LINEAR ALGEBRA

IMAGE EDITING USING RGB MATRIX

-Anubuthi Kottapalli
-Anshula Aithal
-A.S.Vidisha

Linear Algebra in Image Processing

Every image is composed of pixels. Hence, it can be viewed as an nxn matrix of pixels.

GRAYSCALE ENCODING:

On grayscale, every colour in each cell of the matrix is specified as a number from 0-255, 0 being the darkest(black), and 255 being the lightest(white). All other numbers represent colours between these two extremes.

RGB MATRIX- ENCODING:

In digital image processing, all colours of an image are treated as some combination of 3 colours- red, green and blue. Hence, each pixel is expressed from 0-255 in terms of red, blue and green individually. With the same concept in mind, image editing is easily accomplished by altering these values in an image.

REAL-TIME APPLICATION:

A real time application of this can be seen in image filters and effects that many mobile applications use.

Filters such as 'sepia','fade','vintage', etc are brought about by altering the original rgb values of every pixel in an image by a specific factor.

For example, 'sepia' employs a transformation matrix of:

$$\begin{bmatrix} 0.393, 0.769, 0.189 \\ 0.349, 0.686, 0.168 \\ 0.272, 0.534, 0.131 \end{bmatrix}$$

PYTHON IN IMAGE PROCESSING

Python is a high-level processing language that has a lot of modules that help in image processing. These include-

~numpy

~matplotlib

~opencv

~PIL

Each of these modules view images in different manners and consequently, different operations can be applied on them.

BASIC CODE FOR IMAGE EDITING :

```
import cv2
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
print("Original Image is: ")
img=Image.open('img1.png')
img=cv2.imread('img1.png')
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

print("Image Editing using RGB Matrix. Choose an option to visualise the different effects t
n=1
while(n>0):
    print("Click 1 to view the negative of the original image")
    print("Click 2 to overlap two images")
    print("Click 3 'subtract' two images,i.e, remove components of 1 from the other")
    print("Click 4 to flip the original image")
    print("Click 5 to view the grayscale version of the original image")
    print("Click 6 to split the original image into it's red, blue and green images")
    print("Click 7 to view the original image in Sepia filter")
    print("Click 0 to exit")
    n=int(input("Enter option: "))
    if(n==1):
        print("Negative Image")
        img_not=cv2.bitwise_not(img)
        cv2.imshow('Inverted',img_not)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    elif(n==2):
        print("Overlapping 2 images")
        img1=cv2.imread('img1.png')
        img2=cv2.imread('img2.png')
        cv2.imshow('Image 2',img2)
        img2_adjust=cv2.resize(img2,(img1.shape[1],img1.shape[0]))
        img=cv2.add(img1,img2_adjust)

        cv2.imshow('Added Image',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    elif(n==3):
        print("Subtracting two images")
        print("The second image is displayed")
        img2=cv2.imread('img2.png')
        cv2.imshow('Image 2',img2)
        img2_adjust=cv2.resize(img2,(img1.shape[1],img1.shape[0]))
        img = cv2.subtract(img2_adjust, img1)
        cv2.imshow('Subtraction',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    elif(n==4):
        print("Flipping the image")
        img=cv2.imread('img1.png')
        img_flip=cv2.flip(img,1)
        cv2.imshow('Flip',img_flip)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

```
    elif(n==5):
        print("Image in Grayscale is displayed")
        image = Image.open('img2.png')
        image_grayscale = image.convert(mode='L')
        print(image_grayscale)
        plt.imshow(image_grayscale,cmap='Greys_r')

    elif(n==6):
        print("Spiltting the image into its red, green and blue components now")
        img = Image.open('img2.png')
        data = img.getdata()
        r = [(d[0], 0, 0) for d in data]
        g = [(0, d[1], 0) for d in data]
        b = [(0, 0, d[2]) for d in data]

        img.putdata(r)
        img.save('r.png')
        imgx=cv2.imread('r.png')
        cv2.imshow('image-red',imgx)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

        img.putdata(g)
        img.save('g.png')
        imgy=cv2.imread('g.png')
        cv2.imshow('image-green',imgy)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

        img.putdata(b)
        img.save('b.png')
        imgz=cv2.imread('b.png')
        cv2.imshow('image-blue',imgz)
        cv2.waitKey(0)
```

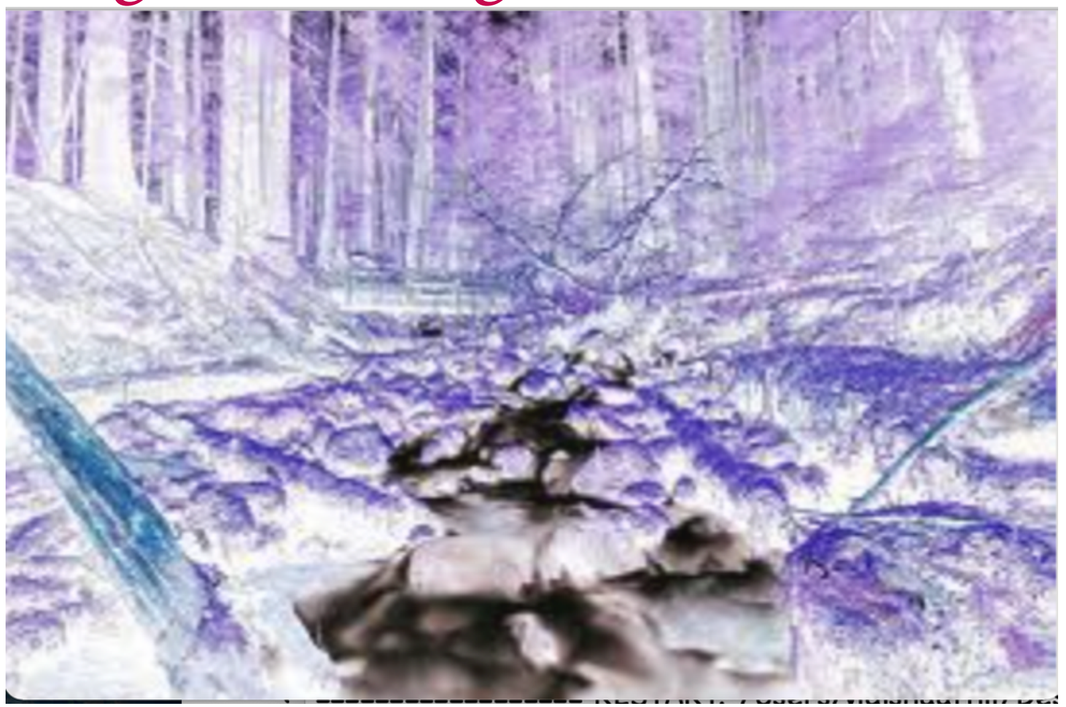
```
    elif(n==7):
        print("Sepia filter applied on the image")
        img = cv2.imread('img2.png')
        original = img.copy()
        img = np.array(img, dtype = np.float64)
        img = cv2.transform(img, np.matrix([[ 0.393, 0.769, 0.189],
                                            [ 0.349, 0.686, 0.168],
                                            [ 0.272, 0.534, 0.131]]))
        img[np.where(img>255)] = 255
        img = np.array(img, dtype = np.uint8)
        cv2.imshow("original", original)
        cv2.imshow("Output", img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

    else:
        break
print("Thank you")
```

IMAGES :

Original Image :

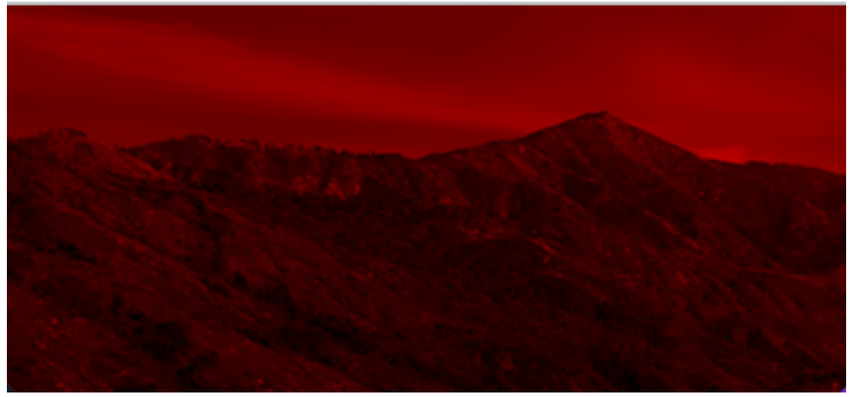
Negative Image :

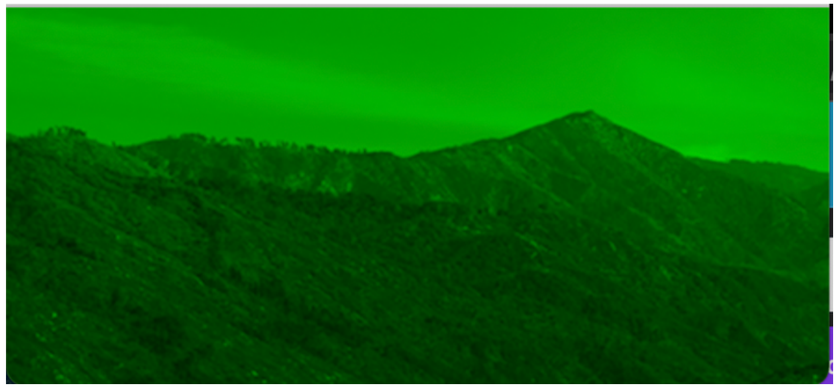


Flipped Image :

Original Image :

Image after applying RGB Filter :





OpenCV Kernal and Convolution Transformations

Convolution is the operatioon of transforming an image by splitting it into small parts called windows and applying an operator called kernel for every part.

The kernel is usually fixed, a small 2D array containing numbers. An important part in the kernel is the center which is called anchor point.

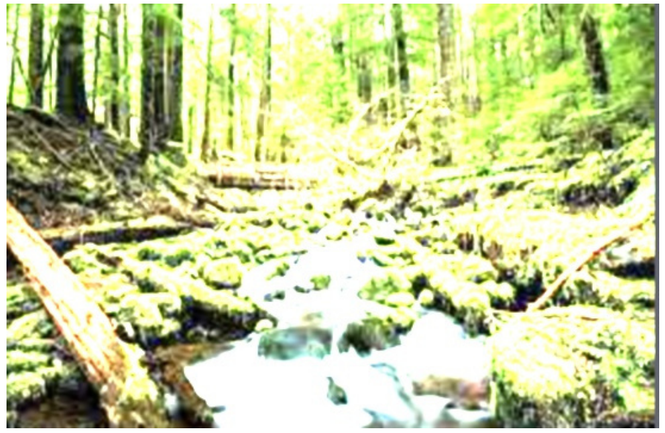
Examples of Kernel transformation:

Sharpening an image

-1	-1	-1
-1	9	1
-1	-1	-1

Sepia:

0.272	0.534	0.131
0.349	0.686	0.168
0.393	0.769	0.189



REFERENCES

1). Applications of Linear Algebra in Image Filters [Part I]- Operations:
By Hemanth Nag
2).How to apply filters to images with Python: By Juan Benitaz

THANK YOU