EX.NO:2	**DATE: 23.01.2025**

RGB TO CMY COLOR MODEL AND RGB TO HSV COLOR MODEL AND SPLITTING RGB AND HSV

Aim

To Understand and execute the different color models and also visualizing the respective color split planes

Algorithm

1. RGB to CMY Color Model

i. Load the Input Image:
   - Use cv2.imread() to load the image in RGB format.

ii. Convert Color Space:
   - Convert the image from BGR to RGB using cv2.cvtColor().

iii. Normalize Pixel Values:
   - Scale the pixel values to the range [0, 1] by dividing each value by 255.

iv. Compute CMY Values:
   - Apply the formulas:
      i. C=1−RC = 1 - RC=1−R, M=1−GM = 1 - GM=1−G, Y=1−BY = 1 - BY=1−B.
   - Scale the CMY values back to the range [0, 255] and convert them to integers.

v. Save and Display:
   - Save the resultant CMY image using cv2.imwrite() and display it.

2. RGB Color Splitting

i. Import Necessary Modules:
   - Use libraries like cv2 and matplotlib.pyplot.

ii. Load the Image:
   - Read the RGB image with cv2.imread().

iii. Separate Color Channels:
   - Use cv2.split() to extract the R, G, and B channels.

iv. Save Results (Optional):
   - Save the individual grayscale channel images with cv2.imwrite().

3. RGB to HSV Color Model

i. Import Libraries:

- Include cv2 and matplotlib.pyplot.

ii. Load the Image:
   - Read the input image using cv2.imread().

iii. Convert RGB to HSV:
   - Use cv2.cvtColor() with the cv2.COLOR_BGR2HSV flag.

iv. Display the Result:
   - Visualize the HSV image.

4. HSV Color Splitting

i. Import Required Libraries:
   - Import numpy and cv2.

ii. Load the Image:
   - Use cv2.imread() to load the image in BGR format.

iii. Convert to HSV:
   - Convert the image to HSV color space using cv2.cvtColor() with the cv2.COLOR_BGR2HSV flag.

iv. Split HSV Channels:
   - Extract the Hue, Saturation, and Value channels using cv2.split().

v. Display Results:
   - Show the separated channels.

5. RGB to YIQ Color Model

i. Load the RGB Image:
   - Use cv2.imread() to load the image.

ii. Define YIQ Conversion Matrix:
   - Use the following transformation matrix:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

iii. Normalize Pixel Values:
   - Scale RGB values to the range [0, 1].

iv. Perform Matrix Multiplication:
   - Apply the transformation matrix to compute the YIQ values.

v. Save and Display:
   - Save the YIQ image using cv2.imwrite() and display the result.

CODE

COLOR MODELS

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

image = cv2.imread("FRUITS.jpg",1)
image1 = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
image.shape

(1414, 2119, 3)

image = cv2.imread("image1.jpg",1)
image2 = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
image.shape

(2160, 3840, 3)

fig, axs = plt.subplots(1, 2)
axs[0].imshow(image1)
axs[0].axis('off')
axs[1].imshow(image2)
axs[1].axis('off')
plt.show()
```

R,G,B Image Planes

```
#IMAGE1
B, G, R = cv2.split(image)

zeros = np.zeros_like(B)

red_image = cv2.merge([zeros, zeros, R])
green_image = cv2.merge([zeros, G, zeros])
blue_image = cv2.merge([B, zeros, zeros])
```

```
red_image = cv2.cvtColor(red_image, cv2.COLOR_BGR2RGB)
green_image = cv2.cvtColor(green_image, cv2.COLOR_BGR2RGB)
blue_image = cv2.cvtColor(blue_image, cv2.COLOR_BGR2RGB)

fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(red_image)
axs[0].set_title('Red Channel')
axs[0].axis('off')

axs[1].imshow(green_image)
axs[1].set_title('Green Channel')
axs[1].axis('off')

axs[2].imshow(blue_image)
axs[2].set_title('Blue Channel')
axs[2].axis('off')

plt.show()

cv2.imwrite('red_channel.jpg', red_image)
cv2.imwrite('green_channel.jpg', green_image)
cv2.imwrite('blue_channel.jpg', blue_image)
```


```
True
```

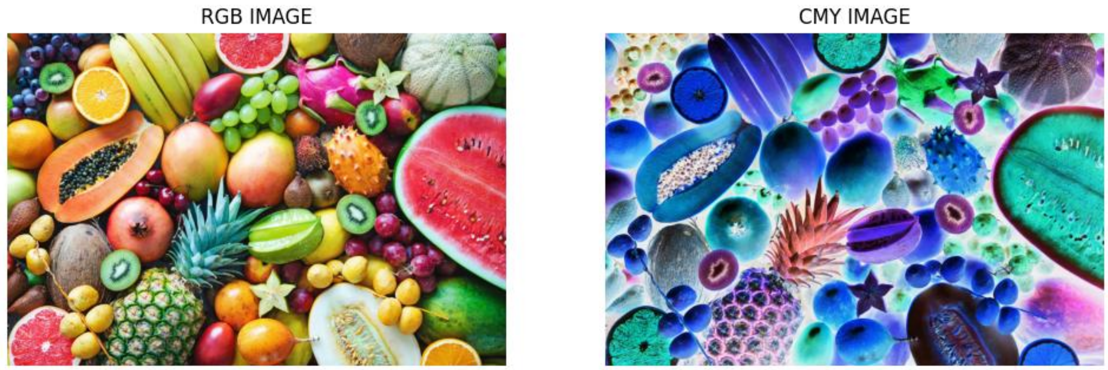
```
#IMAGE2
B, G, R = cv2.split(image)

zeros = np.zeros_like(B)

red_image = cv2.merge([zeros, zeros, R])
green_image = cv2.merge([zeros, G, zeros])
blue_image = cv2.merge([B, zeros, zeros])

red_image = cv2.cvtColor(red_image, cv2.COLOR_BGR2RGB)
green_image = cv2.cvtColor(green_image, cv2.COLOR_BGR2RGB)
blue_image = cv2.cvtColor(blue_image, cv2.COLOR_BGR2RGB)

fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(red_image)
axs[0].set_title('Red Channel')
axs[0].axis('off')
```

```
axs[1].imshow(green_image)
axs[1].set_title('Green Channel')
axs[1].axis('off')

axs[2].imshow(blue_image)
axs[2].set_title('Blue Channel')
axs[2].axis('off')

plt.show()

cv2.imwrite('red_channel.jpg', red_image)
cv2.imwrite('green_channel.jpg', green_image)
cv2.imwrite('blue_channel.jpg', blue_image)
```

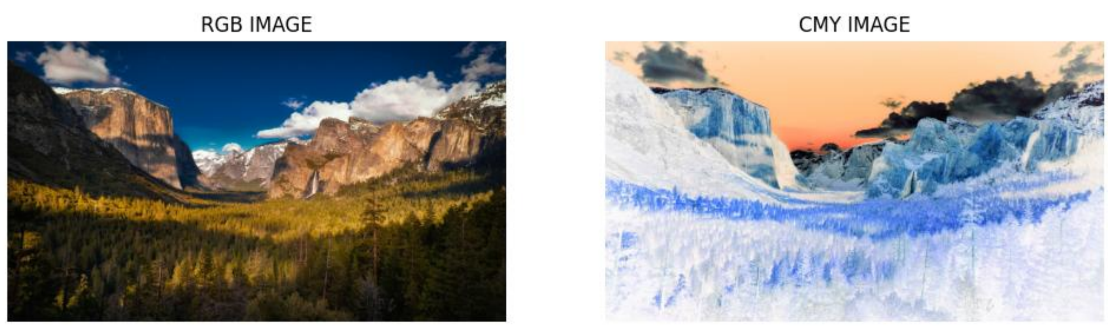

```
True
```

RGB to CMY

```
#IMAGE1
cmy = 255 - image1
plt.figure(figsize=(12, 10))

plt.subplot(1, 2, 1)
plt.imshow(image1)
plt.title('RGB IMAGE')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cmy)
plt.title('CMY IMAGE')
plt.axis('off')

plt.show()
```

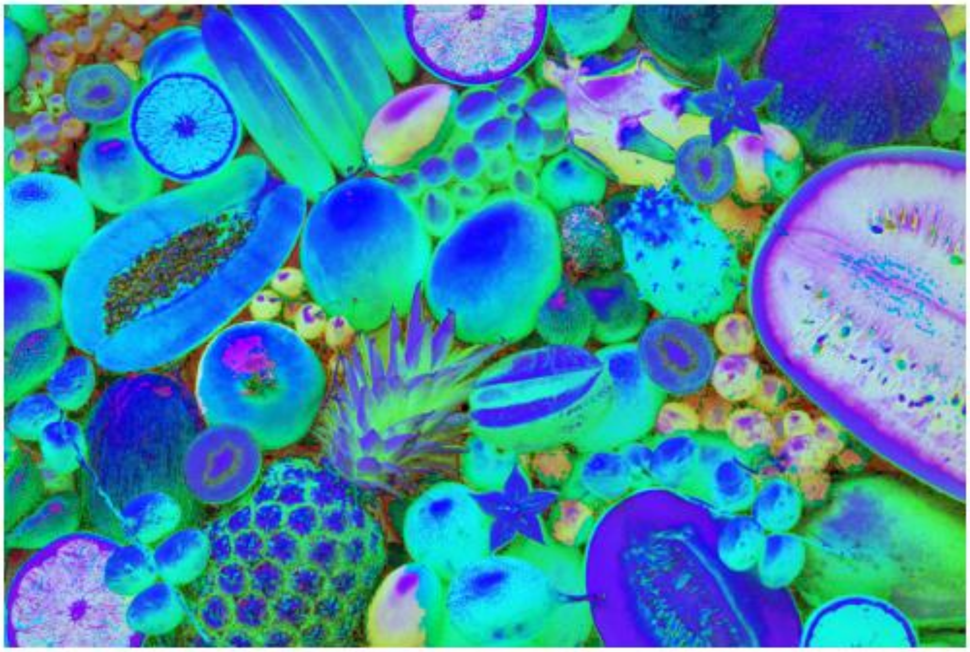



```
#IMAGE2
cmy = 255 - image2
plt.figure(figsize=(12, 10))

plt.subplot(1, 2, 1)
plt.imshow(image2)
plt.title('RGB IMAGE')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cmy)
plt.title('CMY IMAGE')
plt.axis('off')

plt.show()
```

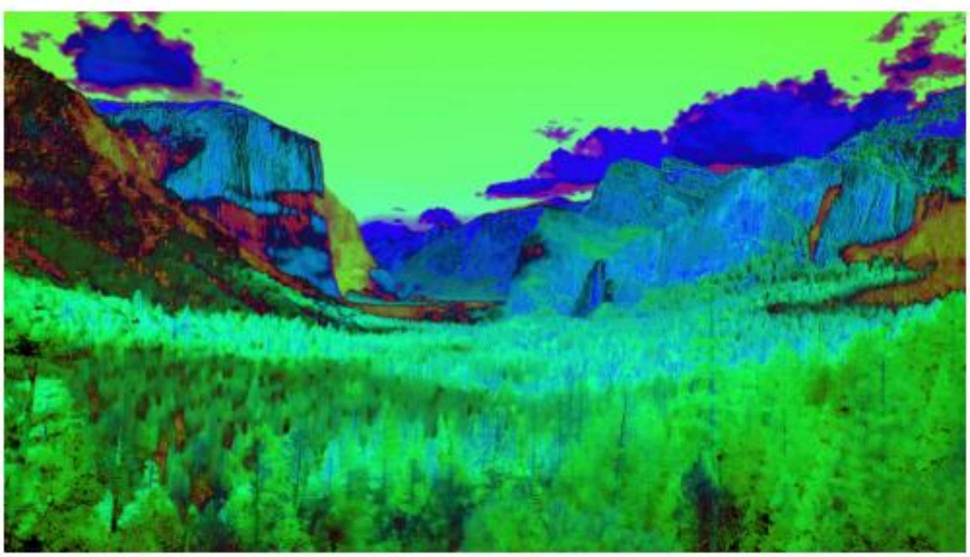


RGB TO HSV

```
#IMAGE 1
hsv_image = cv2.cvtColor(image1,cv2.COLOR_RGB2HSV)

plt.imshow(hsv_image)
plt.axis('off')
plt.show()
```



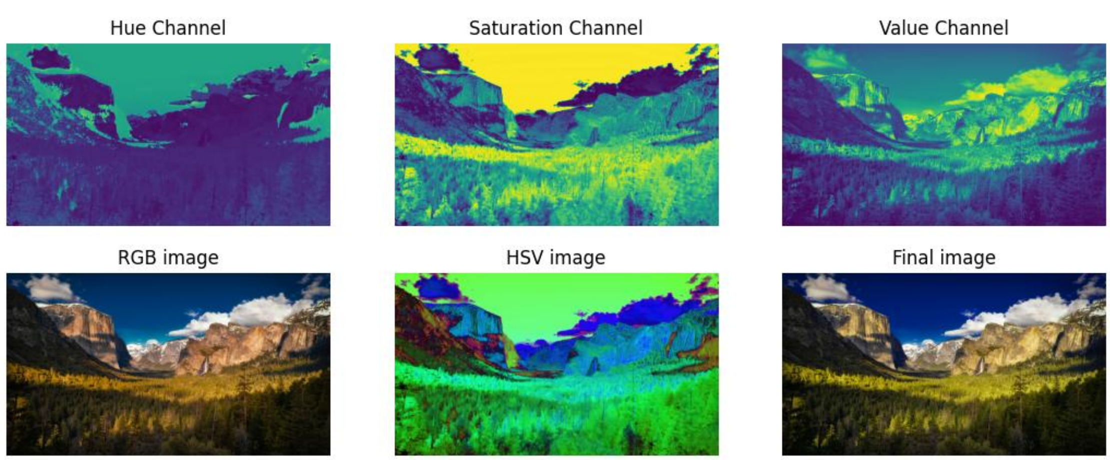
```
#IMAGE 2
hsv_image2 = cv2.cvtColor(image2,cv2.COLOR_RGB2HSV)

plt.imshow(hsv_image2)
plt.axis('off')
plt.show()
```



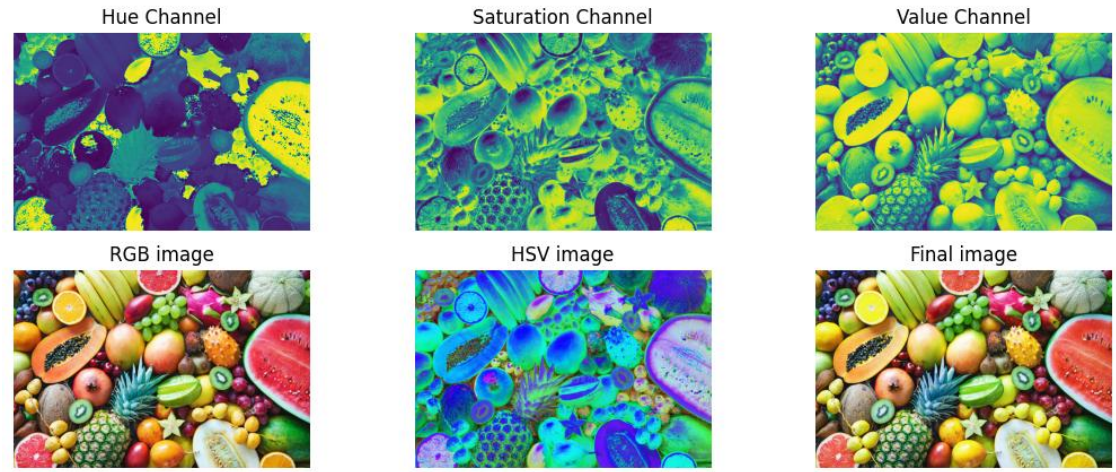
HSV Individual Channels

```
#IMAGE 1
img = cv2.imread("image1.jpg")
```

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)
h, s, v = cv2.split(img_hsv)

h += 7
s += 0

img_adj = cv2.merge([h,s,v])
imgf = cv2.cvtColor(img_adj, cv2.COLOR_HSV2RGB)
plt.figure(figsize=(13,5))
plt.subplot(2, 3, 1)
plt.imshow(h)
plt.title("Hue Channel")
plt.axis('off')
plt.subplot(2, 3, 2)
plt.imshow(s)
plt.title("Saturation Channel")
plt.axis('off')
plt.subplot(2, 3, 3)
plt.imshow(v)
plt.title("Value Channel")
plt.axis('off')
plt.subplot(2, 3, 4)
plt.imshow(img_rgb)
plt.title("RGB image")
plt.axis('off')
plt.subplot(2, 3, 5)
plt.imshow(img_hsv)
plt.title("HSV image")
plt.axis('off')
plt.subplot(2, 3, 6)
plt.imshow(imgf)
plt.title("Final image")
plt.axis('off')
plt.show()
```



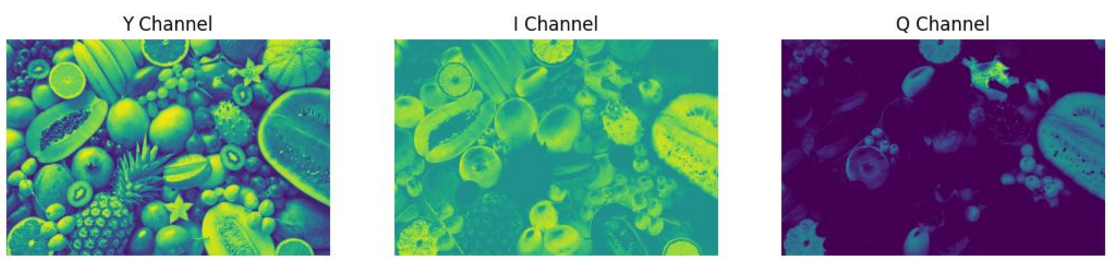
```python
#IMAGE 2
img = cv2.imread("FRUITS.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)
h, s, v = cv2.split(img_hsv)

h += 7
s += 0

img_adj = cv2.merge([h,s,v])
imgf = cv2.cvtColor(img_adj, cv2.COLOR_HSV2RGB)
plt.figure(figsize=(13,5))
plt.subplot(2, 3, 1)
plt.imshow(h)
plt.title("Hue Channel")
plt.axis('off')
plt.subplot(2, 3, 2)
plt.imshow(s)
plt.title("Saturation Channel")
plt.axis('off')
plt.subplot(2, 3, 3)
plt.imshow(v)
plt.title("Value Channel")
plt.axis('off')
plt.subplot(2, 3, 4)
plt.imshow(img_rgb)
plt.title("RGB image")
plt.axis('off')
plt.subplot(2, 3, 5)
plt.imshow(img_hsv)
plt.title("HSV image")
plt.axis('off')
plt.subplot(2, 3, 6)
plt.imshow(imgf)
plt.title("Final image")
plt.axis('off')
plt.show()
```



RGB TO YIQ

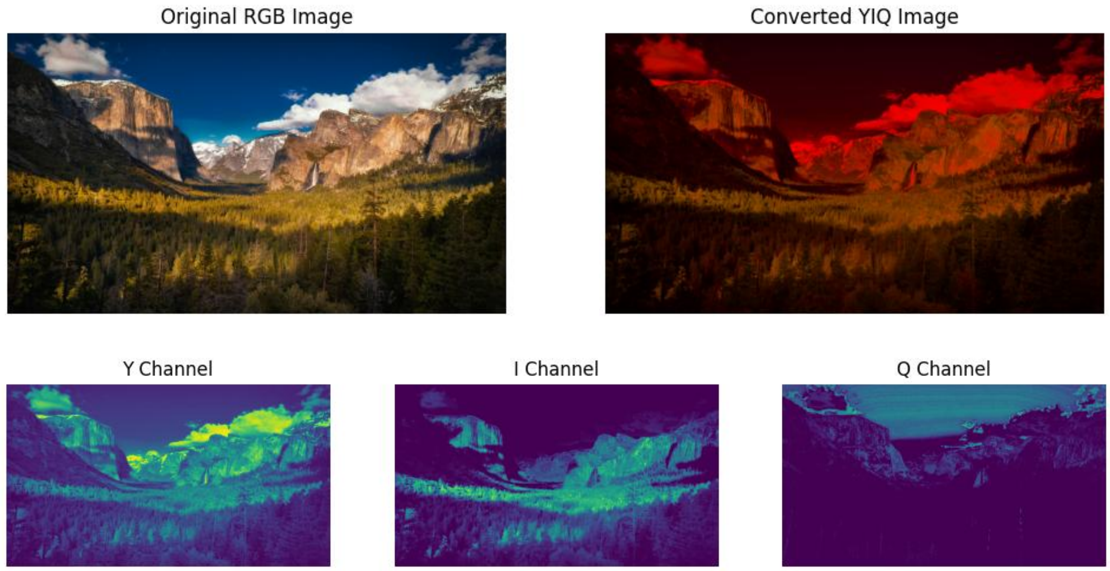
```
#IMAGE 1
image_rgb = image1 / 255.0

transformation_matrix = np.array([
        [0.299,  0.587,  0.114],
        [0.5957, -0.2746, -0.3213],
        [0.2115, -0.5227,  0.3113]])

reshaped_image = image_rgb.reshape(-1, 3)
yiq_image = reshaped_image.dot(transformation_matrix.T)
yiq_image = yiq_image.reshape(image_rgb.shape)
yiq_image = np.clip(yiq_image * 255, 0, 255).astype(np.uint8)

plt.figure(figsize=(12,10))

plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title("Original RGB Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(yiq_image)
plt.title("Converted YIQ Image")
plt.axis('off')

plt.show()

Y = yiq_image[:, :, 0]
I = yiq_image[:, :, 1]
Q = yiq_image[:, :, 2]

I = np.clip(I + 128, 0, 255)
```

```
Q = np.clip(Q + 128, 0, 255)

plt.figure(figsize=(13,7))

plt.subplot(2, 3, 1)
plt.imshow(Y)
plt.title("Y Channel")
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(I)
plt.title("I Channel")
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(Q)
plt.title("Q Channel")
plt.axis('off')

plt.show()
```




```
#IMAGE 2
image_rgb = image2 / 255.0

transformation_matrix = np.array([
        [0.299,  0.587,  0.114],
        [0.5957, -0.2746, -0.3213],
        [0.2115, -0.5227,  0.3113]])
```

```
reshaped_image = image_rgb.reshape(-1, 3)
yiq_image = reshaped_image.dot(transformation_matrix.T)
yiq_image = yiq_image.reshape(image_rgb.shape)
yiq_image = np.clip(yiq_image * 255, 0, 255).astype(np.uint8)

plt.figure(figsize=(12,10))

plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title("Original RGB Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(yiq_image)
plt.title("Converted YIQ Image")
plt.axis('off')

plt.show()

Y = yiq_image[:, :, 0]
I = yiq_image[:, :, 1]
Q = yiq_image[:, :, 2]

I = np.clip(I + 128, 0, 255)
Q = np.clip(Q + 128, 0, 255)

plt.figure(figsize=(13,7))

plt.subplot(2, 3, 1)
plt.imshow(Y)
plt.title("Y Channel")
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(I)
plt.title("I Channel")
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(Q)
plt.title("Q Channel")
plt.axis('off')

plt.show()
```



INFERENCE

Had understood the concepts of color model and the effect on their change and splitting with the help of library open-cv

RESULT

Thus, the concepts of change of color model and splitting executed successfully for sample images.