology
Second Semester- 2024-2025

Digital Image Processing

Grade:THREE

Eng. Aya Ayad

Outline

- Histogram
- Histogram using calcHist( )
- Histogram using plt.hist( )
- Histogram Equalization

Histogram

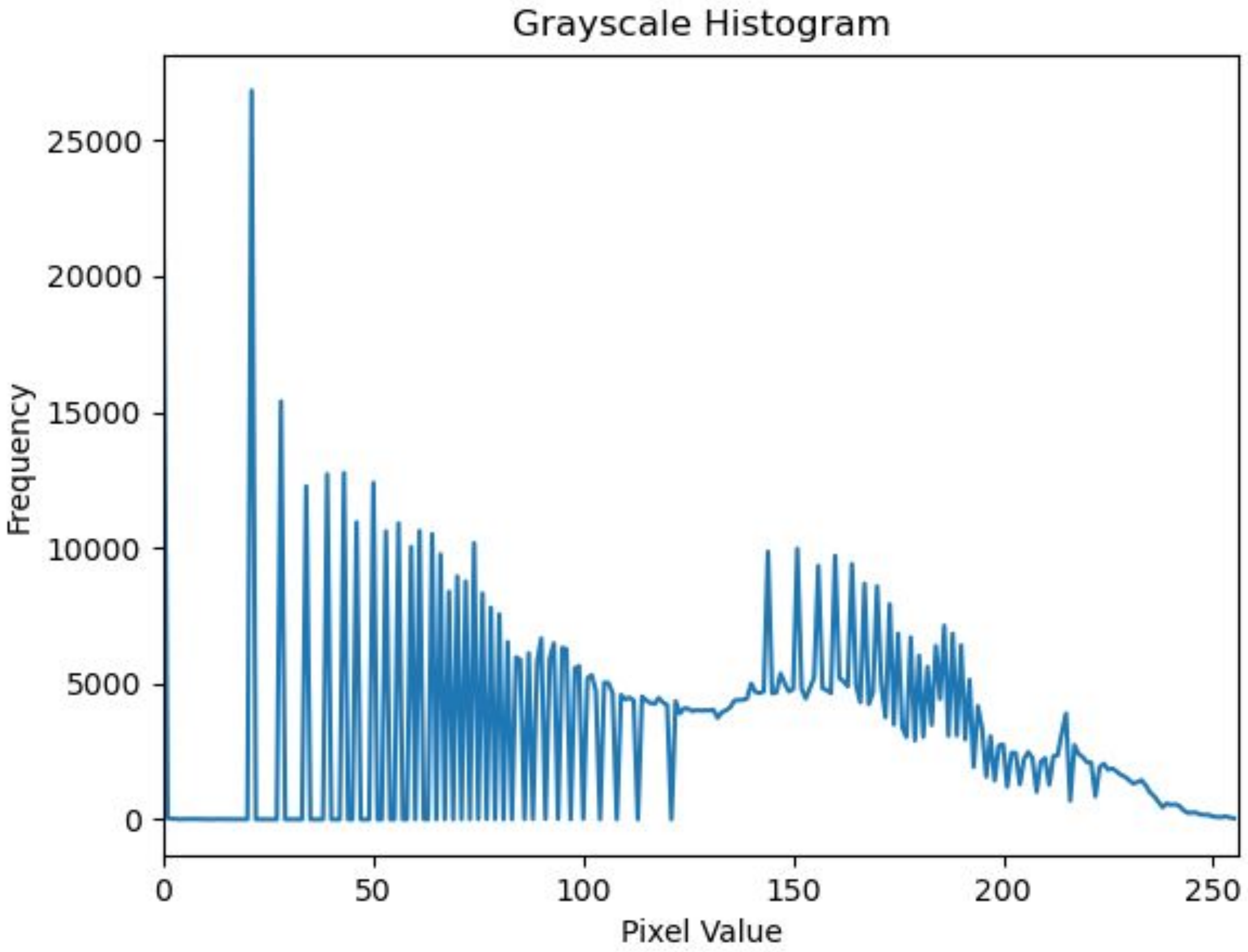
- A **histogram** is a graphical representation of the pixel intensity distribution in an image.
- It shows the **frequency** of each pixel intensity in a **grayscale image** or a specific color channel of a color image.
- In **grayscale images**, the pixel values range from **0 to 255**, where:
  - **0** represents black,
  - **255** represents white,
  - Values in between represent different shades of gray.
- The histogram helps in analyzing the contrast, brightness, and exposure of an image.

- To compute an image histogram in **OpenCV**, we use the cv2.calcHist() function.
  - cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

Parameter	**Description**
images	The source image. It should be of type uint8 or float32. It is passed as a list: [img].
channels	The index of the channel to calculate the histogram for. - For a **grayscale image**, use [0]. - For a **color image**, you can pass [0], [1], or [2] to calculate the histogram for the **Blue, Green, or Red** channel, respectively.
mask	A mask image to focus on a specific region. If we want the histogram of the **full image**, we set it to None.
histSize	The number of bins (or intervals) used to represent the histogram. Typically, **[256]** is used to cover all pixel values from 0 to 255.
ranges	The range of pixel values. Normally, it is **[0,256]**, meaning pixel values from **0 to 255** are considered.

```
import cv2
from matplotlib import pyplot as plt

img=cv2.imread("flower.png", cv2.IMREAD_GRAYSCALE)
histo=cv2.calcHist([img], [0], None, [256], [0, 256])
plt.plot(histo)
plt.title("Grayscale Histogram")
plt.xlabel("Pixel Value")
plt.ylabel("Frequency")
plt.xlim([0, 256])
plt.show()
```

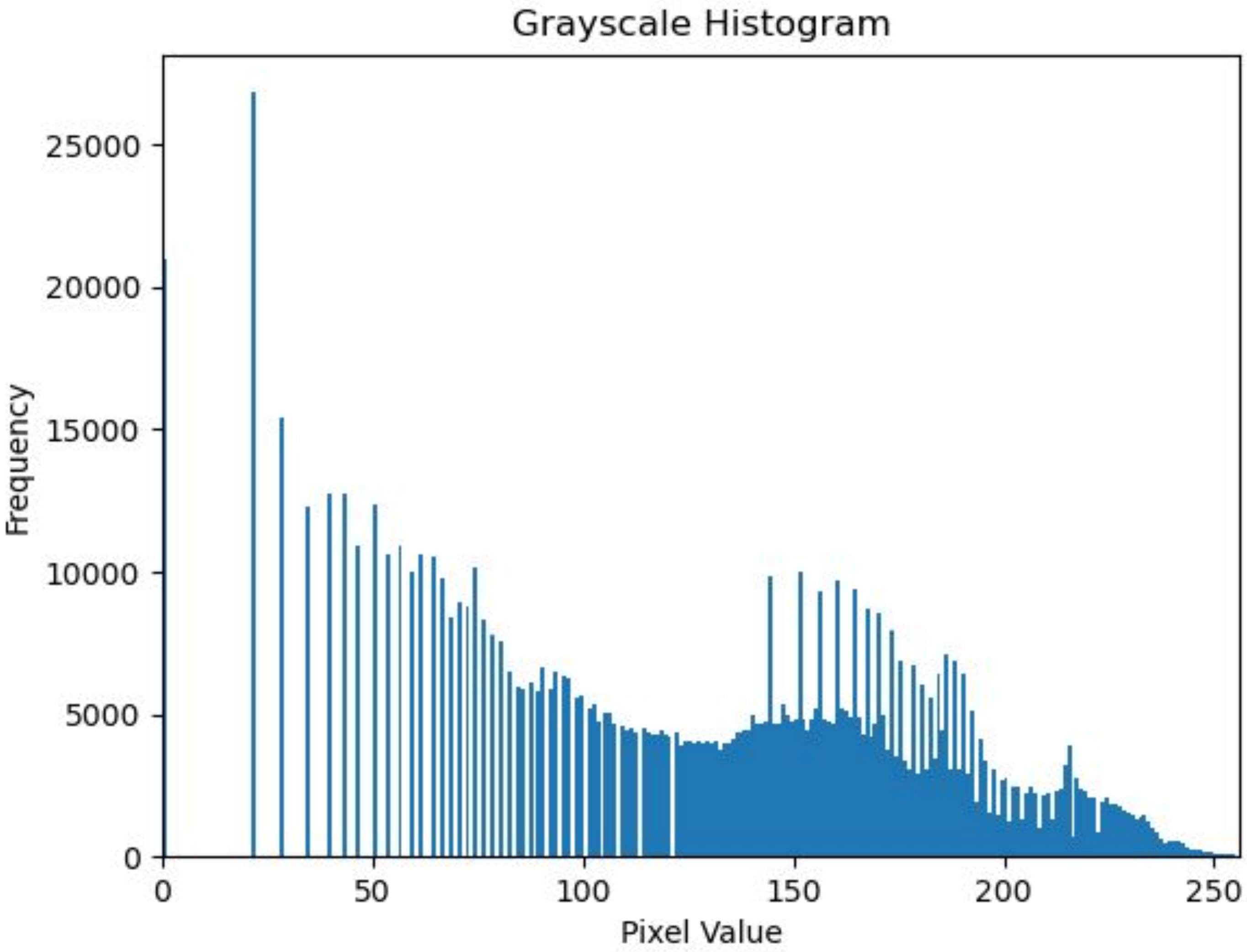

Grayscale Histogram
Frequency
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
Pixel Value

Histogram using plt.hist( )

```
import cv2
from matplotlib import pyplot as plt

img=cv2.imread("flower.png", cv2.IMREAD_GRAYSCALE)
plt.hist(img.ravel(),256,[0,256])
plt.title("Grayscale Histogram")
plt.xlabel("Pixel Value")
plt.ylabel("Frequency")
plt.xlim([0, 256])
plt.show()
```

Histogram using plt.hist( )



NOTE

- Pay attention to the difference in histogram results when converting an image using cvtColor versus changing the read mode.
- This difference occurs because of how OpenCV converts the image to grayscale.
- When using cvtColor, the grayscale value is calculated as **(0.299R + 0.587G + 0.114B)**, which accounts for human perception of color.
- When using the read mode (cv2.IMREAD_GRAYSCALE), the grayscale value is computed as **(R + G + B) / 3**, averaging the three channels equally.

Histogram Equalization

- Is a popular technique for improving the appearance of a poor image.

- It's function is similar to that of a histogram stretch but often provides more visually pleasing results across a wide range of images.

- Histogram equalization is a technique where the histogram of the resultant image is as flat as possible (with histogram stretching the overall shape of the histogram remains the same).

- Histogram equalization **redistributes** the pixel intensities to improve contrast.

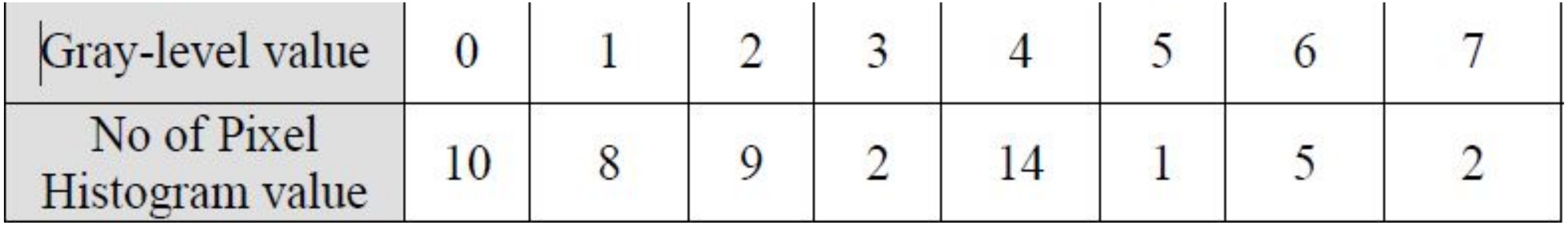
Histogram Equalization(cont.)

The histogram equalization process for digital images consists of four steps:

1. Find the running sum of the histogram values)convoluted sum)
2. Normalize the values from step1 by dividing by total number of pixels.
3. Multiply the values from step2 by the maximum gray level value and round.
4. Map the gray-level values to the results from step 3, using a one-to one correspondence.

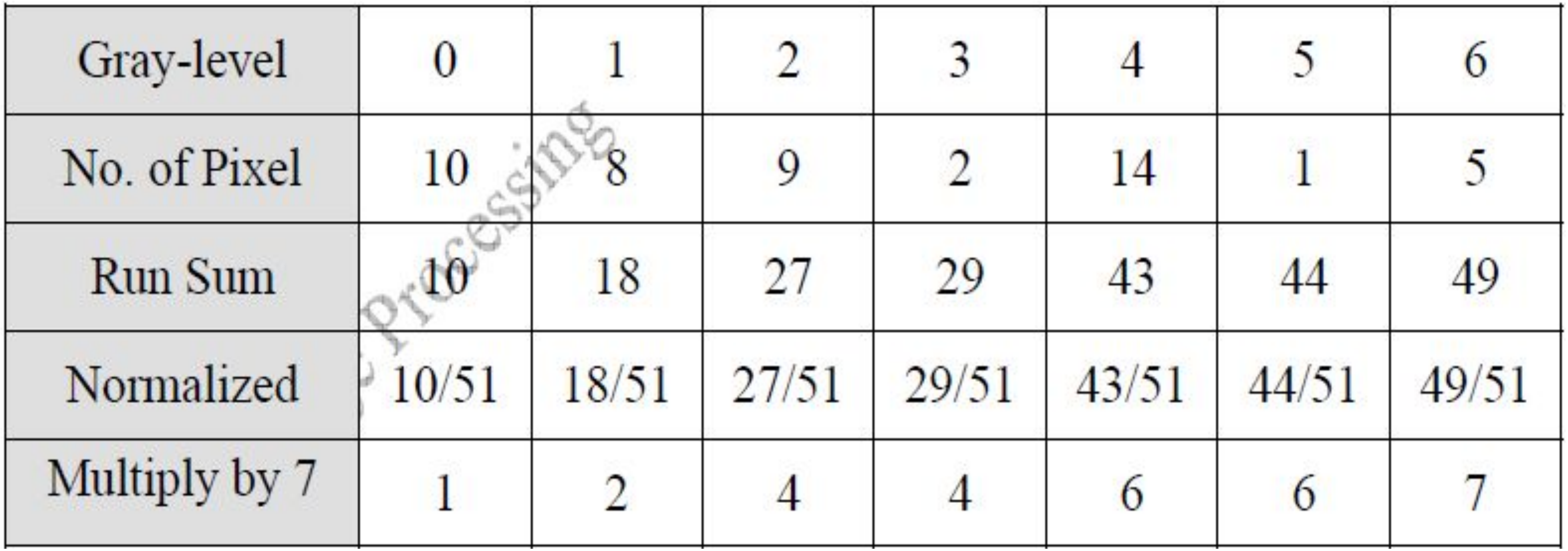
Example:-

- We have an image with 3 bit /pixel, so the possible range of values is 0 to 7.We have an image with the following histogram:



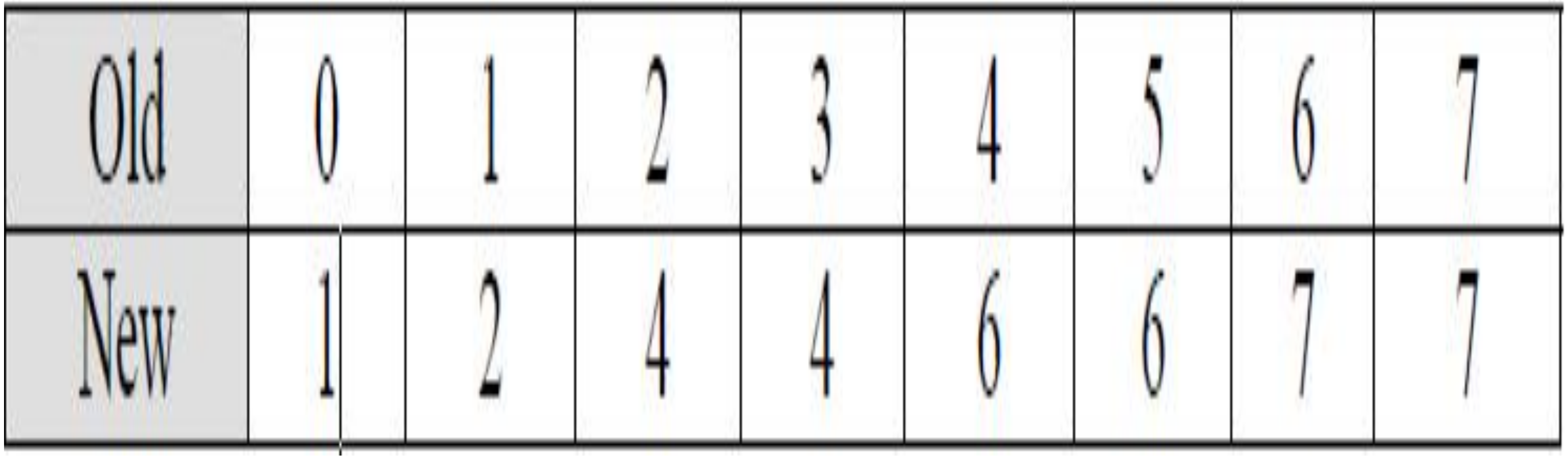
Gray-level value	0	1	2	3	4	5	6	7
No of Pixel Histogram value	10	8	9	2	14	1	5	2

- **The first three steps:**



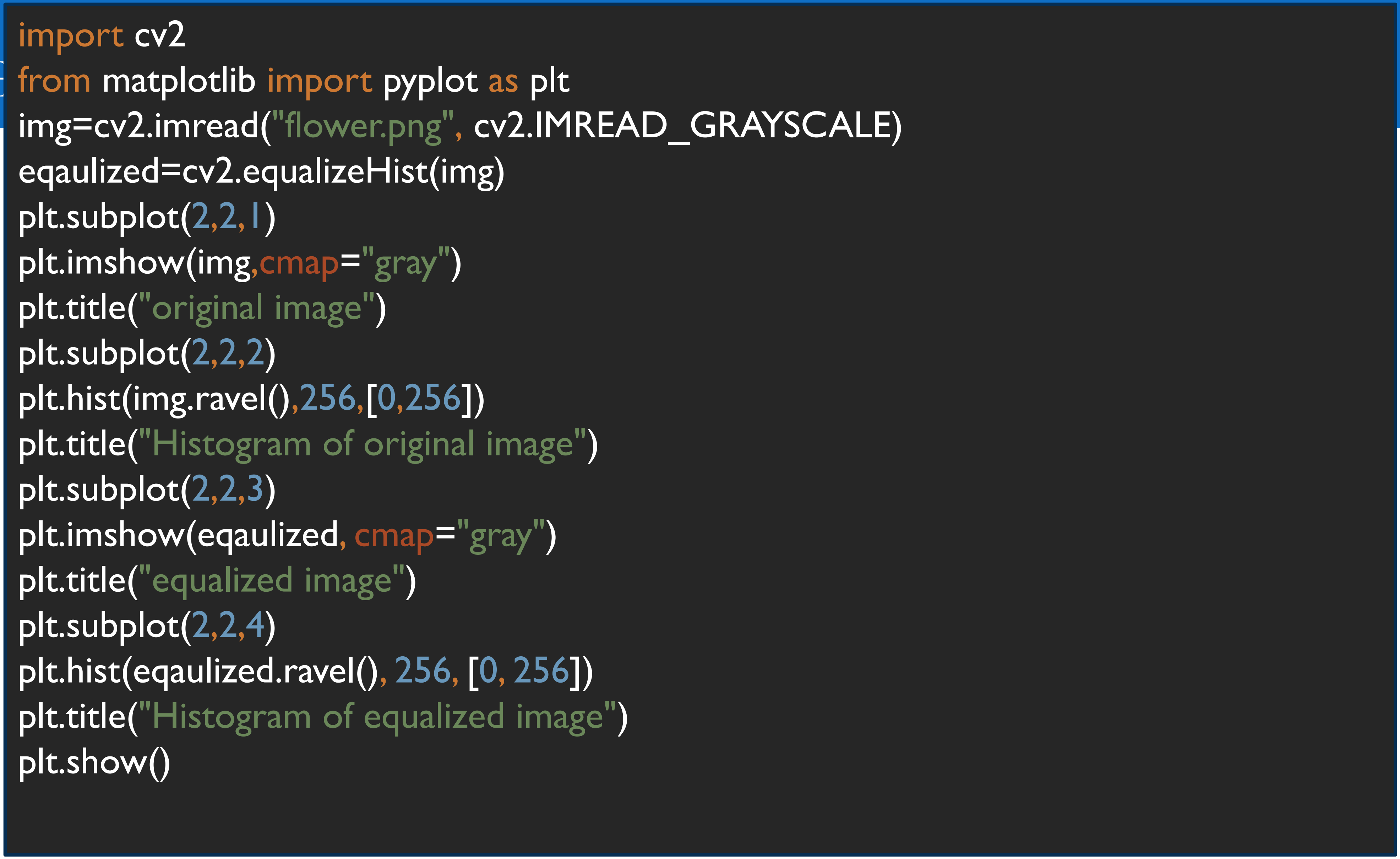
Gray-level	0	1	2	3	4	5	6	7
No. of Pixel	10	8	9	2	14	1	5	2
Run Sum	10	18	27	29	43	44	49	51
Normalized	10/51	18/51	27/51	29/51	43/51	44/51	49/51	51/51
Multiply by 7	1	2	4	4	6	6	7	7

- **The fourth step:**

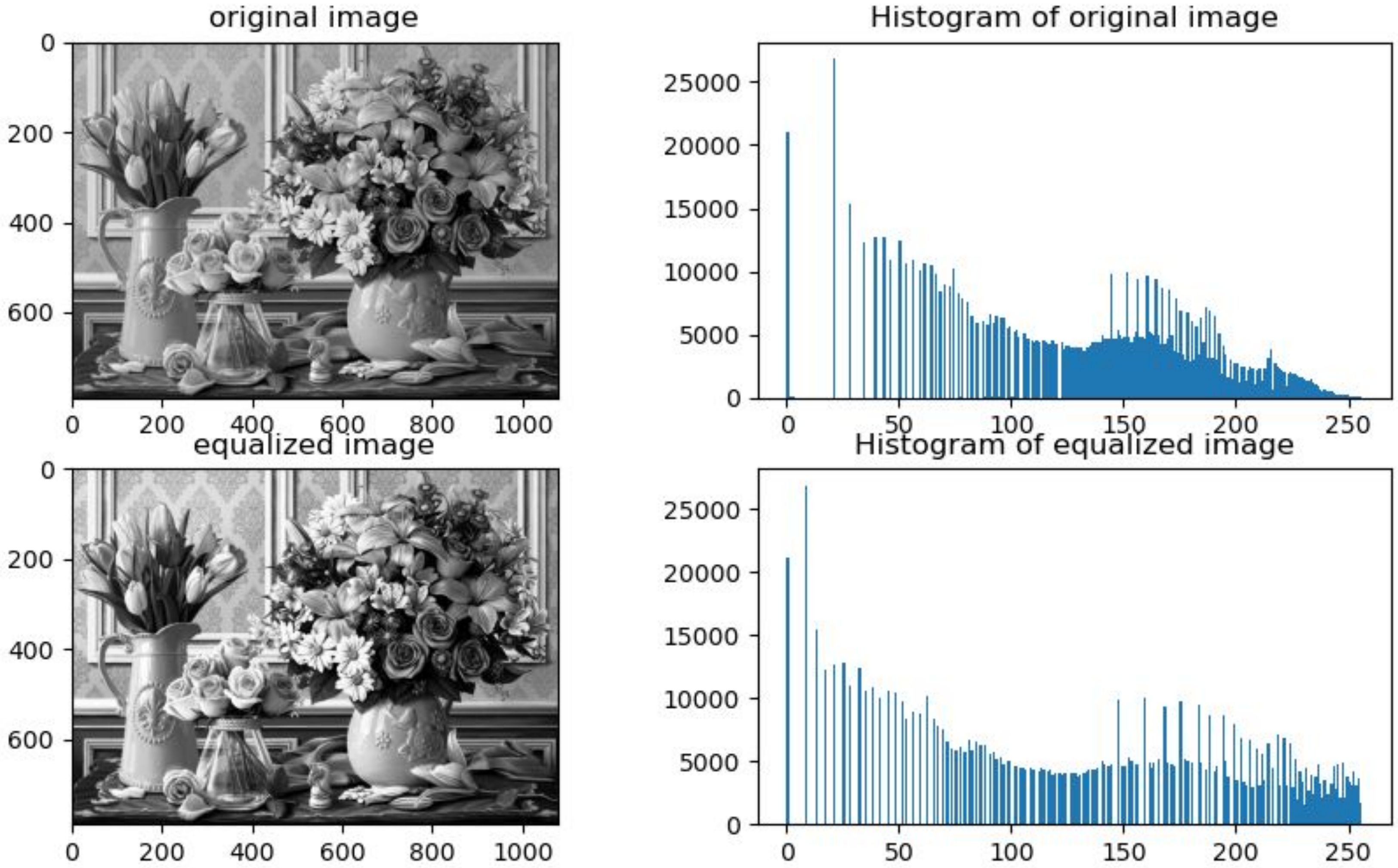


Old	0	1	2	3	4	5	6	7
New	1	2	4	4	6	6	7	7

H



```
import cv2
from matplotlib import pyplot as plt
img=cv2.imread("flower.png", cv2.IMREAD_GRAYSCALE)
eqaulized=cv2.equalizeHist(img)
plt.subplot(2,2,1)
plt.imshow(img,cmap="gray")
plt.title("original image")
plt.subplot(2,2,2)
plt.hist(img.ravel(),256,[0,256])
plt.title("Histogram of original image")
plt.subplot(2,2,3)
plt.imshow(eqaulized, cmap="gray")
plt.title("equalized image")
plt.subplot(2,2,4)
plt.hist(eqaulized.ravel(), 256, [0, 256])
plt.title("Histogram of equalized image")
plt.show()
```


original image
0
200
400
600
0
200
400
600
800
1000
Histogram of original image
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
equalized image
0
200
400
600
0
200
400
600
800
1000
Histogram of equalized image
25000
20000
15000
10000
5000
0
0
50
100
150
200
250

Thank you