Assignment 1

COMP 478 Image Processing

Etienne Pham Do

40130483

Etienne Pham Do 40130483

COMP 478 Assignment 1

1.

a) The discrete histogram equalization uses a finite set of numbers, which are integers as well to approximate a PDF through a discrete sum as opposed to its continuous counterpart that uses an integral instead. Though an input intensity of a pixel would be mapped to a new output intensity, the probability of the new intensity would be roughly equal to the probability of the input intensity. In other words, the equalized histogram would not be flat, but would have its pixels spread uniformly across all intensity levels.

b) A histogram-equalized image would have its pixels spread uniformly across all intensity levels. Doing a second pass would only yield a linear transformation as the input image is already histogram-equalized. In other words, there would be no change in intensity.

2.

Given 2 sets of numbers, A = {1,-1,9} and B = {4,5,6}, and a and b both equal to 1, we define H to be the median operator. Using H[af(x,y) + bg(x,y)] = H[A + B] = H[ {5,4,15} ], the operator yields 5. Now, H[f(x,y)] = H[A] = 1 and H[B] = 5, given a sum of 6, which is not equal to H[A + B], violating the property: H[af(x,y) + bg(x,y)] = aH[f(x,y)] + bH[g(x,y)]. Therefore, the median operator is non-linear.

3.

a) Chart

Description automatically generated

b)

i)

We can find s­­k using this formula: sk = with given values:

Table

Description automatically generated

Yields:

Graphical user interface, application, table

Description automatically generated

ii)

Chart, histogram

Description automatically generated

Chart

Description automatically generated

c)

Chart

Description automatically generated

4. For f(x,y) + g(x,y), let h(x,y) be the sum of the 2 images. Given that pixels of f(x,y) have an intensity level of rk and that the pixels of g(x,y) have an intensity of constant value c, the output image h(x,y) would have the resulting intensities of its pixels shifted by c (as in rk + c), but the spacing in the resulting histogram remains the same. Similarly for the case of f(x,y) \* g(x,y), let j(x,y) be the product of the 2 images. This output image would have all the intensities of f(x,y) multiplied by constant c, which doesn’t affect the ratio of the number of pixels per intensity, meaning that the spacing in the resulting histogram, also, remains the same.

**Programming**

**1.**

import cv2

from matplotlib import pyplot as plt

#Greyscaling image and displaying it

current\_image = cv2.imread('HawkesBay.jpeg',0)

cv2.imshow('Greyscale Image', current\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

print('Done showing image')

**2.**

# Calculating histogram from image array (not the most optimal algorithm, so takes a while to run)

rk = []

for i in range(256):

rk.append(i)

pixels\_count\_per\_intensity = []

sum = 0

for x in rk:

for i in range(len(current\_image)):

for j in range(len(current\_image[i])):

if current\_image[i][j] == x:

sum += 1

pixels\_count\_per\_intensity.append(sum)

sum = 0

# Displaying histogram from algorithm

plt.figure()

plt.title("Grayscale Histogram from my algorithm")

plt.xlabel("Gray level")

plt.ylabel("# of Pixels")

plt.plot(rk, pixels\_count\_per\_intensity)

plt.xlim([0, 256])

plt.show()

Chart

Description automatically generated

**3.**

# Using built-in histogram calculation function and displaying histogram

hist = cv2.calcHist([current\_image], [0], None, [256], [0,256])

plt.figure()

plt.title("Grayscale Histogram")

plt.xlabel("Gray level")

plt.ylabel("# of Pixels")

plt.plot(hist)

plt.xlim([0, 256])

plt.show()

Chart, line chart

Description automatically generated

**4.**

# Histogram equalization calculation

def get\_length(a):

return int(a[0])

np = map(get\_length,hist)

np = list(np)

s = []

sum\_of\_prob\_r = 0

sum\_of\_all\_pixels = sum(np)

max\_intensity = len(np) - 1

for i in range(len(np)):

prob\_of\_r = np[i]/sum\_of\_all\_pixels

sum\_of\_prob\_r += prob\_of\_r

s.append(max\_intensity \* sum\_of\_prob\_r)

rounded\_s = [int(round(num)) for num in s]

# Function to associate the rounded sk values to the number of pixels

def get\_number\_of\_pixels\_per\_intensity(intensities\_ptr, new\_np):

sum = 0

intensities\_ptr.append(rounded\_s[0])

for i in range(len(rounded\_s)):

if i == len(rounded\_s) - 1:

new\_np.append(sum)

break

if rounded\_s[i] in intensities\_ptr:

sum += np[i]

else:

intensities\_ptr.append(rounded\_s[i])

new\_np.append(sum)

sum = np[i]

# Populating the arrays that will be used to plot histogram

intensities\_ptr = []

new\_np = []

get\_number\_of\_pixels\_per\_intensity(intensities\_ptr, new\_np)

# Displaying equalized histogram from algorithm

plt.figure()

plt.title("Grayscale Equalized Histogram with algorithm")

plt.xlabel("Gray level")

plt.ylabel("# of Pixels")

plt.plot(intensities\_ptr, new\_np)

plt.xlim([0, 256])

plt.show()

Chart, line chart

Description automatically generated

**5.**

# Displaying equalized histogram using built-in function

equilized\_image = cv2.equalizeHist(current\_image)

hist2 = cv2.calcHist([equilized\_image], [0], None, [256], [0,256])

plt.figure()

plt.title("Grayscale Equalized Histogram with built-in function")

plt.xlabel("Gray level")

plt.ylabel("# of Pixels")

plt.plot(hist2)

plt.xlim([0, 256])

plt.show()

Chart, histogram

Description automatically generated