Name: Krish Sukhani

UID: 2018140059

Batch: D

Branch: IT

Equalization

In [1]:
```
from PIL import ImageFilter
from PIL import Image
import math
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
ent/drive", force_remount=True).
```

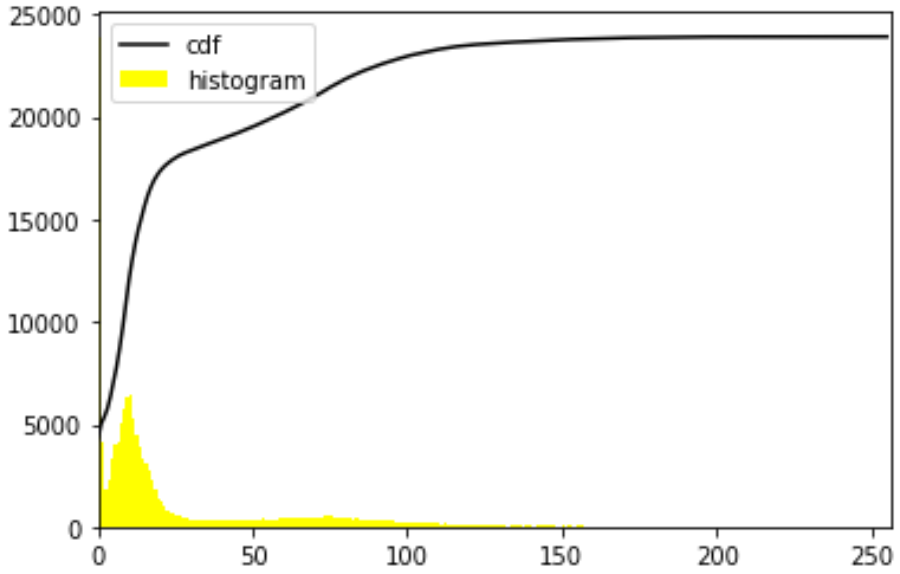
In [3]:
```
img_path = '/content/drive/MyDrive/Sem-7/DIP-Lab/Equalization/dip_3.jpg'
```

In [4]:
```
img = cv2.imread(img_path)
```

In [5]:
```
plt.imshow(img)
```

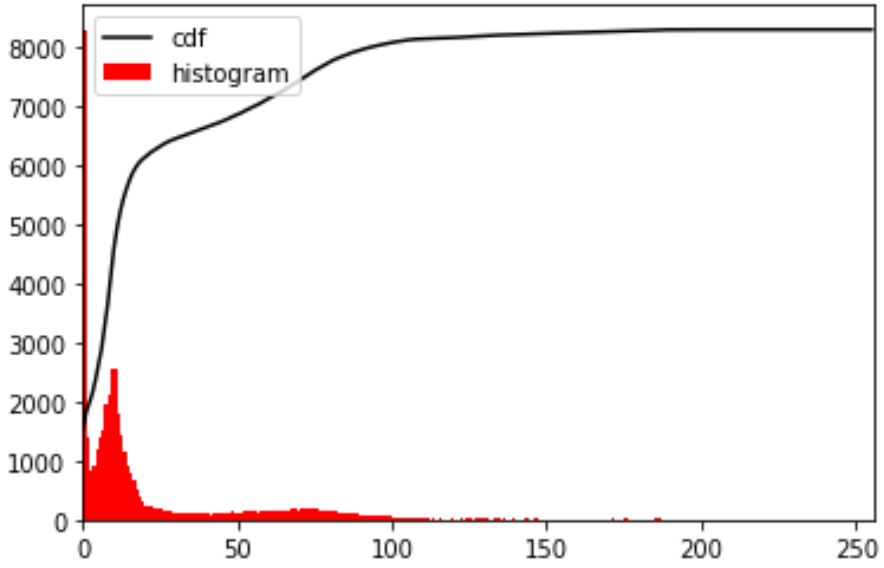
Out[5]: <matplotlib.image.AxesImage at 0x7ffa2b8b00d0>

In [6]:
```
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(img.flatten(),256,[0,256], color = 'yellow')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

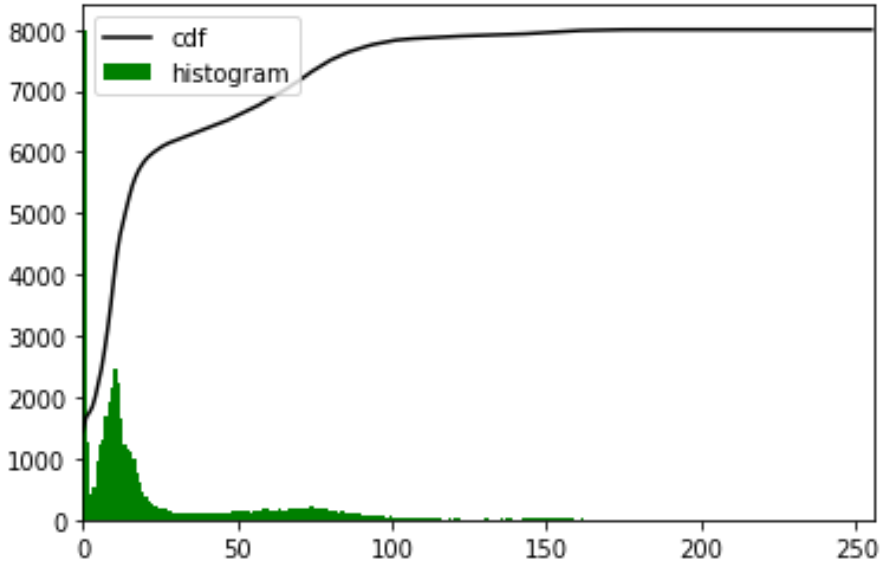


In [7]:
```
R, G, B = cv2.split(img)
```

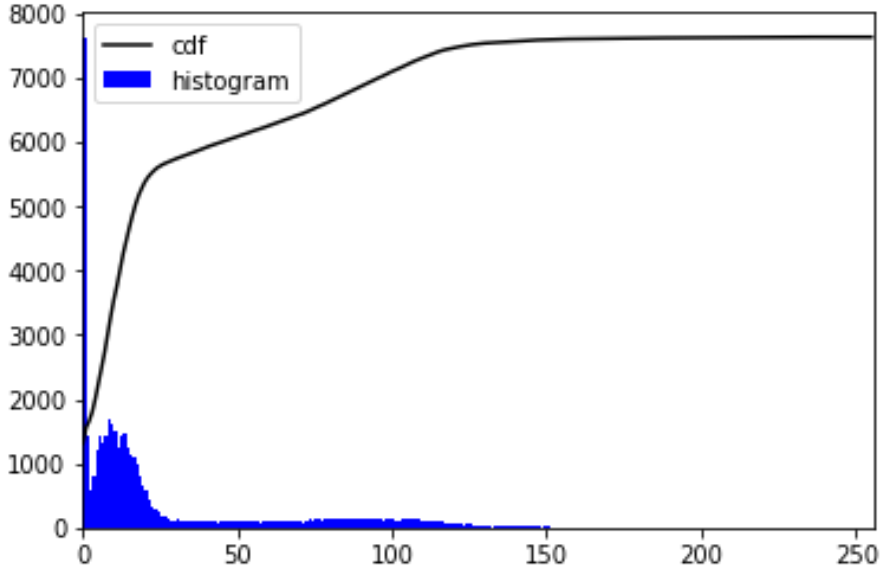
In [8]:
```
hist,bins = np.histogram(R.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(R.flatten(),256,[0,256], color = 'red')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [9]:
```
hist,bins = np.histogram(G.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(G.flatten(),256,[0,256], color = 'green')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



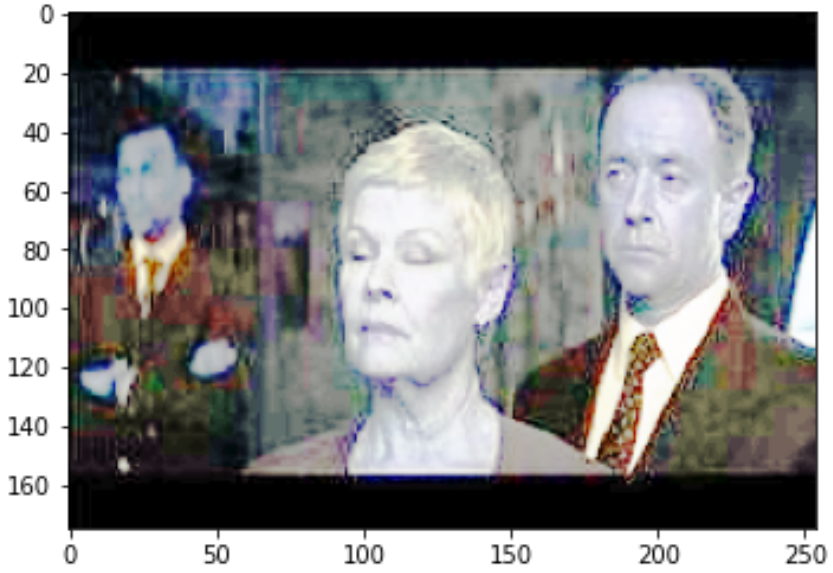
In [10]:
```
hist,bins = np.histogram(B.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(B.flatten(),256,[0,256], color = 'blue')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



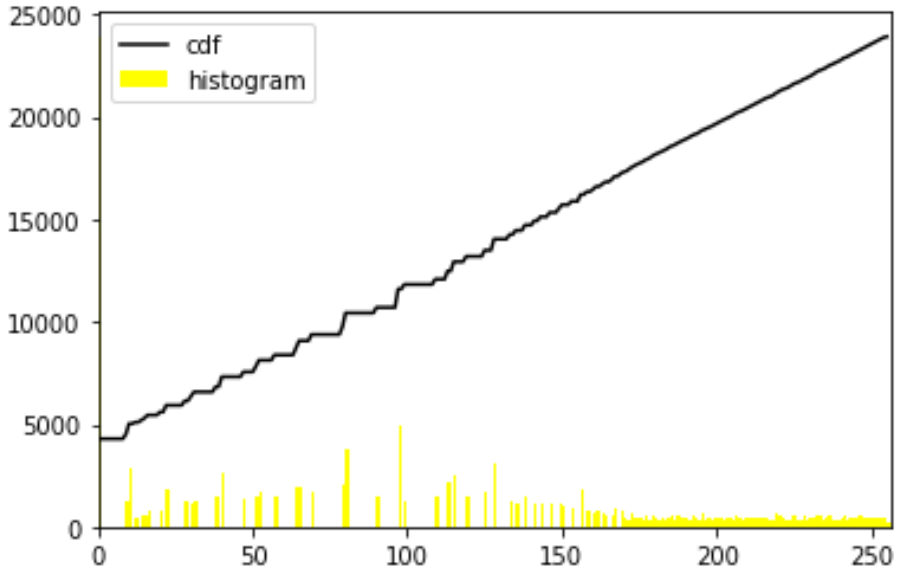
In [11]:
```
output1_R = cv2.equalizeHist(R)
output1_G = cv2.equalizeHist(G)
output1_B = cv2.equalizeHist(B)

equ = cv2.merge((output1_R, output1_G, output1_B))
```

In [12]:
```
plt.imshow(equ)
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x7ffa1bd9e5d0>
```



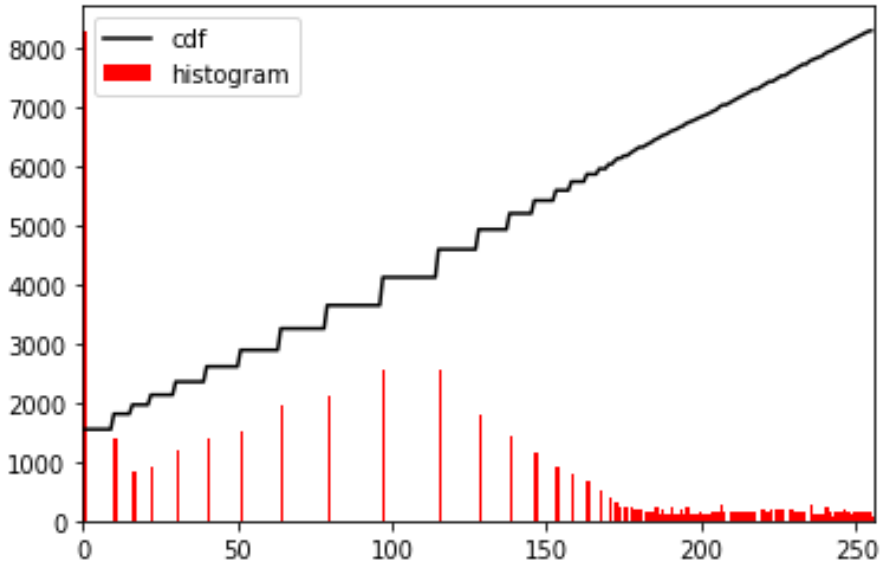
In [13]:
```
hist,bins = np.histogram(equ.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(equ.flatten(),256,[0,256], color = 'yellow')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



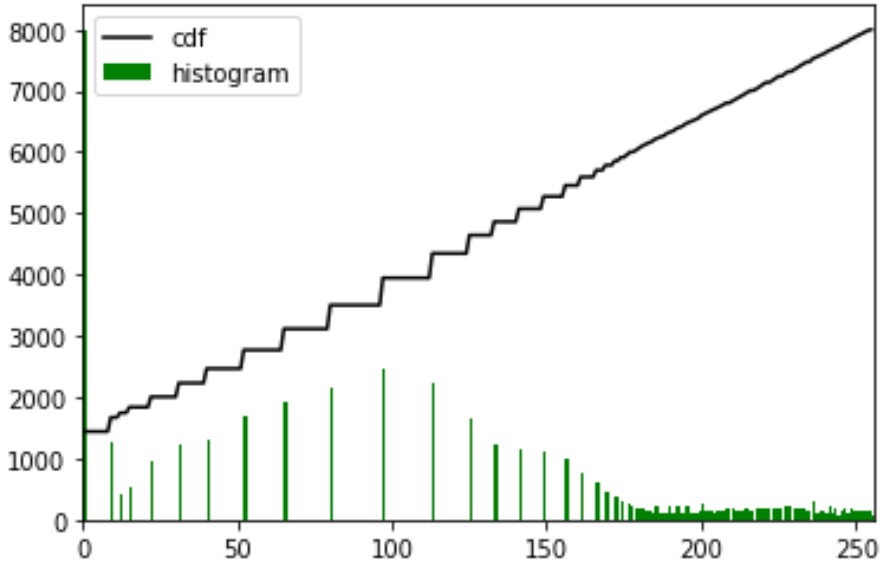
Observation : The grey levels of the output historgram are distributed equally as compared to the original histogram

In [14]:

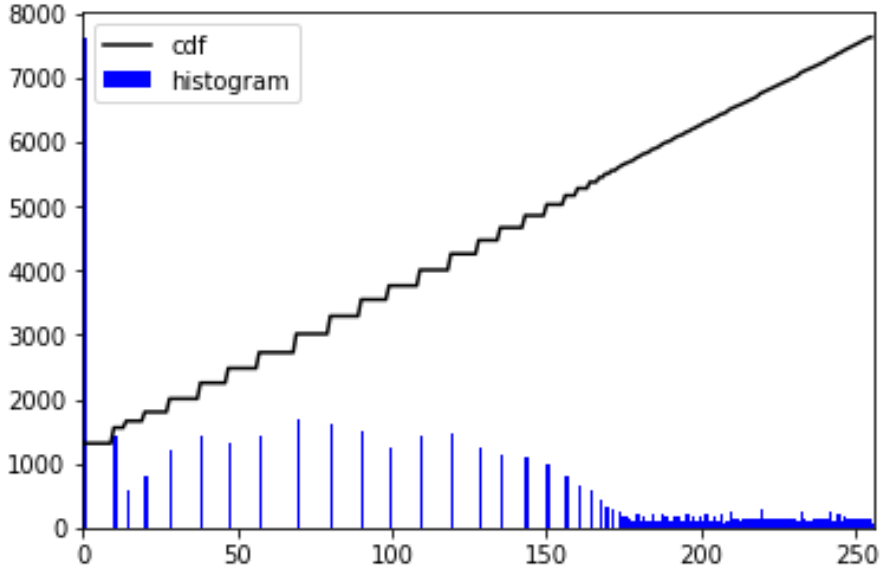
```
hist,bins = np.histogram(output1_R.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(output1_R.flatten(),256,[0,256], color = 'red')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [15]:
```
hist,bins = np.histogram(output1_G.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(output1_G.flatten(),256,[0,256], color = 'green')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [16]:
```
hist,bins = np.histogram(output1_B.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'black')
plt.hist(output1_B.flatten(),256,[0,256], color = 'blue')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



Mathematical implementation

In [17]:
```
n = np.array([790,1023,850,656,329,245,122,81])
```

In [18]:
```
N = sum(n)
```

In [19]:
```
N
```

Out[19]:
```
4096
```

In [20]:
```
pdf = n/N
```

In [21]:
```
pdf
```

Out[21]:
```
array([0.19287109, 0.24975586, 0.20751953, 0.16015625, 0.08032227,
       0.05981445, 0.02978516, 0.01977539])
```

In [22]:
```
cdf = np.zeros(len(pdf))
```

In [23]:
```
cdf
```

Out[23]:
```
array([0., 0., 0., 0., 0., 0., 0., 0.])
```

In [24]:
```
cdf[0] = pdf[0]
```

In [25]:
```
cdf
```

Out[25]:
```
array([0.19287109, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        ])
```

In [26]:
```
for i in range(1,len(pdf)):
  cdf[i] = cdf[i-1] + pdf[i]
```

In [27]:
```
cdf
```

Out[27]:
```
array([0.19287109, 0.44262695, 0.65014648, 0.81030273, 0.890625  ,
       0.95043945, 0.98022461, 1.        ])
```

In [28]:
```
scale_values = (len(n) - 1)*cdf
```

In [29]:
```
scale_values
```

Out[29]:
```
array([1.35009766, 3.09838867, 4.55102539, 5.67211914, 6.234375  ,
       6.65307617, 6.86157227, 7.        ])
```

In [30]:
```
final_value = list()
```

In [31]:
```
for i in range(0,len(scale_values)):
  final_value.append(round(scale_values[i]))
```

In [32]:
```
final_value
```

Out[32]:
```
[1, 3, 5, 6, 6, 7, 7, 7]
```

In [33]:
```
old_grey = list()
```

In [34]:
```
for i in range(0,len(n)):
  old_grey.append(i)
```

In [35]:
```
new_grey = np.zeros(len(n))
```

In [36]:
```
for i in range(0,len(final_value)):
  new_grey[final_value[i]] += n[i]
```

In [37]:
```
new_grey
```

Out[37]:
```
array([   0.,  790.,    0., 1023.,    0.,  850.,  985.,  448.])
```

Conclusion: Performed mathemetical operation to understand the math behind equilization technique. Used the library to check the output histogram for RGB values as well as the images