**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Image Processing**

**Lab Number**

**4**

**SUBMITTED TO:**

**Dr. Asad Khan**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Filtering on Images and Histogram Calculations

**Related Topic/Chapter in theory class:**

Spatial Filtering

**Hardware/Software required:**

Hardware: PC

Software Tool: Pycharm

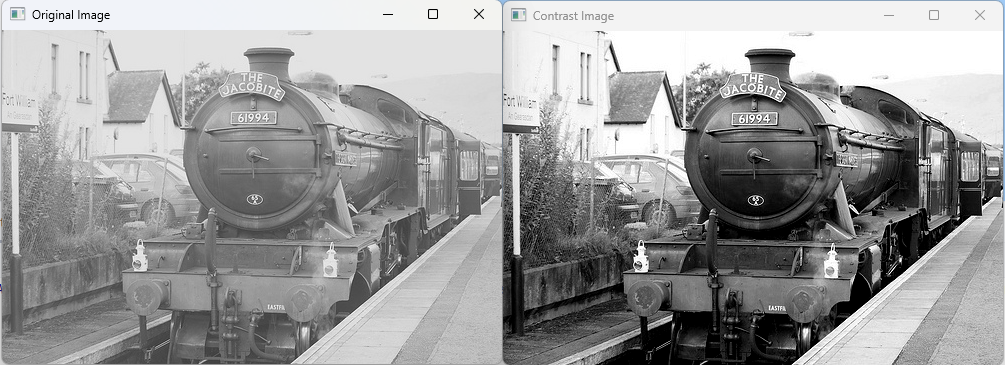
**Task 1:**

**Apply contrast stretching on the image provided by setting 5th and 95th percentiles of the input values to 0 and 255 respectively. The remaining values will be stretched as: *N*𝑒𝑤 𝑉𝑎𝑙𝑢𝑒 𝑓𝑜𝑟 𝑃𝑖𝑥𝑒𝑙 = 255 ∗ (𝐶𝑢𝑟𝑟𝑒𝑛𝑡 𝑣𝑎𝑙𝑢𝑒 𝑓𝑜𝑟 𝑃𝑖𝑥𝑒𝑙 − 5𝑡ℎ 𝑃𝑒𝑟𝑐𝑒𝑛𝑡𝑖𝑙𝑒) / (95𝑡ℎ 𝑃𝑒𝑟𝑐𝑒𝑛𝑡𝑖𝑙𝑒 − 5𝑡ℎ 𝑃𝑒𝑟𝑐𝑒𝑛𝑡𝑖𝑙𝑒)**

**Solution:**

import cv2 as cv  
import numpy as np  
  
def contrast\_stretch(image):  
 im\_min\_5 = np.percentile(image, 5)  
 im\_max\_95 = np.percentile(image, 95)  
 rows,cols = image.shape  
 new\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 for i in range(rows):  
 for j in range(cols):  
 if(image[i][j] < im\_min\_5):  
 new\_img[i][j] = 0  
 elif(image[i][j] > im\_max\_95):  
 new\_img[i][j] = 255  
 else:  
 new\_img[i][j] = 255 \* ((image[i][j] - im\_min\_5) / (im\_max\_95 - im\_min\_5))  
  
 return new\_img  
  
  
*#---------------------------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 4/Lab 4/low\_con.jpg", 0)  
cont\_img = contrast\_stretch(image)  
  
cv.imshow('Original Image', image)  
cv.imshow('Contrast Image', cont\_img)  
cv.waitKey()

**Output:**



**Task 2:**

**Perform the following steps to do Hist. equalization to enhance the grayscale image.**

* **Calculate the histogram of the image and display it using the appropriate command.**

**(Don’t use the built in function of OpenCV or Numpy or Matplotlib etc.)**

* **Calculate probability density function (PDF) from the histogram and display it using the appropriate command PDF = H/(R\*C). Where H is the Histogram and R and C is the number of Rows and Columns of the image respectively.**
* **Calculate cumulative PDF and display it using the appropriate command.**
* **Multiply the Cumulative PDF with 255 to find the transformation function then display it too using the appropriate command.**
* **From the transformation function, replace the gray levels of the image to create contrast enhanced (histogram equalized) image.**
* **Display the enhanced image.**

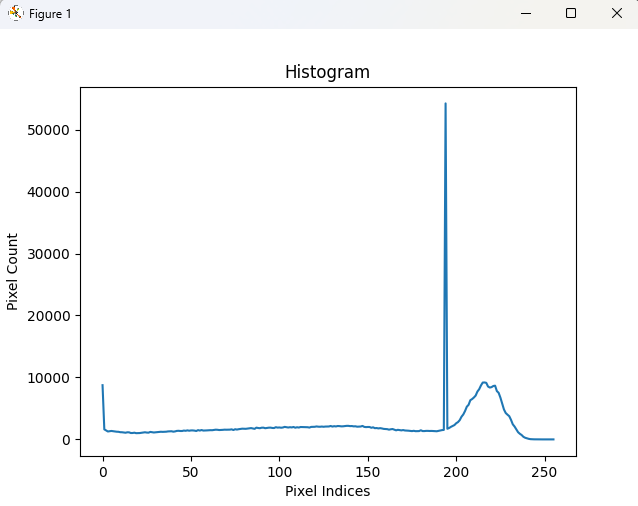
**Solution**

import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
  
def histogram\_creating(image):  
 rows, cols = image.shape  
 histogram = np.zeros(256, dtype = int)  
  
 for i in range(rows):  
 for j in range(cols):  
 val = image[i][j]  
 histogram[val] += 1  
  
 return histogram  
  
def calc\_pdf(histogram, image):  
 rows, cols = image.shape  
  
 histogram = histogram/(rows\*cols)  
  
 return histogram  
  
def calc\_cum\_pdf(pdf):  
 cum\_pdf = np.zeros(len(pdf), dtype = float)  
 cum\_pdf[0] = pdf[0]  
 for i in range(1, len(pdf)):  
 cum\_pdf[i] = cum\_pdf[i-1] + pdf[i]  
  
 return cum\_pdf  
  
def transformation\_fun(cum\_pdf):  
 return np.uint8((cum\_pdf\*255))  
  
def apply\_trans(image, trans\_fun):  
 rows, cols = image.shape  
 new\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 for i in range(rows):  
 for j in range(cols):  
 new\_img[i][j] = trans\_fun[image[i][j]]  
  
 return new\_img  
  
*# ---------------------------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 4/Lab 4/fig05.tif", 0)  
cv.imshow('Original Image', image)  
cv.waitKey()  
histogram = histogram\_creating(image)  
  
plt.plot(histogram)  
plt.xlabel('Pixel Indices')  
plt.ylabel('Pixel Count')  
plt.title('Histogram')  
plt.show()  
  
pdf = calc\_pdf(histogram, image)  
plt.plot(pdf)  
plt.xlabel('Pixel Indices')  
plt.ylabel('Pixel Intensity')  
plt.title('Probability Density Function')  
plt.show()  
  
cum\_pdf = calc\_cum\_pdf(pdf)  
plt.plot(cum\_pdf)  
plt.xlabel('Pixel Indices')  
plt.ylabel('Pixel Intensity')  
plt.title('Cumulative PDF')  
plt.show()  
  
transformation\_function = transformation\_fun(cum\_pdf)  
plt.plot(transformation\_function)  
plt.xlabel('Pixel Indices')  
plt.ylabel('Pixel Count')  
plt.title('Transformation Function')  
plt.show()  
  
print(transformation\_function)  
  
equilized\_img = apply\_trans(image, transformation\_function)  
cv.imshow('Enhanced Image', equilized\_img)  
cv.waitKey()

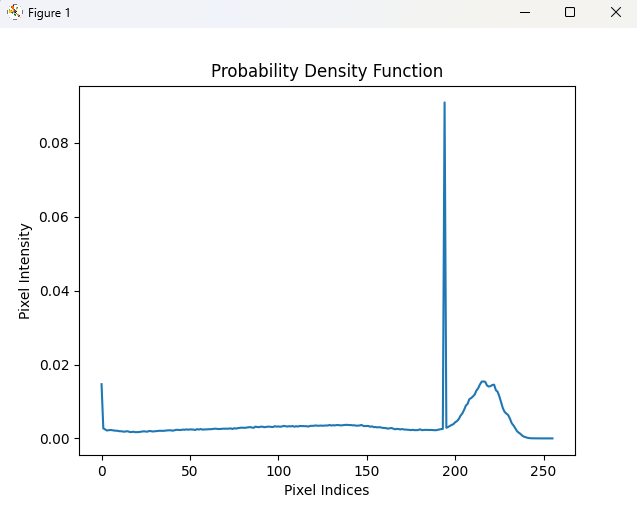
**Output:**



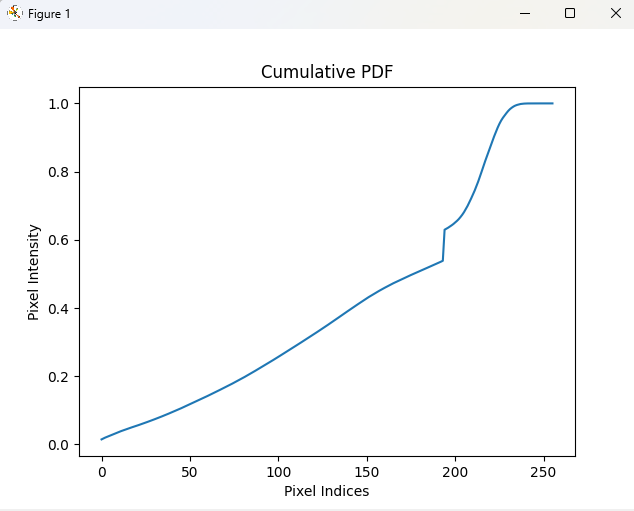
**Original Image**



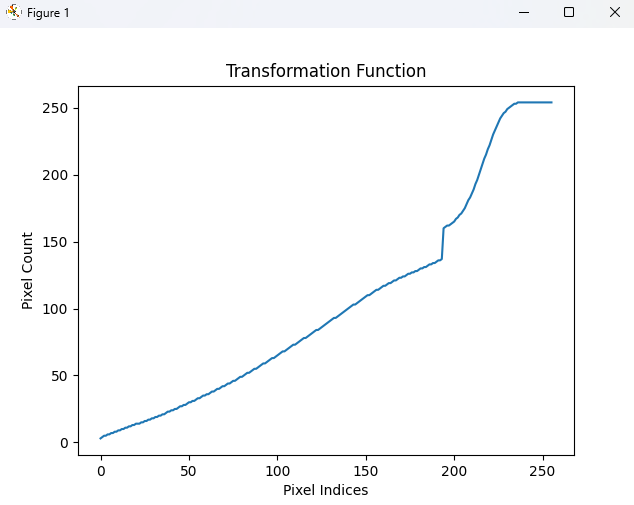
**Histogram**



**Probability Density Function**



**Cumulative PDF**



**Transformation Function**



**Low Contrast Image**

**Task 3:**

**Write a generic function that takes a mask of a specific size from and applies it to an image.**

**Create the following masks and apply it to an image and display the results. The fiter will**

**nxn size. You can use fig05.tif for this task.**

**Solution**

import cv2 as cv  
import numpy as np  
  
def padding(pad, orig):  
 rows, cols = orig.shape  
 padded\_arr = np.ones((rows+ 2 \* pad, cols+ 2 \* pad), dtype = np.uint8)\*255  
  
 for i in range(rows):  
 for j in range(cols):  
 padded\_arr[i+pad][j+pad] = orig[i][j]  
  
 return padded\_arr  
  
def remove\_padding(padded\_img, pad):  
 rows, cols = padded\_img.shape  
 return padded\_img[pad:rows-pad, pad:cols-pad]  
  
def filter(image, filter\_size, filter\_var):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 filtered\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 padded\_img = padding(pad, filtered\_img)  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = image[i-pad:i+pad+1, j-pad:j+pad+1]  
 sub\_img = np.multiply(sub\_img, filter\_var)  
 sub\_img\_val = np.sum(sub\_img)  
  
 padded\_img[i][j] = sub\_img\_val  
  
 filtered\_img = remove\_padding(padded\_img, pad)  
  
 return filtered\_img  
  
*# ----------------------------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 4/Lab 4/fig05.tif", 0)  
cv.imshow('Original Image', image)  
cv.waitKey()  
  
filtered = filter(image, 3, 1/9)  
cv.imshow('Filtered Image', filtered)  
cv.waitKey()

**Output:**

****

**Original Image**



**Log Transform**