Assignment 3

1. Find the formula for mapping among these color models
2. Write a program for converting an image to different color domains. Also, present image in each channel
3. What is LAB color domain and write the program for presenting each channel in LAB domain

CMYK

For converting BGR to CMYK format, I use the formula

Black (K) = 1 - ($\frac{max(R,G,B)}{255}$)

Cyan (C) = $\frac{1-R'-K}{1-K}$

Magenta (M) = $\frac{1-G'-K}{1-K}$

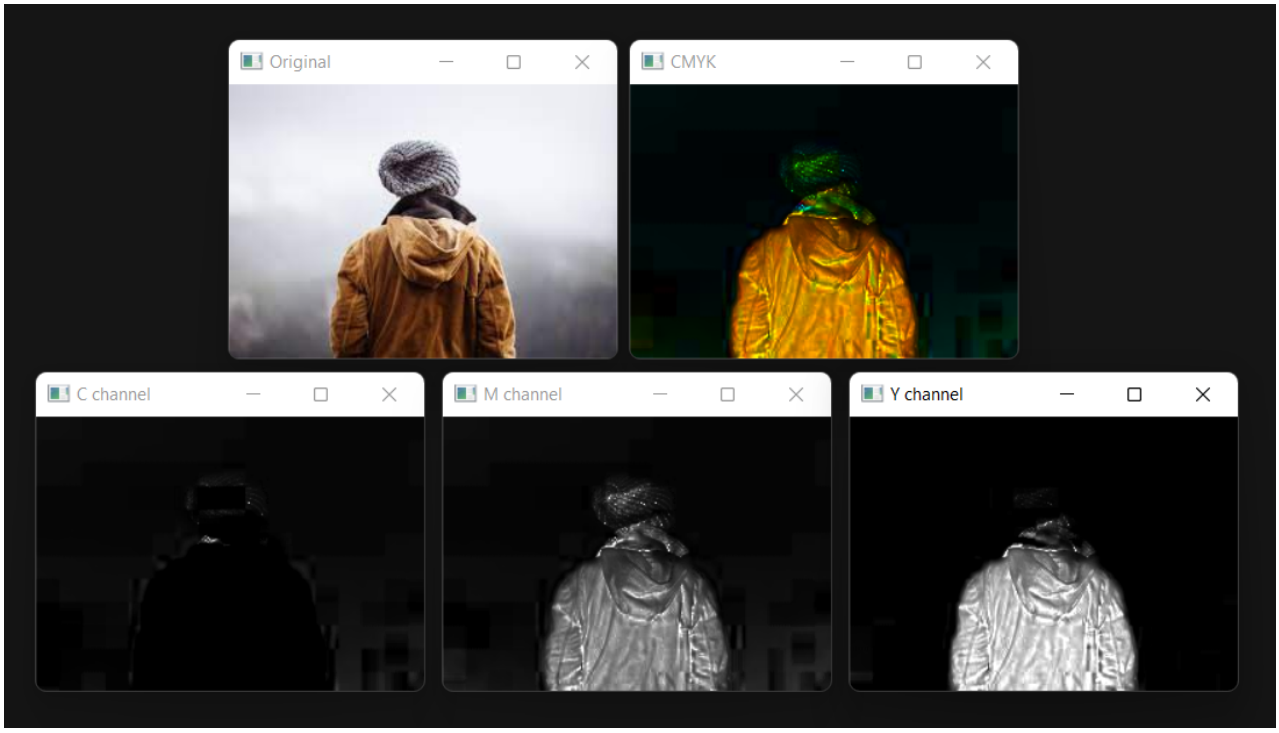
Yellow (Y) = $\frac{1-B'-K}{1-K}$

Where,

R' = $\frac{R}{255}$, G' = $\frac{G}{255}$, and B' = $\frac{B}{255}$

According to the formula, we can write Python as the coding below. The results in each channel are also shown below the coding by using imshow.

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img = cv.imread('images.jpg')
cv.imshow('Original',img)

# convert BGR to HSI by formula
b = img[:,:,0] / 255.0
g = img[:,:,1] / 255.0
r = img[:,:,2] / 255.0

k = 1-np.max(img/255, axis=2)
c = (1-r-k)/(1-k)
m = (1-g-k)/(1-k)
y = (1-b-k)/(1-k)

CMYK_image= (np.dstack((c,m,y,k)) * 255).astype(np.uint8)
cv.imshow("CMYK",CMYK_image)
cv.imshow("C channel",c)
cv.imshow("M channel",m)
cv.imshow("Y channel",y)
cv.waitKey(0)
```


Original
CMYK
C channel
M channel
Y channel

HSV

To convert the image to HSV domain, I follow the condition that

V = Max

When When Max = R and G≥B,

$$H = 60 \times \frac{G-B}{Max-Min} + 0$$

When Max = R and G<B,

$$H = 60 \times \frac{G-B}{Max-Min} + 360$$

When Max = G,

$$H = 60 \times \frac{B-R}{Max-Min} + 120$$

When Max = B,

$$H = 60 \times \frac{R-G}{Max-Min} + 240$$

When Max = 0,

S = 0

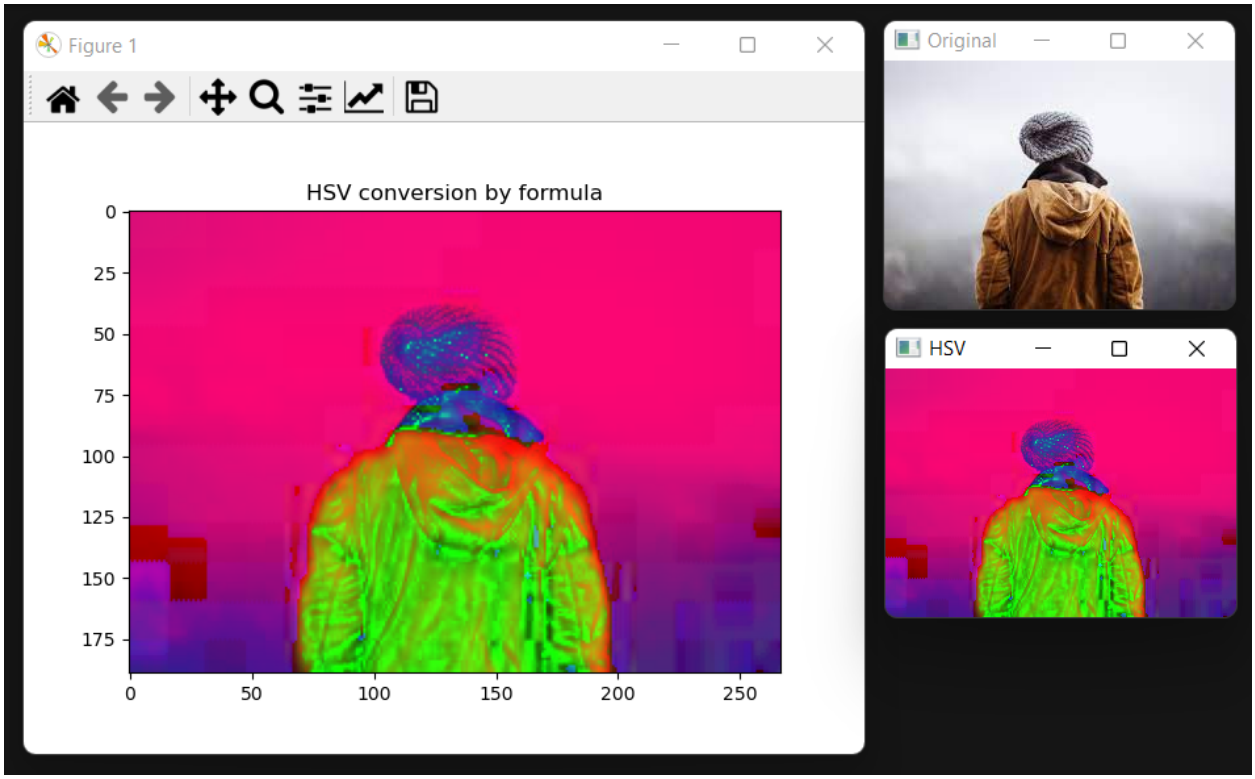
Otherwise,

$$S = 1 - \frac{Min}{Max}$$

Then, write them in the format of Python. Finally, I compare the result which uses the formulas to the result using the openCV library as well.

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
img = cv.imread('images.jpg')
cv.imshow('Original',img)

# convert RGB to HSV by openCV
hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)
cv.imshow("HSV", hsv_img)

# convert RGB to HSV by formula
def hsv_conversion(r, g, b):
    r, g, b = r/255, g/255, b/255
    mx = np.max([r, g, b])
    mn = np.min([r, g, b])
    df = mx-mn
    if mx == mn:
        h = 0
    elif mx == r:
        h = (60 * ((g-b)/df) + 360) % 360
    elif mx == g:
        h = (60 * ((b-r)/df) + 120) % 360
    elif mx == b:
        h = (60 * ((r-g)/df) + 240) % 360
    if mx == 0:
        s = 0
    else:
        s = (df/mx)*255
    v = mx*255

    return [h/2, s, v]
```

```python
h = np.zeros(img.shape[:2])
s = np.zeros(img.shape[:2])
v = np.zeros(img.shape[:2])

row, col = img.shape[0], img.shape[1]
for i in range (0,row):
    h1 = []
    s1 = []
    v1 = []
    for j in range (0,col):
        r = img[i,j][2]
        g = img[i,j][1]
        b = img[i,j][0]
        hsv = hsv_conversion(r,g,b)
        h1.append(hsv[0])
        s1.append(hsv[1])
        v1.append(hsv[2])
    h[i] = h1
    s[i] = s1
    v[i] = v1

hsv = np.round(cv.merge((v,s,h))).astype(np.int)
plt.imshow(hsv)
plt.title("HSV conversion by formula")
plt.show()
```



LAB

The LAB color space identifies 3 components of image which are lightness(L), color ranging from Green to Magenta (a), and color ranging from Blue to Yellow (b). Therefore, the L channel is independent and has duty on the brightness only while others are about the color responsibility.

To make the BGR image to LAB space, I use the formula that

In case of 8-bit and 16-bit images, R, G, and B are converted to the floating-point format and scaled to fit the 0 to 1 range.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow X/X_n, \text{where} X_n = 0.950456$$

$$Z \leftarrow Z/Z_n, \text{where} Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 * Y^{1/3} - 16 & \text{for } Y > 0.008856 \\ 903.3 * Y & \text{for } Y \leq 0.008856 \end{cases}$$

$$a \leftarrow 500(f(X) - f(Y)) + delta$$

$$b \leftarrow 200(f(Y) - f(Z)) + delta$$

where

$$f(t) = \begin{cases} t^{1/3} & \text{for } t > 0.008856 \\ 7.787t + 16/116 & \text{for } t \leq 0.008856 \end{cases}$$

and

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 0 & \text{for floating-point images} \end{cases}$$

This outputs $0 \leq L \leq 100$, $-127 \leq a \leq 127$, $-127 \leq b \leq 127$. The values are then converted to the destination data type:

- 8-bit images: $L \leftarrow L * 255/100,\ a \leftarrow a + 128,\ b \leftarrow b + 128$
- 16-bit images: (currently not supported)
- 32-bit images: L, a, and b are left as is

Figure from https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

Then, coding the above condition into Python and plotting the result and comparing it to the original image and LAB image which uses the library from openCV.

```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img = cv.imread('images.jpg')
cv.imshow('Original',img)

LAB = cv.cvtColor(img,cv.COLOR_BGR2LAB)
cv.imshow('LAB from openCV',LAB)

img = img.astype(np.uint8)

b = img[:,:,0] / 255
g = img[:,:,1] / 255
r = img[:,:,2] / 255
x = ((0.412453*r) + (0.357580*g) + (0.180423*b)) / 0.950455
y = ((0.212671*r) + (0.715160*g) + (0.072169*b)) / 1.0
z = ((0.019334*r) + (0.119193*g) + (0.950227*b)) / 1.088753

if y.any() > 0.008856:
    y1 = np.power(y,1/3)
    L = ((116*y1)-16)*(255/100)
else:
    L = (903.3*y)*(255/100)

def equation(a):
    if a.any() > 0.008856:
        a = np.power(a,1/3)
    else:
        a = (7.787*a) + (16/116)
    return a
```

```python
A = (500*(equation(x) - equation(y))) + 128
B = (200*(equation(y) - equation(z))) + 128

LAB1 = np.dstack((B,A,L)).astype(np.uint8)
plt.imshow(LAB1)
plt.title("LAB conversion by formula")
plt.show()
```

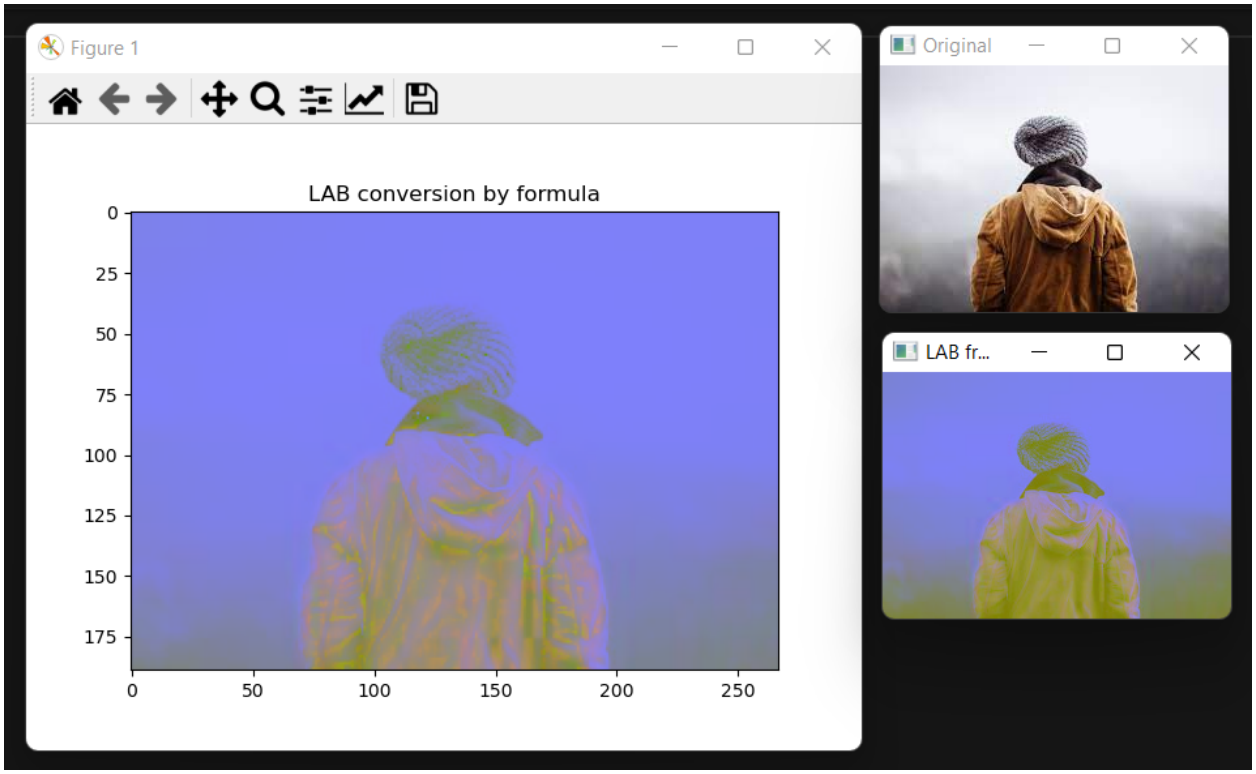


Figure 1
LAB conversion by formula
0
25
50
75
100
125
150
175
0
50
100
150
200
250
Original
LAB fr...