Experiment # 2

AIM: Display of an Image, Negative of an Image (Binary & Gray Scale)

Description:

Digital Image Processing is a significant aspect of data science. It is used in image modification and enhancement so that image attributes can be acquired to lead to a greater understanding of data.

An image is made up of elements called pixels. They are arranged in a two-dimensional manner, and are represented using squares.

There are three main types of images:

1. RGB: Each pixel contains three values for the red, green, and blue color and is stored in three bytes. Each value is in the range 0−2550−255. The values combined make up the resultant color of the pixel.
2. Greyscale: Values range from 0−2550−255 and represent the pixel intensity. Each pixel is stored in 88 bits. 00 depicts a white pixel, while 255255 depicts a black pixel.
3. Binary: Each pixel is stored in one bit, and can have 00 or 255255 as its value. 00 depicts a white pixel, while 255255 depicts a black pixel.

Negative transformation refers to subtracting pixel values from $(L-1)$, where L is the maximum possible value of the pixel, and replacing it with the result.

To negatively transform an image, we loop through the pixels using two for loops. If the image is RGB, the red, green, and blue values are subtracted from $(L-1)$ and the result is stored in place of the values. In the case of greyscale images, the intensity of the pixels is subtracted instead.

Negative transformation is done to bring attention to detail in the darker regions of the image.

Tasks:

I)

One could follow the below given steps to convert a color image to a binary image-

- Import the required library. In all the following examples, the required Python library is **OpenCV**. Make sure you have already installed it.
- Read an the input image using **cv2.imread()**. The RGB image read using this method is in BGR format. Optionally assign the read BGR image to img.
- Now convert this BGR image to grayscale image as below using **cv2.cvtColor()** function. Optionally assign the converted grayscale image to gray.
- Apply thresholding **cv2.threshold()** on the grayscale image gray to convert it to a binary image. Adjust the second parameter (**threshValue**) for better binary image.
- Display the converted binary image.

Example

In this Python program, we convert a color image to a binary image. We also display the binary image.

```
# import required libraries
import cv2


# load the input image
img = cv2.imread('architecture1.jpg')


# convert the input image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


# apply thresholding to convert grayscale to binary image
ret,thresh = cv2.threshold(gray,70,255,0)


# Display the Binary Image
cv2.imshow("Binary Image", thresh)
```

```
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Output??

Example

In this example, we convert a color image to a binary image. We also display the original, grayscale and binary images.

```
# import required libraries
import cv2
import matplotlib.pyplot as plt

# load the input image
img = cv2.imread('architecture1.jpg')

# convert the input image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# apply thresholding to convert grayscale to binary image
ret,thresh = cv2.threshold(gray,70,255,0)

# convert BGR to RGB to display using matplotlib
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Output=??

II) Read color image, then R, G, B part.

```
import cv2

# Load the image
image = cv2.imread("gfg.jpeg",cv2.IMREAD_UNCHANGED)

# Display the image
cv2.imshow("Image", image)

# Wait for the user to press a key
cv2.waitKey(0)

# Close all windows
cv2.destroyAllWindows()
```

imread() and Color Channels

A NumPy array is produced after reading an image file with the cv2.imread() method. By default, the image is saved in the BGR color space. As a result, the blue, green, and red color channels, respectively, correspond to the first three channels of the NumPy array.

```
r1 = image[:,:,0] # get blue channel
g1 = image[:,:,1] # get green channel
b1 = image[:,:,2] # get red channel
```