Exp (4): Color space conversion

Aim: To learn how to convert color images from one color space to another.

Theory: There are many color spaces employed to represent images in certain coordinates. Some color spaces are employed to separate luminance (**intensity**) information from the chrominance (**hue** and **saturation**) information which enables the processing of color images using luminance information only.

RGB: The best-known format is RGB which encodes the color as three channels (red, green, and blue). The Channel operator allows the user the ability to manipulate individual channels of a color image.

HSV: While **RGB** is an easy method of representing color, it is not a useful format for the purposes of analysis and recognition. In the **RGB** format, the intensity of a pixel is mixed in with its hue, and this creates a difficulty in many analysis techniques. The **HSV** color model represents data in three channels known as **hue**, **saturation**, and **value**. The **V** channel represents the intensity information. The **H** channel represents the **hue**, and is measured in angles between **0**◦ and **360**◦. The **S** represents the saturation which is a measure of the color purity.

The YUV family: The **YUV** family of color models has several similar variants, and for many applications, the selection of which variant to use offers very little advantage. This family includes two color models: **YUV** and **YIQ**. The **YUV** representation is linear transformation, in which each channel is a weighted linear combination of the original **RGB** values. The **YIQ** is similar to the **YUV**. It is also called **NTSC** and is used in **televisions** in the United States. The **Y** channel is the **Intensity**, while the **I** and **Q** channels made up the **chrominance** information.

YCbCr: The model is widely used for digital **video**. In this format, **intensity** information is stored as a single component (**Y**), and **chrominance** information is stored as two color difference components (**Cb** and **Cr**).

CIE L*A*B*: This model is designed to emulate human perception as it is based on human responses of different color frequencies. To compute the **L*a*b*** values, the **RGB** values are first converted to **XYZ** space through a linear conversion. After that, the second conversion is done to the **CIE L*a*b*** space.

Procedure:

(1) Load color image (astronaut):	(2) Convert the RGB image to the HSV

import numpy as np
from skimage import data, color
import pylab as pl
RGB = data.astronaut()
	HSV = color.rgb2hsv(RGB)
	H, S, V = HSV[:, :, 0], HSV[:, :, 1], HSV[:, :, 2]
	pl.imshow(np.concatenate((H, S, V), axis = 1))
	pl.show()

(3) Squares the H channel	(4) Create a new image using the processed H with S and V
H_square = np.square(H)	**HSV[:, :, 0] = H_square**
(5) Convert the image back to the RGB	(6) Convert the image in (1) to YCbCr

new_RGB = color.hsv2rgb(HSV)
pl.imshow(new_RGB)
pl.show()
	Ycbcr = color.rgb2ycbcr(RGB)
	Y, cb, cr = Ycbcr[:, :, 0], Ycbcr[:, :, 1], Ycbcr[:, :, 2]

(7) Threshold luminance channel and then concatenate the Y, Cb, and Cr channels along the horizontal direction	(8) Display the result of (7)

Y = 255. * (Y > 100) **# threshold luminance channel to generate binary image**
Ycbcr_concatenate = np.concatenate((Y, cb, cr), axis = 1)
	pl.imshow(Ycbcr_concatenate)
	pl.show()

(9) Brighten the image in (1) by adding 0.5 to the Y channel of the YIQ space	(10) Convert the YIQ image to the RGB space and display it

YIQ = color.rgb2yiq(RGB)
Y, I, Q = YIQ[:, :, 0], YIQ[:, :, 1], YIQ[:, :, 2]
Y_new = Y + 0.5
	YIQ[:, :, 0] = Y_new
	YIQ2RGB = color.yiq2rgb(YIQ)
	pl.imshow(YIQ2RGB)
	pl.show()

Discussion:

(1) Copy a color image from the data directory of the skimage to a new folder, then read the image using PIL and numpy, then convert the image into (1) grayscale, [**Hint: skimage.color.rgb2gray**] (2) CIE L*a*b* [**Hint: skimage.color. rgb2lab**]?

(2) Convert the RGB image in (1) into HSV space, then set the value of pixels in the middle square region of the luminance channel to zero and return the image back into the RGB and display the resulting image?