**CV Practical No.: 5**

**Aim: Color model conversion**

**New Concept:**

**i. cv2.COLOR\_BGR2HSV:** This is a flag used in OpenCV (cv2), which denotes a color space conversion from BGR (Blue, Green, Red) to HSV (Hue, Saturation, Value). The BGR color space is used by default in OpenCV, but the HSV color space is often preferred for certain image processing tasks because it separates the chromatic content (hue) from intensity (value), which can be more intuitive for color manipulation.

**ii. cv2.COLOR\_BGR2LAB:** This flag is used for color conversion from BGR (Blue, Green, Red) to LAB (Lightness, A, B). The LAB color space is based on human vision and is device-independent, making it useful in various applications like image segmentation and recognition.

**iii. cv2.COLOR\_BGR2YCrCb:** This is used to convert an image from BGR to the YCrCb color space. YCrCb separates the luminance (Y) from the chrominance (Cr and Cb), making it particularly useful for compression (e.g., JPEG) and video applications.

**iv. cv2.COLOR\_BGR2CMYK:** This conversion flag indicates converting an image from BGR to CMYK (Cyan, Magenta, Yellow, Key/Black). The CMYK color space is commonly used in printing. However, OpenCV does not have a direct function for this conversion because CMYK is not typically used in image processing tasks for display. Often, conversion from RGB to CMYK is done manually.

**v. np.zeros\_like:** This is a NumPy function that creates a new array of the same shape and type as the provided input array, but with all elements initialized to zero. It is useful when you want to create an empty (black) image or array with the same dimensions and data type as an existing one.

**Theory:**

**i. HSV Image:**

The HSV color space is a cylindrical representation of colors that is often used in computer vision, especially in tasks involving color segmentation, filtering, and image enhancement. It stands for Hue, Saturation, and Value, which are the three components of a color in this space.

**Components:**

* **Hue (H):** Represents the type of color (e.g., red, blue, green, etc.). It is measured in degrees on the color wheel, typically ranging from 0° to 360°:
  + 0° or 360° is red,
  + 120° is green,
  + 240° is blue, and so on.
* **Saturation (S):** Represents the intensity or purity of the color. A saturation of 0 means the color is a shade of gray, and a saturation of 100 means the color is fully saturated, without any gray.
* **Value (V):** Represents the brightness of the color. A value of 0 means black, and a value of 100 means full brightness.

**ii. LAB Image:**

The LAB color space is a perceptually uniform color model that was developed to approximate human vision. It is device-independent, meaning it can represent colors consistently across different devices (e.g., monitors, printers, etc.). The LAB color space consists of three components: L (Lightness), A (Green to Red), and B (Blue to Yellow).

**Components:**

* **L (Lightness):** Represents the brightness of the color, where 0 is black and 100 is white. This is similar to the intensity component in grayscale images.
* **A (Green to Red):** Represents the color spectrum from green to red. Negative values indicate green, and positive values indicate red.
* **B (Blue to Yellow):** Represents the color spectrum from blue to yellow. Negative values indicate blue, and positive values indicate yellow.

**iii. YCbCr Image:**

The YCbCr color space is primarily used in video compression and broadcast standards (e.g., JPEG, MPEG, and digital television). It separates the image into one luminance component and two chrominance components.

**Components:**

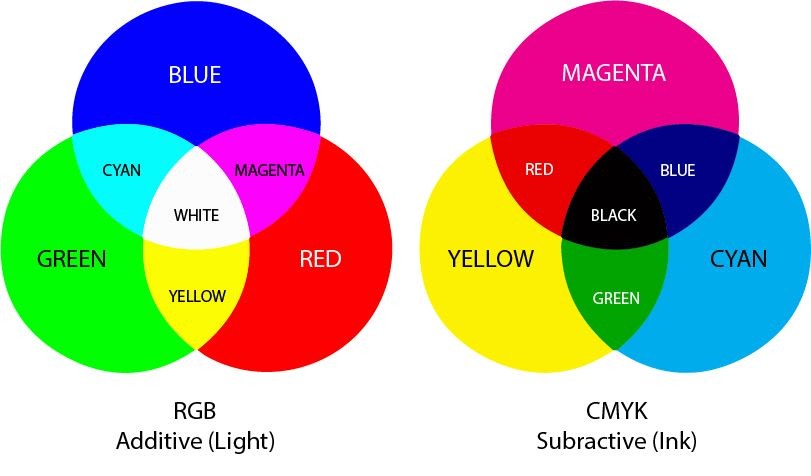
* **Y (Luma):** Represents the brightness or lightness of the color. It is a grayscale image that contains all the intensity information of the image.
* **Cb (Chroma Blue):** Represents the difference between the blue component and the luminance. It encodes the chrominance (color) information, but specifically how much blue is present in the image.
* **Cr (Chroma Red):** Represents the difference between the red component and the luminance. Similar to Cb, it encodes the chrominance, but specifically how much red is present.

**iv. CMYK Image:**

The CMYK color space is primarily used in colorprinting. It is a subtractive color model, meaning that it represents colors based on the subtraction of various percentages of the four color components from a white background.

**Components:**

* **C (Cyan):** A greenish-blue color.
* **M (Magenta):** A purplish-red color.
* **Y (Yellow):** A bright yellow color.
* **K (Key/Black):** Used to add depth and detail to the image, and to prevent the colors from being washed out due to the limits of cyan, magenta, and yellow inks.

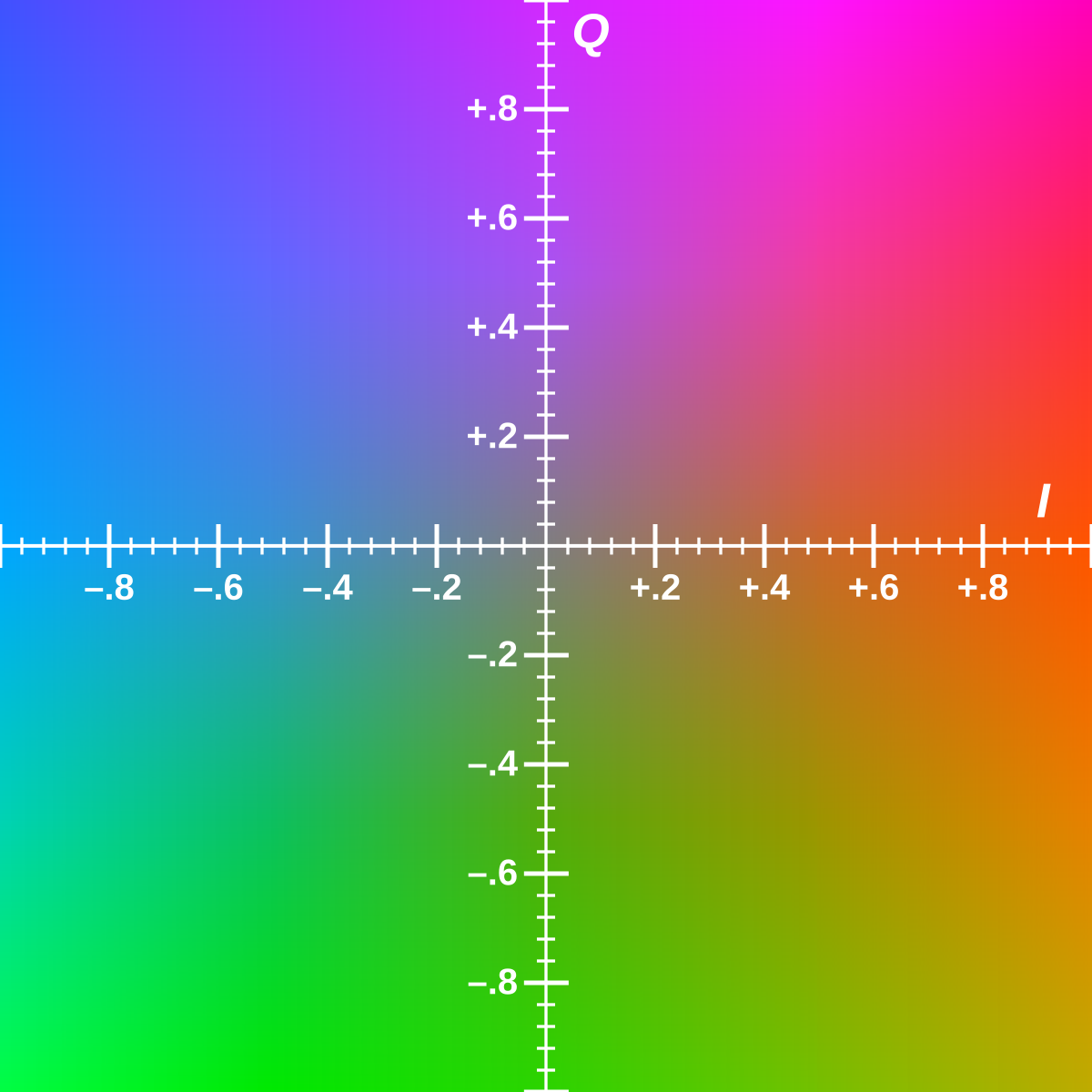


**v. YIQ Image:**

The YIQ color space is used primarily in the NTSC television broadcast standard. It is similar to YCbCr but was designed to be more compatible with color TV systems in the United States. Like YCbCr, it separates luminance from chrominance.

**Components:**

* **Y (Luminance):** Represents the brightness of the image, similar to the Y component in YCbCr.
* **I (In-phase):** Represents the red-to-green color component. It indicates how much red is present in relation to green.
* **Q (Quadrature):** Represents the blue-to-yellow color component, similar to the Cb and Cr components in YCbCr.



**Program:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the image

image = cv2.imread('images.jpeg')

rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

# Convert to HSV (Hue, Saturation, Value)

hsv\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

# Convert to LAB (CIELAB) color space

lab\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2LAB)

# Convert to YCbCr

ycbcr\_image = cv2.cvtColor(rgb\_image, cv2.COLOR\_BGR2YCrCb)

# Convert to CMYK

#cmyk\_image = cv2.cvtColor(rgb\_image, cv2.COLOR\_BGR2CMYK)

# Convert to YIQ

yiq\_image = np.zeros\_like(rgb\_image, dtype=np.float32)

yiq\_image[:,:,0] = 0.299 \* rgb\_image[:,:,2] + 0.587 \* rgb\_image[:,:,1] + 0.114 \* rgb\_image[:,:,0]

yiq\_image[:,:,1] = 0.596 \* rgb\_image[:,:,2] - 0.274 \* rgb\_image[:,:,1] - 0.322 \* rgb\_image[:,:,0]

yiq\_image[:,:,2] = 0.211 \* rgb\_image[:,:,2] - 0.523 \* rgb\_image[:,:,1] + 0.312 \* rgb\_image[:,:,0]

yiq\_image = np.clip(yiq\_image, 0, 255).astype(np.uint8)

# Display the original and converted images

#cv2.imshow('Original Image', image)

#cv2.imshow('HSV Image', hsv\_image)

#cv2.imshow('LAB Image', lab\_image)

#cv2.imshow('YCbCr Image', ycbcr\_image)

#cv2.imshow('CMYK Image', cmyk\_image)

#cv2.imshow('YIQ Image', yiq\_image)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(4,2,1)

plt.imshow(rgb\_image)

#plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(4,2,2)

plt.imshow(hsv\_image)

#plt.imshow(hsv\_image, cmap = 'gray')

plt.title('HSV Image')

plt.axis('off')

#Image3

plt.subplot(4,2,3)

plt.imshow(rgb\_image)

#plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image4

plt.subplot(4,2,4)

plt.imshow(lab\_image)

#plt.imshow(lab\_image, cmap = 'gray')

plt.title('LAB Image')

plt.axis('off')

#Image5

plt.subplot(4,2,5)

plt.imshow(rgb\_image)

#plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image6

plt.subplot(4,2,6)

plt.imshow(ycbcr\_image)

#plt.imshow(ycbcr\_image, cmap = 'gray')

plt.title('YCbCr Image')

plt.axis('off')

#Image7

plt.subplot(4,2,7)

plt.imshow(rgb\_image)

#plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image8

plt.subplot(4,2,8)

plt.imshow(yiq\_image)

#plt.imshow(yiq\_image, cmap = 'gray')

plt.title('YIQ Image')

plt.axis('off')

plt.show()

# Wait for a key press and then close the windows

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output:**

