Computer Vision

Name: Abhishek Choudhary

Branch: AIML-B1

Lab - 1 : Basic Image processing operations.

Objectives:

The objective of this lab is to introduce the student to OpenCV/python, especially for image processing.

1.Reading an image in python

2.Convert Images to another format

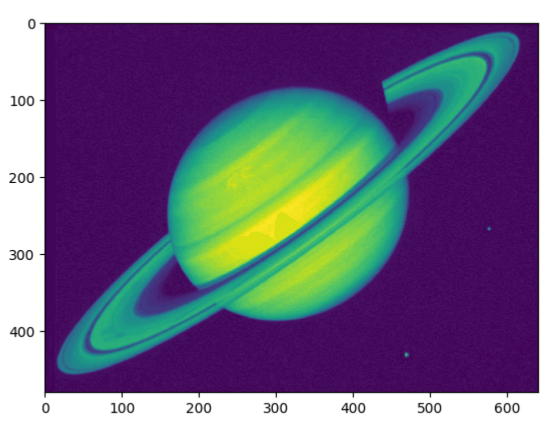
3.Convert an Image to Grayscale

4.Perform Image enhancement operations

1.Reading and displaying an image in **python**

In [30]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
image = mpimg.imread('/content/saturn (1).tif')
plt.imshow(image)
```

Out[30]:
```
<matplotlib.image.AxesImage at 0x79b8592e3950>
```



2.Convert Images to another format and compare them

In [2]:
```
from PIL import Image
import os
img = Image.open('/content/saturn (1).tif')
img.save('saturn (1).tif')
img.save('saturn (1).jpg')
img.save('saturn (1).png')
img.save('saturn (1).jpeg')
img.save('saturn (1).gif')
print("size of file in tiff",round(os.path.getsize('saturn (1).tif')/1024,2),"KB")
print("size of file in jpg",round(os.path.getsize('saturn (1).jpg')/1024,2),"KB")
print("size of file in jpeg",round(os.path.getsize('saturn (1).jpeg')/1024,2),"KB")
print("size of file in png",round(os.path.getsize('saturn (1).png')/1024,2),"KB")
print("size of file in gif",round(os.path.getsize('saturn (1).gif')/1024,2),"KB")
```

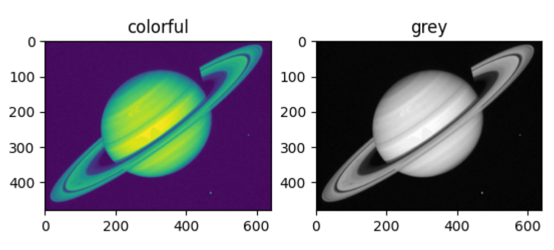
```
size of file in tiff 300.17 KB
size of file in jpg 22.32 KB
size of file in jpeg 22.32 KB
size of file in png 112.16 KB
size of file in gif 113.26 KB
```

3.Convert an Image to Grayscale and display

In [3]:
```
from PIL import Image
img = Image.open('/content/saturn (1).tif').convert('L')
img.save('saturn_gray.tif')
```

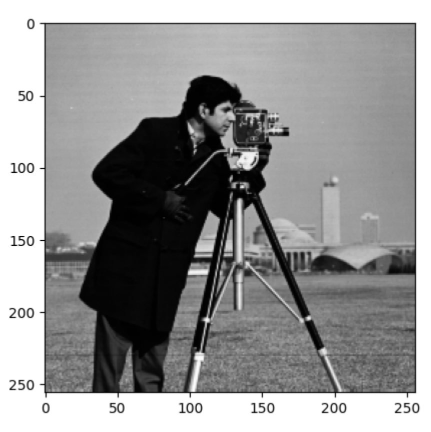
In [4]:
```
plt.subplot(1,2,1)
plt.imshow(mpimg.imread('saturn_gray.tif'))
plt.title("colorful")
plt.subplot(1,2,2)
plt.imshow(img, cmap='gray')
plt.title("grey")
```

Out[4]:
```
Text(0.5, 1.0, 'grey')
```



In [5]:
```
img = mpimg.imread('/content/cameraman (1).tif')
plt.imshow(img, cmap='gray')
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x79b8844fdb10>
```



4.Perform Image enhancement operations

Perform Image Enhancement Operations Image enhancement is a vital step in image processing that focuses on improving the visual quality of an image or converting it into a format that is more suitable for interpretation by humans or analysis by machines. This process becomes essential due to factors like poor lighting, low contrast, noise, or other types of degradation that can negatively affect image clarity.

Common image enhancement techniques include:

Histogram Equalization: Enhances image contrast by redistributing the most frequent intensity values across the entire range. This is particularly helpful for images where both the foreground and background are either too dark or too bright.

Contrast Stretching: Improves contrast by expanding the range of pixel intensity values to cover the full spectrum. It's especially useful for images that suffer from low contrast.

Smoothing: Used to reduce noise and eliminate fine details, smoothing is often applied in the preprocessing stage to suppress high-frequency noise.

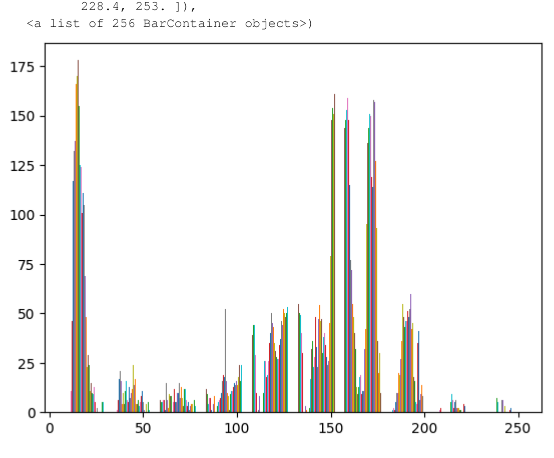
Sharpening: This technique highlights edges and fine details, making important features in the image more pronounced.

Filtering: A variety of filters (e.g., Gaussian, median) can be used to enhance images by reducing noise or emphasizing edges, depending on the desired outcome.

Applying these techniques can significantly enhance image quality, making the images more appropriate for detailed analysis and interpretation.

In [6]:
```
plt.hist(img)
```

Out[6]:
```
(array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  5., ...,  0.,  0.,  0.],
        ...,
        [ 2.,  4.,  6., ...,  0.,  0.,  0.],
        [ 3.,  3., 12., ...,  0.,  0.,  0.],
        [ 5.,  1., 12., ...,  0.,  0.,  0.]]),
 array([  7. ,  31.6,  56.2,  80.8, 105.4, 130. , 154.6, 179.2, 203.8,
        228.4, 253. ]),
 <a list of 256 BarContainer objects>)
```



In [7]:
```
img.shape
```

Out[7]:
```
(256, 256)
```

In [8]:
```
import numpy as np
```

In [9]:
```
def calculate_histogram(image):
    histogram = [0] * 256
    for row in image:
        for pixel in row:
            histogram[pixel] += 1
    return histogram
```

In [10]:
```
hist=calculate_histogram(img)
```

In [11]:
```
print(np.array(hist))
```

```
[   0    0    0    0    0    0    0    4  423 1477 1259 1175 1456 1529
 1685 1338  968  471  261  187  169  140  107  109  104   96   99  114
   86  108   91   76   99   92   83  105   85  100   98   97   81   86
   80  106   68   59   72   77   65   70   61   67   50   64   74   67
   59   70   85   80   68   72   74   86   87   76   67   61   67   62
   43   57   55   43   67   52   53   36   47   52   50   45   64   52
   68   69   64   55   82   92   88   75  110   78  101   86  116  135
  144  145  180  154  179  179  231  203  196  225  245  236  266  266
  276  307  296  285  284  295  296  331  326  340  351  351  360  359
  410  374  391  417  434  417  421  407  412  376  365  406  379  353
  354  357  375  377  404  396  463  524  524  517  601  569  566  547
  536  663  597  668  744  773  900  916 1221 1174 1206 1187 1104  958
  994  924  778  729  704  674  669  632  668  668  690  631  676  706
  635  639  539  545  439  330  232  183  127   75   83   49   50   49
   45   35   28   34   33   19   25   29   18   26   17   22   19   21
   19   11   13   26   24   18   14    8    9    9   14   15   16   19
   14   22   14   13   14   15   14   11    9   17    9   14   17   11
   26   22   22   14   11    6    7    2    4    7    7   12   13    5
    3    1    0    0]
```

In [12]:
```
len(hist)
```

Out[12]:
```
256
```

In [13]:
```
def calculate_cdf(histogram):
    cdf = [0] * len(histogram)
    cdf[0] = histogram[0]
    for i in range(1, len(histogram)):
        cdf[i] = cdf[i - 1] + histogram[i]
    return cdf
```

In [14]:
```
cdf=calculate_cdf(hist)
```

In [15]:
```
print(np.array(cdf))
```

```
[    0     0     0     0     0     0     0     4   427  1904  3163  4338
  5794  7323  9008 10346 11314 11785 12046 12233 12402 12542 12649 12758
 12862 12958 13057 13171 13257 13365 13456 13532 13631 13723 13806 13911
 13996 14096 14194 14291 14372 14458 14538 14644 14712 14771 14843 14920
 14985 15055 15116 15183 15233 15297 15371 15438 15497 15567 15652 15732
 15800 15872 15946 16032 16119 16195 16262 16323 16390 16452 16495 16552
 16607 16650 16717 16769 16822 16858 16905 16957 17007 17052 17116 17168
 17236 17305 17369 17424 17506 17598 17686 17761 17871 17949 18050 18136
 18252 18387 18531 18676 18856 19010 19189 19368 19599 19802 19998 20223
 20468 20704 20970 21236 21512 21819 22115 22400 22684 22979 23275 23606
 23932 24272 24623 24974 25334 25693 26103 26477 26868 27285 27719 28136
 28557 28964 29376 29752 30117 30523 30902 31255 31609 31966 32341 32718
 33122 33518 33981 34505 35029 35546 36147 36716 37282 37829 38365 39028
 39625 40293 41037 41810 42710 43626 44847 46021 47227 48414 49518 50476
 51470 52394 53172 53901 54605 55279 55948 56580 57248 57916 58606 59237
 59913 60619 61254 61893 62432 62977 63416 63746 63978 64161 64288 64363
 64446 64495 64545 64594 64639 64674 64702 64736 64769 64788 64813 64842
 64860 64886 64903 64925 64944 64965 64984 64995 65008 65034 65058 65076
 65090 65098 65107 65116 65130 65145 65161 65180 65194 65216 65230 65243
 65257 65272 65286 65297 65306 65323 65332 65346 65363 65374 65400 65422
 65444 65458 65469 65475 65482 65484 65488 65495 65502 65514 65527 65532
 65535 65536 65536 65536]
```

In [16]:
```
len(cdf)
```

Out[16]:
```
256
```

In [17]:
```
def normalize_cdf(cdf):
    cdf_normalized = [0] * len(cdf)
    min_cdf = min([x for x in cdf if x > 0])
    max_cdf= max([x for x in cdf if x > 0])
    for i in range(len(cdf)):
        cdf_normalized[i] = int(((cdf[i] - min_cdf) / (max_cdf - min_cdf)) * 255)
    return cdf_normalized
```

In [18]:
```
n_cdf=normalize_cdf(cdf)
```

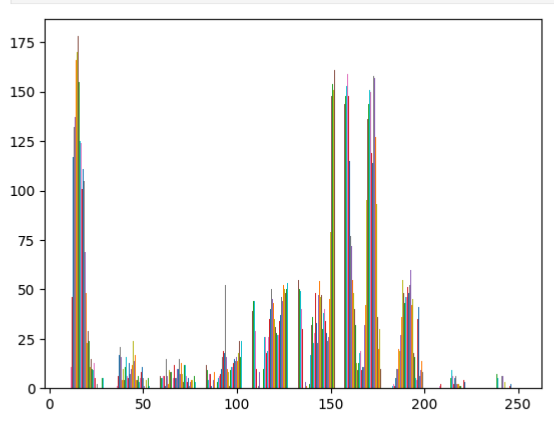
In [19]:
```
print(np.array(n_cdf))
```

```
[  0   0   0   0   0   0   0   0   1   7  12  16  22  28  35  40  44  45
  46  47  48  48  49  49  50  50  50  51  51  51  52  52  53  53  53  54
  54  54  55  55  55  56  56  56  57  57  57  58  58  58  58  59  59  59
  59  60  60  60  60  61  61  61  62  62  62  63  63  63  63  64  64  64
  64  64  65  65  65  65  65  65  66  66  66  66  67  67  67  67  68  68
  68  69  69  69  70  70  71  71  72  72  73  73  74  75  76  77  77  78
  79  80  81  82  83  84  86  87  88  89  90  91  93  94  95  97  98  99
 101 103 104 106 107 109 111 112 114 115 117 118 120 121 122 124 125 127
 128 130 132 134 136 138 140 142 145 147 149 151 154 156 159 162 166 169
 174 179 183 188 192 196 200 203 206 209 212 215 217 220 222 225 228 230
 233 235 238 240 242 245 246 248 248 249 250 250 250 250 251 251 251 251
 251 251 252 252 252 252 252 252 252 252 252 252 252 252 252 253 253 253
 253 253 253 253 253 253 253 253 253 253 253 253 253 254 254 254 254
 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254
 254 255 255 255]
```

In [20]:
```
def apply_histogram_equalization(image, cdf_normalized):
    equalized_image = []
    for row in image:
        new_row = []
        for pixel in row:
            new_row.append(cdf_normalized[pixel])
        equalized_image.append(new_row)
    return equalized_image
```

In [21]:
```
final_img=apply_histogram_equalization(img,n_cdf)
```

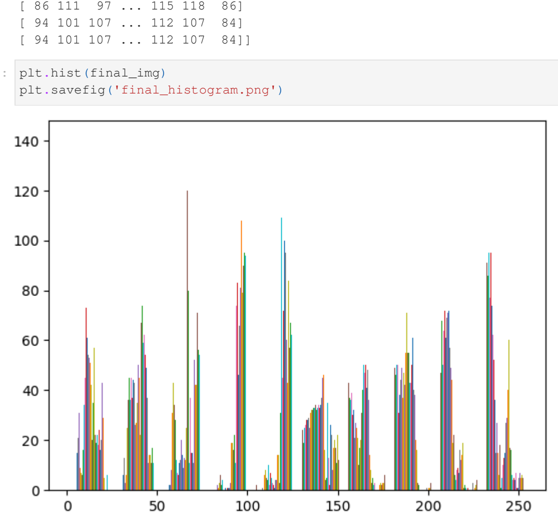
In [22]:
```
plt.hist(img)
plt.savefig('histogram_plot.png')
```



In [23]:
```
print(np.array(final_img))
```

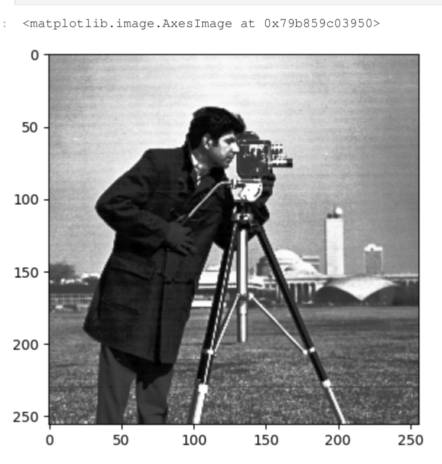
```
[[154 162 159 ... 142 145 145]
 [166 149 156 ... 149 151 147]
 [154 162 159 ... 142 145 145]
 ...
 [ 86 111  97 ... 115 118  86]
 [ 94 101 107 ... 112 107  84]
 [ 94 101 107 ... 112 107  84]]
```

In [24]:
```
plt.hist(final_img)
plt.savefig('final_histogram.png')
```



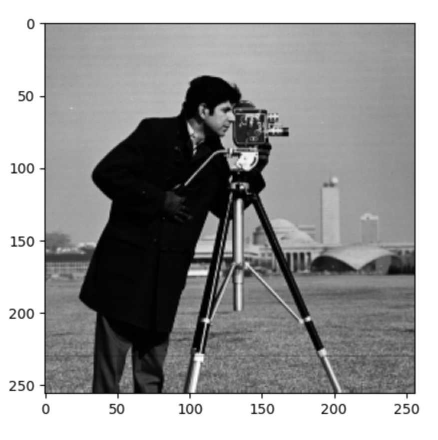
In [25]:
```
plt.imshow(final_img, cmap='gray')
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x79b859c03950>
```



In [26]:
```
plt.imshow(img, cmap='gray')
```

Out[26]:
```
<matplotlib.image.AxesImage at 0x79b86c5c7650>
```



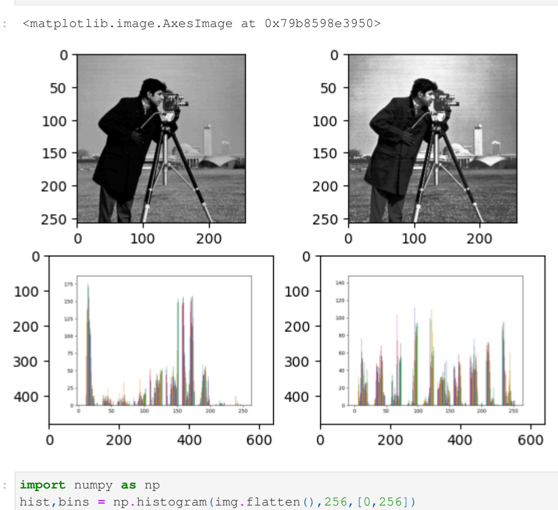
In [27]:
```
plt.subplot(2,2,1)
plt.imshow(img, cmap='gray')

plt.subplot(2,2,2)
plt.imshow(final_img, cmap='gray')

plt.subplot(2,2,3)
a=plt.imread('histogram_plot.png')
plt.imshow(a)

plt.subplot(2,2,4)
b=plt.imread('final_histogram.png')
plt.imshow(b)
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x79b8598e3950>
```



In [28]:
```
import numpy as np
hist,bins = np.histogram(img.flatten(),256,[0,256])
```

In [29]:
```
hist
```

Out[29]:
```
array([   0,    0,    0,    0,    0,    0,    0,    4,  423, 1477, 1259,
       1175, 1456, 1529, 1685, 1338,  968,  471,  261,  187,  169,  140,
        107,  109,  104,   96,   99,  114,   86,  108,   91,   76,   99,
         92,   83,  105,   85,  100,   98,   97,   81,   86,   80,  106,
         68,   59,   72,   77,   65,   70,   61,   67,   50,   64,   74,
         67,   59,   70,   85,   80,   68,   72,   74,   86,   87,   76,
         67,   61,   67,   62,   43,   57,   55,   43,   67,   52,   53,
         36,   47,   52,   50,   45,   64,   52,   68,   69,   64,   55,
         82,   92,   88,   75,  110,   78,  101,   86,  116,  135,  144,
        145,  180,  154,  179,  179,  231,  203,  196,  225,  245,  236,
        266,  266,  276,  307,  296,  285,  284,  295,  296,  331,  326,
        340,  351,  351,  360,  359,  410,  374,  391,  417,  434,  417,
        421,  407,  412,  376,  365,  406,  379,  353,  354,  357,  375,
        377,  404,  396,  463,  524,  524,  517,  601,  569,  566,  547,
        536,  663,  597,  668,  744,  773,  900,  916, 1221, 1174, 1206,
       1187, 1104,  958,  994,  924,  778,  729,  704,  674,  669,  632,
        668,  668,  690,  631,  676,  706,  635,  639,  539,  545,  439,
        330,  232,  183,  127,   75,   83,   49,   50,   49,   45,   35,
         28,   34,   33,   19,   25,   29,   18,   26,   17,   22,   19,
         21,   19,   11,   13,   26,   24,   18,   14,    8,    9,    9,
         14,   15,   16,   19,   14,   22,   14,   13,   14,   15,   14,
         11,    9,   17,    9,   14,   17,   11,   26,   22,   22,   14,
         11,    6,    7,    2,    4,    7,    7,   12,   13,    5,    3,
          1,    0,    0])
```

Conclusion

In this notebook, we explored various fundamental image processing operations using Python and popular libraries such as OpenCV, PIL, and Matplotlib. We started by reading and displaying images using matplotlib.pyplot and matplotlib.image . We then converted images to different formats and compared their file sizes to understand the impact of different formats on storage. Next, we converted an image to grayscale and displayed both the original and grayscale images using subplots for comparison. We performed image enhancement operations, including histogram equalization, to improve the visual quality of the images. We calculated the cumulative distribution function (CDF) and applied it to enhance the image contrast. Throughout this notebook, we learned how to manipulate and enhance images, understand the importance of different image formats, and apply various image processing techniques to improve image quality. These skills are essential for any computer vision task and provide a solid foundation for more advanced image processing and analysis. By implementing these operations, we gained hands-on experience with image processing, which is crucial for practical applications in computer vision and related fields.