

# Q1 part1

from PIL import Image

import matplotlib.pyplot as plt

input\_img = Image.open('9.jpeg')

output\_img=input\_img.convert('LA')

plt.figure()

plt.subplot(2,1,1)

plt.imshow(input\_img)

plt.subplot(2,1,2)

plt.imshow(output\_img)

plt.show()



# Q1 part 2

import matplotlib.pylab as plt

import numpy as np

im = plt.imread("mango.JPEG")

print(im.shape)

plt.axis("off")

#Plotting Image

plt.imshow(im)

# converting to greyscales

rgb\_weights = [0.2989, 0.5870, 0.1140]

grayscale\_image = np.dot(im[...,:3], rgb\_weights)

plt.imshow(grayscale\_image, cmap=plt.get\_cmap("gray"))

#Q2

from PIL import Image

from numpy import asarray

# load the image

image1 = Image.open('9.jpeg')

image2 = Image.open('10.jpeg')

data = asarray(image1)

data2= asarray(image2)

#print(data)

#print(data2)

# summarize shape

print(data.shape)

print(data2.shape)

data3= data2.reshape(1280, 590, 3)

#print(data3)

print(data3.shape)

arr1=np.add(data,data3)

arr2=np.subtract(data,data3)

#print(arr2)

arr3=np.multiply(data,data3)

#Displaying the addition of image

img = Image.fromarray(arr1, 'RGB')

img.save('my.png')

img.show()

A screenshot of a cell phone

Description automatically generated

#Displaying the Substraction of image

img2 = Image.fromarray(arr2, 'RGB')

img2.save('my2.png')

img2.show()

A picture containing screenshot

Description automatically generated

#Displaying the Substraction of image

img2 = Image.fromarray(arr2, 'RGB')

img2.save('my2.png')

img2.show()

A screenshot of a cell phone

Description automatically generated

Q3 #Converting from one color space into another (RGB to HSV conversion)

def rgb\_to\_hsv(r, g, b):

r, g, b = r / 255.0, g / 255.0, b / 255.0

cmax = max(r, g, b)

cmin = min(r, g, b)

diff = cmax-cmin

if cmax == cmin:

h = 0

elif cmax == r:

h = (60 \* ((g - b) / diff) + 360) % 360

elif cmax == g:

h = (60 \* ((b - r) / diff) + 120) % 360

elif cmax == b:

h = (60 \* ((r - g) / diff) + 240) % 360

if cmax == 0:

s = 0

else:

s = (diff / cmax) \* 100

v = cmax \* 100

return h, s, v

print(rgb\_to\_hsv(129, 88, 47))

#steps to compute

'HSV – (hue, saturation, value)'

'Divide r, g, b by 255

'Compute cmax, cmin, difference

'Hue calculation :

'if cmax and cmin equal 0, then h = 0

'if cmax equal r then compute h = (60 \* ((g – b) / diff) + 360) % 360

'if cmax equal g then compute h = (60 \* ((b – r) / diff) + 120) % 360

'if cmax equal b then compute h = (60 \* ((r – g) / diff) + 240) % 360

'Saturation computation :

'if cmax = 0, then s = 0

'if cmax does not equal 0 then compute s = (diff/cmax)\*100

'Value computation :

'v = cmax\*100'

Q4

import cv2

img = cv2.imread("9.jpeg")

resized = cv2.resize(img, (100,100), interpolation=cv2.INTER\_LINEAR)

cv2.imshow('Original', img)

cv2.waitKey()

cv2.destroyAllWindows()

img2 = cv2.imread("9.jpeg")

resized = cv2.resize(img2, (100,100), interpolation=cv2.INTER\_CUBIC)

cv2.imshow('Original', img2)

cv2.waitKey()

cv2.destroyAllWindows()

img3=cv2.imread("9.jpeg")

resized = cv2.resize(img3, (100,100), interpolation=cv2.INTER\_NEAREST)

cv2.imshow('Original', img3)

cv2.waitKey()

cv2.destroyAllWindows()

Obsrervation: Almost all the images are similar. Nearest neighbour interpolation is the closest to the resized image. Bilenar interpolation results in smoother image than the nearest neighbour. Bicubic interpolation results in sharper image than the other two.

A screenshot of a computer

Description automatically generated



