HW2

Problem1

#%%

#Problem: 01

#importing cv2

import cv2

# Using cv2.imread() method

img = cv2.imread('2\_1.bmp')

# Displaying the image using cv2.imshow()

cv2.imshow('Lena\_Image', img)

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

#Collecting values from 2D array

image\_first\_band = img[:,:,0]

rows = image\_first\_band.shape[0]

coloumns= image\_first\_band.shape[1]

values = []

for i in range(0, rows):

    for j in range(0, coloumns):

        values.append(image\_first\_band[i, j])

frequencies = {x:values.count(x) for x in values}

v, f = frequencies.keys(), frequencies.values()

#%%

#Histogram

import matplotlib.pyplot as plt

ax = plt.subplot(111)

w = 0.3

ax.bar(list(frequencies.keys()), list(frequencies.values()) , width=w, color='b', align='center')

ax.autoscale(tight=True)

plt.title("Histrogram of Lena Image (First Band)")

plt.xlabel("Pixel Intensity")

plt.ylabel("Pixel Frequency")

plt.show()

#%%

#CDF

import numpy as np

probability = []

for item in list(frequencies.values()):

    probability.append(item/sum(list(frequencies.values())))

cp = np.cumsum(probability).tolist()

sorted\_list = sorted(list(frequencies.keys()))

plt.xlabel("Pixel Intensity")

plt.ylabel("Cumulative Probability")

plt.title("Cumulative Distribution Function")

plt.plot(sorted\_list, cp, c='blue')

plt.show()

Result:

A screenshot of a cell phone

Description automatically generated

A close up of a map

Description automatically generated

Problem 2:

#%%

#Problem 02

#importing cv2

import cv2

# Using cv2.imread() method

img = cv2.imread('2\_2.bmp')

# Displaying the image using cv2.imshow()

cv2.imshow('Color Image', img)

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

# Displaying the image using cv2.imshow()

cv2.imshow('Blue Image', img[:, :, 0])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

# Displaying the image using cv2.imshow()

cv2.imshow('Green Image', img[:, :, 1])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

# Displaying the image using cv2.imshow()

cv2.imshow('Red Image', img[:, :, 2])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

#Changing BGR to HSV

import cv2

# Using cv2.imread() method

img = cv2.imread('2\_2.bmp')

hsv\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

# Displaying the image using cv2.imshow()

cv2.imshow('HSV Image', hsv\_image)

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

# Displaying the image using cv2.imshow()

cv2.imshow('Hue', hsv\_image[:, :, 0])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

# Displaying the image using cv2.imshow()

cv2.imshow('Saturation', hsv\_image[:, :, 1])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

#Displaying the image using cv2.imshow()

cv2.imshow('Value', hsv\_image[:, :, 2])

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

Original:

A picture containing tree, sky

Description automatically generated

Red:

A picture containing outdoor, tree, grass, sky

Description automatically generated

Green:

A picture containing outdoor, photo, tree, sky

Description automatically generated

Blue:

A picture containing outdoor, tree, sky, photo

Description automatically generated

HSV image:

A picture containing text

Description automatically generated

Hue:

A picture containing outdoor, sky, flock, nature

Description automatically generated

Saturation:

A picture containing food, military vehicle

Description automatically generated

Value:

A picture containing outdoor, tree, sky, photo

Description automatically generated

Problem 3:

Original Image:



#%%

#Problem 03

#importing cv2

import cv2

# Using cv2.imread() method

X = cv2.imread('books.tif')

# Displaying the image using cv2.imshow()

cv2.imshow('Book Image', X)

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

#%%

img1 = X[:, :, 0]

img2 = X[:, :, 1]

img3 = X[:, :, 2]

rows = img1.shape[0]

coloumns = img1.shape[1]

#%%

import numpy as np

#Forming Blue Channel

z1= (rows, coloumns)

Zimg1 = np.zeros(z1)

for i in range(0, rows, 2):

    for j in range(0, coloumns, 2):

        Zimg1[i, j] = img1[i, j]

#%%

#Forming Green Channel

z2= (rows, coloumns)

Zimg2 = np.zeros(z2)

for i in range(0, rows, 2):

    for j in range(1, coloumns, 2):

        Zimg2[i, j] = img2[i, j]

for i in range(1, rows, 2):

    for j in range(0, coloumns, 2):

        Zimg2[i, j] = img1[i, j]

#%%

#Forming Red Channel

z3= (rows, coloumns)

Zimg3 = np.zeros(z3)

for i in range(1, rows, 2):

    for j in range(1, coloumns, 2):

        Zimg3[i, j] = img1[i, j]

#%%

#Pixel Repeating Blue Channel

zf1 = (rows, coloumns)

ZFimg1 = np.zeros(zf1)

m= 0

n= 0

for i in range(0, rows, 2):

    for j in range(0, coloumns, 2):

        if n<coloumns:

            ZFimg1[m, n] = Zimg1[i, j]

            ZFimg1[m, n+1] = Zimg1[i, j]

            ZFimg1[m+1, n] = Zimg1[i, j]

            ZFimg1[m+1, n+1] = Zimg1[i, j]

            n= n+2

        elif m<rows:

            m= m+2

            n=0

            ZFimg1[m, n] = Zimg1[i, j]

            ZFimg1[m, n+1] = Zimg1[i, j]

            ZFimg1[m+1, n] = Zimg1[i, j]

            ZFimg1[m+1, n+1] = Zimg1[i, j]

            n= n+2

#%%

#Pixel Repeating Green Channel

zf2 = (rows, coloumns)

ZFimg2 = np.zeros(zf2)

m= 0

n= 0

for i in range(0, rows, 2):

    for j in range(1, coloumns, 2):

        if n<coloumns:

            ZFimg2[m, n] = Zimg2[i, j]

            ZFimg2[m, n+1]= Zimg2[i, j]

            n= n+2

        elif m<rows:

            m= m+2

            n=0

            ZFimg2[m, n] = Zimg2[i, j]

            ZFimg2[m, n+1]= Zimg2[i, j]

            n= n+2

s= 1

t= 1

for i in range(1, rows, 2):

    for j in range(0, coloumns, 2):

        if t<coloumns:

            ZFimg2[s, t] = Zimg2[i, j]

            ZFimg2[s, t-1] = Zimg2[i, j]

            t= t+2

        elif s<rows:

            s= s+2

            t=1

            ZFimg2[s, t] = Zimg2[i, j]

            ZFimg2[s, t-1] = Zimg2[i, j]

            t= t+2

#%%

#Pixel Repeating Red Channel

zf3 = (rows, coloumns)

ZFimg3 = np.zeros(zf3)

m= 0

n= 0

for i in range(1, rows, 2):

    for j in range(1, coloumns, 2):

        if n<coloumns:

            ZFimg3[m, n] = Zimg3[i, j]

            ZFimg3[m, n+1] = Zimg3[i, j]

            ZFimg3[m+1, n] = Zimg3[i, j]

            ZFimg3[m+1, n+1] = Zimg3[i, j]

            n= n+2

        elif m<rows:

            m= m+2

            n=0

            ZFimg3[m, n] = Zimg3[i, j]

            ZFimg3[m, n+1] = Zimg3[i, j]

            ZFimg3[m+1, n] = Zimg3[i, j]

            ZFimg3[m+1, n+1] = Zimg3[i, j]

            n= n+2

# %%

image\_final = cv2.merge((ZFimg1, ZFimg2, ZFimg3))

cv2.imshow('Book Color\_Image', image\_final)

#Maintain output window until user presses a key

cv2.waitKey(0)

cv2.destroyAllWindows()

import cv2

#show double type image

#convert to normalized floating point

out = cv2.normalize(image\_final.astype('float'), None, 0.0, 1.0, cv2.NORM\_MINMAX)

cv2.imshow('Color\_Image\_Book', out)

cv2.destroyAllWindows()

Result:

