CVE Assignment Report Outline

# Question 2 (PS1-2)

## Algorithm

The heart of the problem involves identifying the differentiating threshold between the bright and the dark areas of the image which are to be highlighted. The following algorithm describes the solution to the problem

1. Read image based on the filename provided
2. Specify the interested region to be highlighted (bright or dark)
3. Convert RGB image to Grayscale and plot the intensity histogram to identify the differentiating threshold
4. Use Binary based thresholding to create a binary file image (depending on the region to be highlighted)
5. Assign the area of interest red by maximizing the red channel output for those pixels in the interested area and tuning the other channels at that pixel to 0
6. Save and display the image in the required formats

Iterations to find the best threshold for each image provided the following results.

## Results

Image: circuit.png

|  |  |
| --- | --- |
| A picture containing text, electronics, circuit  Description automatically generated  1 Original Image | A picture containing text, electronics  Description automatically generated  Gray Scale Image |
| A picture containing text, electronics  Description automatically generated  3 Binary Image | A picture containing text, electronics, circuit  Description automatically generated  Final Image |

Image: crack.png

|  |  |
| --- | --- |
| A picture containing outdoor, curb  Description automatically generated  5 Original Image | Gray Scale Image |
| 7 Binary Image | Final Image |

## Source Code

import cv2 as cv

def rgb2gray (img):

img\_gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

# dst = cv.calcHist(img\_gray, [0], None, [256], [0,256])

# plt.hist(img\_gray.ravel(),256,[0,256])

# plt.title('Histogram for gray scale image')

# plt.show()

return img\_gray

def grey2bin (img, op):

if op: # For highlighting bright parts

th, out\_img = cv.threshold(img, 70, 255, cv.THRESH\_BINARY)

return out\_img

else: # For highlighting dark parts

th, out\_img = cv.threshold(img, 163, 255, cv.THRESH\_BINARY\_INV)

return out\_img

def main (img, name, op): #Main Function

cv.imshow('Input Color Image', img)

img\_gray = rgb2gray(img) #Function to Convert color image to greyscale

fname = name[0]+'\_grayscale'+'.'+name[1]

cv.imwrite(fname, img\_gray)

img\_bin = grey2bin(img\_gray, op) #Function to Convert to binary map based on User choice

fname = name[0]+'\_binary'+'.'+name[1]

cv.imwrite(fname, img\_bin)

rows = len(img\_bin)

cols = len(img\_bin[0])

img\_out = img

for x in range(0, rows): # Converts white parts of the binary image red in the final image

for y in range(0, cols):

if img\_bin[x][y] == 0 and op == 0: #For highlighting dark parts

img\_out[x][y][0] = 0

img\_out[x][y][1] = 0

img\_out[x][y][2] = 255

elif img\_bin[x][y] == 255 and op == 1: #For highlighting bright parts

img\_out[x][y][0] = 0

img\_out[x][y][1] = 0

img\_out[x][y][2] = 255

fname = name[0]+'\_output'+'.'+name[1]

cv.imwrite(fname, img\_out)

cv.imshow('Grayscale Image', img\_gray)

cv.imshow('Binary Image', img\_bin)

cv.imshow('Output Color Image', img\_out)

cv.waitKey(0)

cv.destroyAllWindows()

filename = input("Enter File name : ")

img = cv.imread(filename)

filename = filename.split('.')

choice = input("Highlight Bright or Dark Portions? : ")

op = 0

if choice.lower() == "bright":

op = 1

main(img, filename, op)

# Question 3 (PS1-3)

## Algorithm

The objective of the problem was to repeatedly change the intensity of the image based on a defined gamma value until an optimum image was found. The following algorithm describes the solution to the problem

1. Read the image from the given filename
2. Accept a gamma value for the gamma correction
3. Split the image into its respective channels
4. For each channel, modify the intensity at each pixel according to the following equation
5. Merge the channels to create the final image
6. Display the new image along with the original image for comparison
7. Repeat from step 2 if a new gamma value is to be used, else escape the loop
8. Save the image

Iterations to find the best gamma value for each image provided the following results.

## Results

Image: carnival.jpg

|  |  |
| --- | --- |
| 9 Original Image | A picture containing text, grass, sky, outdoor  Description automatically generated  Gamma Corrected Image |

Image: smiley.jpg

|  |  |
| --- | --- |
| A person holding a cookie  Description automatically generated with low confidence  11 Original Image | A person holding a cookie  Description automatically generated with medium confidence  Gamma Corrected Image |

## Source Code

import cv2 as cv

import numpy as np

def gammacorrection(img, gamma): #main function

(B,G,R) = cv.split(img) #Split image into individual channels

B = np.array(255\*(B/255)\*\*(1/gamma), dtype='uint8') #Applying correction formula on each channel

G = np.array(255\*(G/255)\*\*(1/gamma), dtype='uint8')

R = np.array(255\*(R/255)\*\*(1/gamma), dtype='uint8')

img\_out = cv.merge([B,G,R]) #merge channels

return img\_out

name = input("Enter file name = ")

img = cv.imread(name)

g = input("Enter Gamma value = ")

while(g != 'x'):

cv.destroyAllWindows() #Destroy all present windows

nimg = gammacorrection(img, float(g)) #Gamma corrected image

cv.imshow('Original Image', img) #Display image

cv.imshow('Edited Image', nimg) #Display corrected image

cv.waitKey(1000) #Stop windows from closing

g = input("Enter Gamma value or press x to save image = ") #new gamma values

name = name.split('.')

fname = name[0]+'\_gcorrected.'+name[1]

cv.imwrite(fname, nimg) #save file

cv.destroyAllWindows()

# System Specifications

Operating System: macOS Monterey Version 12.5.1

Hardware: MacBook Air 2017 (Intel Core i5)

Python: Conda environment utilizing Python 3.9.1

IDLE: Visual Studio Code