Ritvik Choudhary
1W17BG16-0

Image Engineering Fundamentals

Final Report

2020 Spring

Problem 1

For this problem, I made use of the OpenCV library in Python for carrying out the necessary normalization and enhancements.

Before we continue into each of the solutions I would like to share and explain my general code for performing the normalized cumulative histogram processing.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import sys

inputf = sys.argv[1]
outputf = sys.argv[2]

# reading the greyscale image through OpenCV
input_1 = cv2.imread(inputf,0)

print("Input1: Gray level values in pixels")
for i in range (input_1.shape[0]): #traverses through height of the image
    for j in range (input_1.shape[1]): #traverses through width of the image
        print(input_1[i][j], end="\t")
    print("\t")

# calculating the cumulative distribution function (cdf)
def cdf_calc(image):
    # generating the histogram by flattening the image through NumPy
    hist,bins = np.histogram(image.flatten(),256,[0,256])

    # Cumulative Distribution function of the Histogram
    cdf = hist.cumsum()

    # Calculating the normalized cumulative histogram
    cdf_normalized = cdf * float(hist.max()) / cdf.max()

    return cdf, cdf_normalized

# cdf and normalized cdf for the input image
cdf, cdf_normalized = cdf_calc(input_1)

# plotting the histogram of the input image
plt.subplot(121)
plt.title("Input Image 1")
plt.plot(cdf_normalized, color = 'b')
plt.hist(input_1.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper right')

# normalized cumulative histogram (cdf) is used as a mapping function
# utilized the mapping arrays of NumPy for this purpose
cdf_m = np.ma.masked_equal(cdf,0)
cdf_m = (cdf_m - cdf_m.min())*255/(cdf_m.max()-cdf_m.min())
cdf = np.ma.filled(cdf_m,0).astype('uint8')
```

Normalized Cumulative Histogram Processing (1/2)

```
# output image gray values are calculated putting the input image inside the new cdf
output_1 = cdf[input_1]
# the new image is written out as result_1.png
cv2.imwrite(outputf, output_1)

# calculating the cdf and cdf_normalized values for plotting the histogram of output image
cdf1, cdf_normalized1 = cdf_calc(output_1)

print("Output1: Gray level values in pixels")
for i in range (output_1.shape[0]): #traverses through height of the image
    for j in range (output_1.shape[1]): #traverses through width of the image
        print(output_1[i][j], end="\t")
    print("\t")

# plotting the histogram of the output image
plt.subplot(122)
plt.title("Output Image 1")
plt.plot(cdf_normalized1, color = 'b')
plt.hist(output_1.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper right')

# printing the final graphs
plt.show()
```

Normalized Cumulative Histogram Processing (2/2)

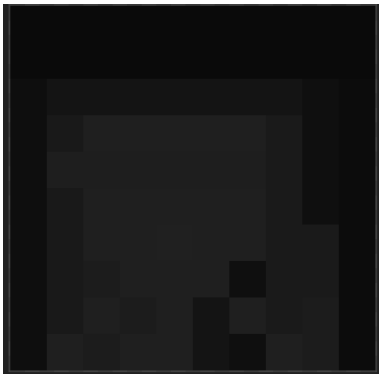
I utilized the above libraries and Numpy for carrying out my calculations, the relevant steps are explained with comments.

(1) Result of contrast enhancement is good.

For this example I took the following as the our input image. I generated this 10x10 image with a varying gray-level range as follows.

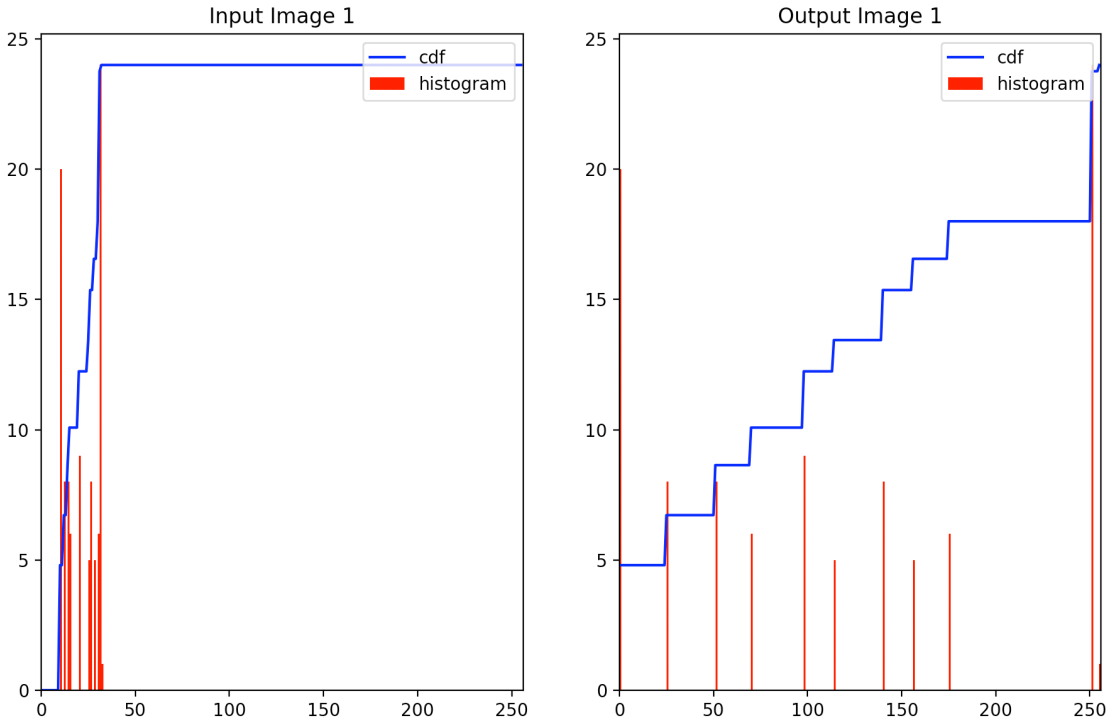
```
Input1: Gray level values in pixels
10      10      10      10      10      10      10      10      10      10
10      10      10      10      10      10      10      10      10      10
14      20      20      20      20      20      20      20      15      12
14      25      31      31      31      31      31      26      15      12
14      30      30      30      30      30      30      26      15      12
14      25      31      31      31      31      31      26      15      12
14      25      31      31      32      31      31      26      26      12
14      25      28      31      31      31      15      26      26      12
14      25      31      28      31      20      31      26      28      12
14      31      28      31      31      20      15      31      28      12
```

Gray level values per pixel for the 10x10 image



Input 10x10 image

After passing the image through the above program, the following gray-level and cumulative gray-level histograms were generated.



Cumulative gray-level and gray-level histograms for the input and output image.

Finally, after performing the contrast enhancement the resulting output image and its pixel detail is as follows.

```
Output1: Gray level values in pixels
0       0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0       0
51      98      98      98      98      98      98      98      70      25
51      114     251     251     251     251     251     140     70      25
51      175     175     175     175     175     175     140     70      25
51      114     251     251     251     251     251     140     70      25
51      114     251     251     255     251     251     140     140     25
51      114     156     251     251     251     70      140     140     25
51      114     251     156     251     98      251     140     156     25
51      251     156     251     251     98      70      251     156     25
```

Gray level values per pixel for the 10x10 enhanced output image

Output 10x10 image after enhancement

As observed in the source code attached above, for the histogram equalization the following methods were utilized.

The cumulative distribution function, cdf was first calculated as follows: (which is also the image's accumulated normalized histogram).

$$cdf_x(i) = \sum_{j=0}^{i} p_x(x = j),$$

Then using a masking function, cdf_m, in the following form, we then replace it back in the cdf to generate the output image.

$$cdf_m = \frac{cdf_m - \min(cdf_m)}{\max cdf_m - \min cdf_m} * 255$$

Results: In this scenario, the technique works well as the gray levels are relatively spread out and as a result increasing the contrast, enhances the details and we get a relatively good result.

(2) Result of enhancement is bad

For an example of bad enhancement, I deliberately made an image with relatively narrower gray level range as follows:

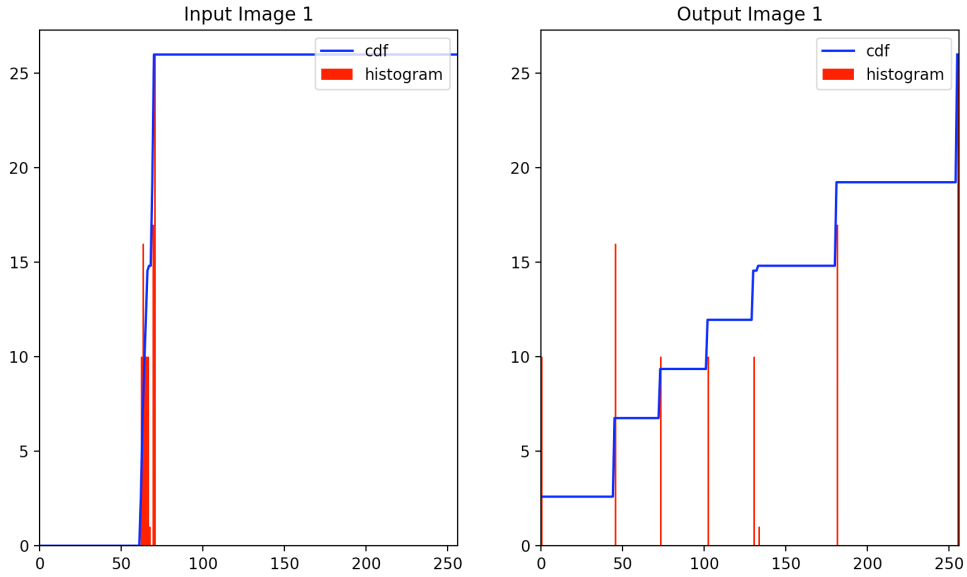
Input1: Gray level values in pixels

64	70	70	70	63	70	70	70	70	64
64	70	63	63	63	63	63	70	70	64
64	70	70	70	70	70	70	70	70	64
64	70	63	63	63	63	63	70	70	64
64	70	70	63	67	62	70	70	70	64
62	62	63	62	63	62	62	62	62	62
69	69	63	69	63	69	62	69	69	69
66	66	66	66	66	66	66	66	66	66
69	69	69	69	69	69	69	69	69	69
65	65	65	65	65	65	65	65	65	65

The gray pixel values for input image 2

Input image 2

After performing the contrast enhancement on this set of image, the results were as follows:

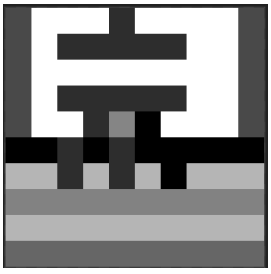


With the resulting output image and the pixels being:

Output1: Gray level values in pixels

73	255	255	255	45	255	255	255	255	73
73	255	45	45	45	45	45	255	255	73
73	255	255	255	255	255	255	255	255	73
73	255	45	45	45	45	45	255	255	73
73	255	255	45	133	0	255	255	255	73
0	0	45	0	45	0	0	0	0	0
181	181	45	181	45	181	0	181	181	181
130	130	130	130	130	130	130	130	130	130
181	181	181	181	181	181	181	181	181	181
102	102	102	102	102	102	102	102	102	102

Gray pixel values for output image 2



The resulting image as output2

Result: Unlike the previous turn we had much closer gray levels and hence by the nature of the contrast enhancement we see a lot of extreme results, in which the detail of the foreground is lost or merged and there is increase in noise as well.

Therefore this method of contrast enhancement with histogram equalization has its advantages and disadvantages and hence must be used depending on the type of the problem.

Problem 4

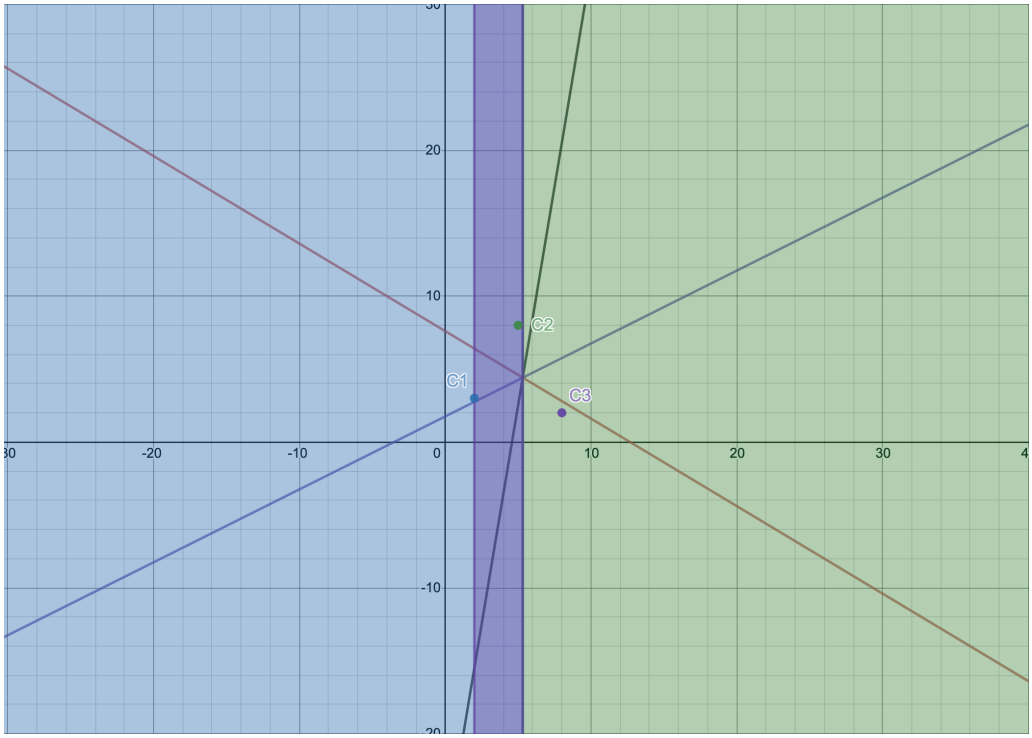
(1) In a 2-dimensional space, I chose the following points for the exercise:

$$C_1(2,3), C_2(5,8), C_3(8,2)$$

The equations for their decision boundaries are lines which are perpendicular to the line joining every two of these points while also passing their mid-points.

For C1 and C2: Line with slope -3/5 and passing through midpoint (3.5,5.5). $y = \frac{-3}{5}x + 7.6$ $(x \leq 5.318)$
For C2 and C3: Line with slope 1/2 and passing through midpoint (6.5,5). $y = \frac{1}{2}x + 1.75$ $(x \geq 5.318)$
For C3 and C1: Line with slope 6 and passing through midpoint (5, 2.5). $y = 6x - 27.5$ $(2 \leq x \leq 5.318)$

On a graph they are plotted as follows.



(2) For each of the above three categories in the Nearest Neighbor rule, an input pattern can be classified as follows based on the shortest Euclid distance from each of the categories.

a. $C_1 \rightarrow (-4,4)$ $(\sqrt{37})$
b. $C_2 \rightarrow (2,8)$ (3)
c. $C_3 \rightarrow (10,4)$ $(2\sqrt{2})$

(3) The three categories above are classified as prototypes ω_i $(i = 1,2,3)$ while the input pattern is an 2-d vector $x = (x_1, x_2)$.
For each of the classes, we give the a-priori probabilities $P(\omega_1) = P(\omega_2) = P(\omega_3) = \frac{1}{3}$.

We can assume an a-prior density function to be, say, the Gaussian Distribution.

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

p(x | ω1) = $\phi(x)$ (x ≦ 5.318); = ; 0 otherwise

p(x | ω1) = $\phi(x)$ (2≦x < 5.318); 0 otherwise

p(x | ω1) = $\phi(x)$ (0x>5.318); 0 otherwise