CLASS : B.E. E &TC **SUBJECT: DIVP**

EXPT. NO. : 2 **DATE: 28/08/2020**

TITLE : PERFORM CONVERSION BETWEEN COLOUR SPACES

CO 1:	Apply the fundamentals of digital image processing to perform various operations on an image-enhancement in spatial domain/ frequency domain, image-restoration, image compression, video filtering and video compression on a given gray image. Examine the effect of varying the mask size and density of noise in an image and comment on the obtained results.
CO4:	Carry out experiments as an individual and in a team, comprehend and write a laboratory record and draw conclusions at a technical level.

AIM :

- **To implement a Matlab code for the conversion between colour spaces**

  1.RGB to HSI

  2. RGB to YIQ

  3. RGB to CMY

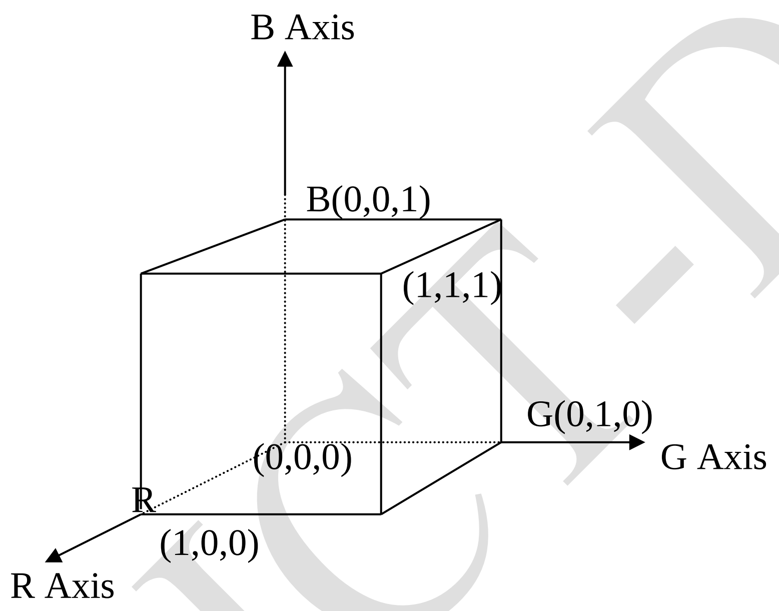
- **To plot Histogram of each colour plane**

SOFTWARES REQUIRED: Matlab 7.0. or above

THEORY:

2.1 The RGB Color Model: (R: Red, G: Green, B: Blue)

Here each color appears in its primary spectral Components of red, green and blue. This model is based on Cartesian coordinate system as shown in fig. 2.1. Here all the values of RGB are assumed in the range 0 to1 for convenience. Images in the RGB Color Model consist of three independent image planes, one of each primary color.



At coordinates (0,0,0) color is black and at coordinate (1,1,1) color is white.

Fig 2.1: Schematic of RGB colour cube

2.2 The YIQ color model: (Y: luminance, I: In-phase, Q: Quadrature)

The YIQ color model is used in commercial TV broadcasting and for maintaining compatibility with Monochrome TV Standards. The Y component provides all the video information required by a monochrome TV. It is called as luminance. Color information is given by I & Q components and is called as chrominance.

2.3 The HSI model: (H: Hue, S: Saturation, I: Intensity)

Hue H is a color attribute that describes a pure color. Whereas, saturation S gives a measure of degree to which a pure color is diluted. This model is ideal for developing Image processing applications based on color descriptions natural to human. This model decouples intensity component I from the color components Hue and Saturation (H & I).

2.4 The CMY model: (C: Cyan, M: Magenta, Y: Yellow)

Cyan, magenta, and yellow are the secondary colors with respect to the primary colors of red, green, and blue. However, in this subtractive model, they are the primary colors and red, green, and blue, are the secondaries. In this model, colors are formed by subtraction, where adding different pigments causes various colors not to be reflected and thus not to be seen. Here, white is the absence of colors, and black is the sum of all of them. This is generally the model used for printing.Most devices that deposit color pigments on paper (such as Color Printers and Copiers) requires CMY data input or perform RGB to CMY conversion internally

2.5 Conversion from RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.2875 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2.6 Conversion from RGB to HSI

The following expressions give the HSI values in the range, 0-1 from a set of RGB values in the same range

$I = 1/3\ (R+G+B)$

$S = 1 - [3/(R+G+B)]\ [\min (R,G,B)]$

$\theta = \mathrm{Cos}^{-1} \{ \tfrac{1}{2} [(R-G) + (R-B)] / [(R-G)^2 + (R-B)(G-B)]^{1/2} \}$

$H = \theta \qquad \text{if } B<=G$

$\quad = 360 - \theta \qquad \text{if } B>G$

2.7 Conversion from RGB to CMY

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2.8 Conclusion:

In this experiment I studied the different colourspace conversion techniques for an image. I also wrote a program for the same and observed the changes that occur in an image after it undergoes changes from RGB to CMY, YIQ and HIS colourspaces respectively.

2.9 References:

i. Gonzalez R, Woods R, “Digital image processing”, Pearson Prentice Hall, 2008.

ii. Gonzalez R, Woods R, Steven E, “Digital Image Processing Using MATLAB®”, McGraw Hill Education, 2010.

iii. Jayaraman S, Esakkirajan S and Veerakumar T, “Digital Image Processing” Tata McGraw Hill, 2010

iv. Joshi, Madhuri A. “Digital Image Processing: an algorithm approach”, PHI Learning Pvt. Ltd., 2006.

v. Pictures taken from: http://www.imageprocessingplace.com/root_files_V3/image_databases.html

____

(Course Teacher)

CLASS	**:**	**B.E (E &TC)**	**COURSE**	**:**	**DIVP**
AY	**:**	**2020-21 (SEM- I)**	**DATE**	**:**	**28/08/2020**
EXPT. NO.	**:**	**2**	**CLASS & ROLL NO**	**:**	**BE VIII 42410**
TITLE	**:**	**PERFORM CONVERSION BETWEEN COLOUR SPACES**			

I. CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math

def getHSI(img):
    with np.errstate(divide='ignore', invalid='ignore'):
        bgr = np.float32(img) / 255
        blue = bgr[:, :, 0]
        green = bgr[:, :, 1]
        red = bgr[:, :, 2]
def calc_intensity(red, blue, green):
       return np.divide(blue + green + red, 3)
def calc_saturation(red, blue, green):
        minimum = np.minimum(np.minimum(red, green), blue)
        saturation = 1 - (3 / (red + green + blue + 0.001) * minimum)
        return saturation
def calc_hue(red, blue, green):
         hue = np.copy(red)
         for i in range(0, blue.shape[0]):
             for j in range(0, blue.shape[1]):
                hue[i][j] = 0.5 * ((red[i][j] - green[i][j]) + (red[i][j] - blue[i][j])) / \
                          math.sqrt((red[i][j] - green[i][j]) ** 2 +
                                  ((red[i][j] - blue[i][j]) * (green[i][j] - blue[i][j])))
                 hue[i][j] = math.acos(hue[i][j])

                 if blue[i][j] <= green[i][j]:
                     hue[i][j] = hue[i][j]
                 else:
                     hue[i][j] = ((360 * math.pi) / 180.0) - hue[i][j]
return hue

        # Merge channels into picture and return image
        hsi = cv2.merge(
            (calc_hue(red, blue, green), calc_saturation(red, blue, green), calc_intensity(red, blue, green)))
        return hsi
def getYIQ(img):
    # y=0.2989 * R + 0.5870 * G + 0.1140 * B
    # I=0.60*R - 0.28*G-0.32*B
    # Q=0.21*R -0.52*G+0.31*B
    YIQ = np.zeros([img.shape[0], img.shape[1], 3])
    for i in range(0, img.shape[0]):
        for j in range(0, img.shape[1]):
            YIQ[i, j, 0] = 0.2989 * img[i, j, 0] + 0.5870 * img[i, j, 1] + 0.1140 * img[i, j, 2];
            YIQ[i, j, 1] = 0.596 * img[i, j, 0] - 0.274 * img[i, j, 1] - 0.322 * img[i, j, 2];
            YIQ[i, j, 2] = 0.211 * img[i, j, 0] - 0.523 * img[i, j, 1] + 0.312 * img[i, j, 2];
    return YIQ

def getCMY(img):
    # Load image with 32 bit floats as variable type
    bgr = np.float32(img) / 255

    # Separate color channels
    blue = bgr[:, :, 0]
    green = bgr[:, :, 1]
    red = bgr[:, :, 2]

    for i in range(0, blue.shape[0]):
        for j in range(0, blue.shape[1]):
            c = 1 - red
            m = 1 - green
            y = 1 - blue

    # Merge channels into picture and return image
    cmy = cv2.merge((c, m, y))
    return cmy

# read a coloured image
img = cv2.imread('Images/icecream.jpg', cv2.IMREAD_COLOR)

# convert RGB to HSI
hsi = getHSI(img)
# convert RGB to YIQ
yiq = getYIQ(img)
# convert RGB to CMY
cmy = getCMY(img)

plt.figure(0)
plt.imshow(img)
plt.title('Original RGB image')

plt.figure(1)
plt.imshow(hsi)
plt.title('HSI image')

plt.figure(2)
plt.imshow(yiq)
plt.title('YIQ image')

plt.figure(3)
plt.imshow(cmy)
plt.title('CMY image')

plt.show()
```

RESULTS: