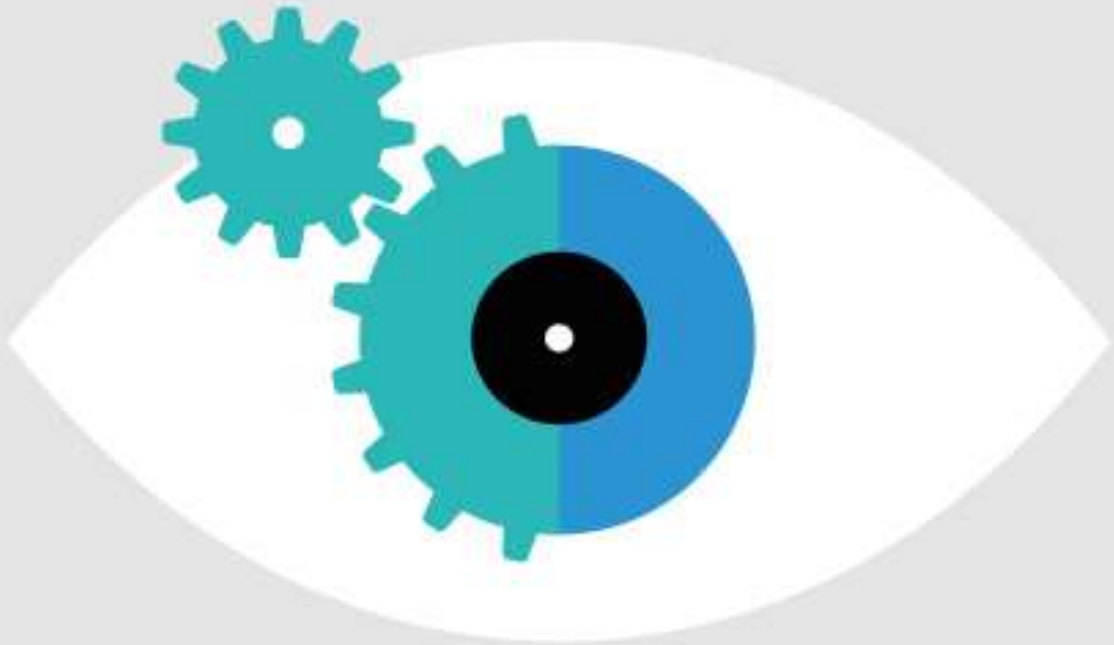


Computer Vision

THỊ GIÁC MÁY TÍNH

ThS. Huỳnh Minh Vũ
Khoa Kỹ thuật cơ khí
Trường Đại học Kỹ thuật – Công nghệ Cần Thơ
Email: hmvu@ctuet.edu.vn

Chương 2: Các phép xử lý đơn giản trong OpenCV

2.1 Đọc, hiển thị và lưu hình ảnh (video)

2.2 Không gian màu và chuyển đổi giữa các không gian màu

2.3 Các hàm vẽ trong OpenCV

2.4 Các thao tác cơ bản trên ảnh

2.5 Phép nhân tích chập (convolution)

2.1 Đọc, hiển thị và lưu hình ảnh (video)

Đọc ảnh ←

Hiển thị

Lưu ảnh và thoát

```
import cv2
img1 = cv2.imread('Lighthouse1.jpg')

print(img1.shape)   #xem kich thuoc anh
print(img1) #xem ma tran diem anh

cv2.imshow('Image', img1) #hien thi img

k = cv2.waitKey(0)

if k == 27: # bam ESC de thoat
   cv2.destroyAllWindows()

elif k == ord('s'): # bam 's' de luu và thoat
   img2 = cv2.imwrite('Lighthouse2.png', img1) #tao anh moi
   cv2.destroyAllWindows()
```

2.1 Đọc, hiển thị và lưu hình ảnh (video)

Đọc video

Hiển thị

Thoát

```
import cv2

cap = cv2.VideoCapture(IP)

while(True):
    ret, frame = cap.read()
    #frame = cap.read()[1]

    #Doan nay se chen thuat toan

    cv2.imshow('frame',frame)

    if cv2.waitKey(1) == ord('q'):
      break

cap.release()
cv2.destroyAllWindows()
```

IP = 0
Đọc Webcam laptop

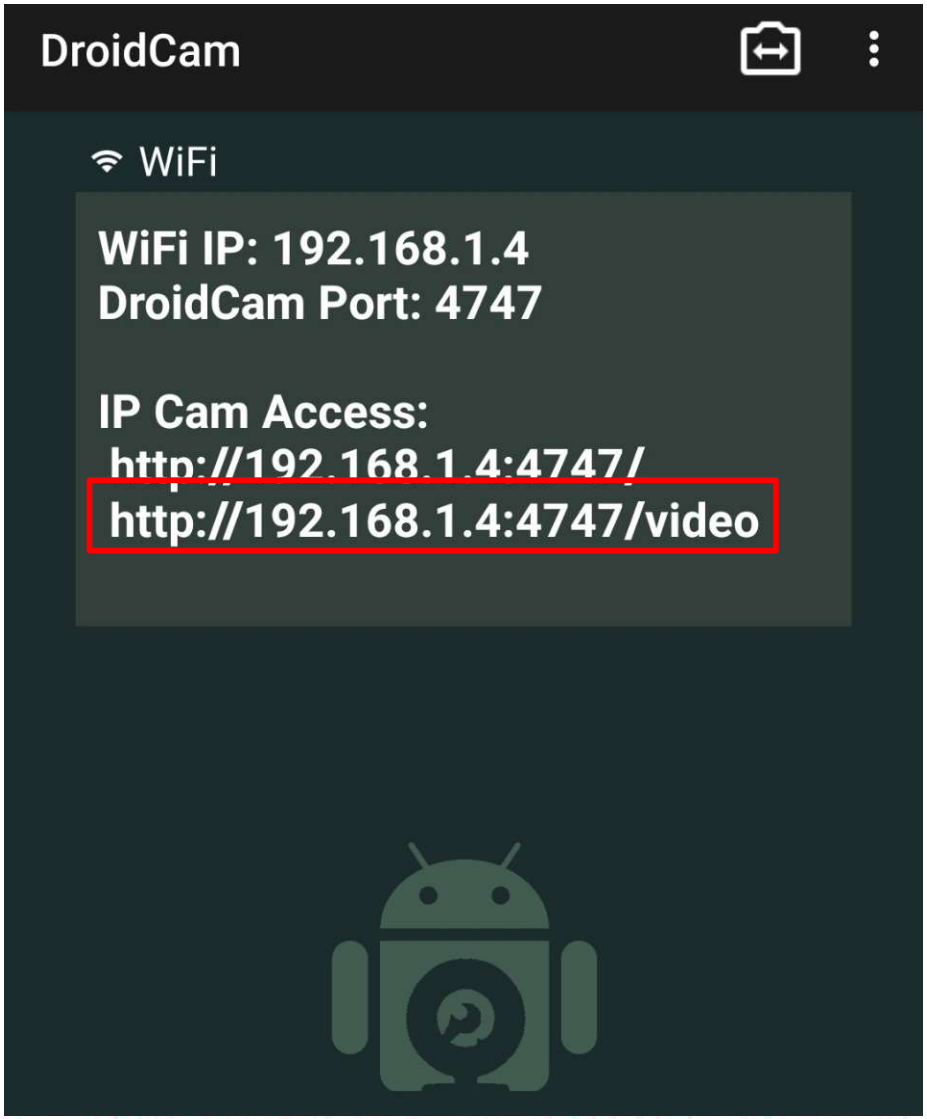
IP = 1
Đọc Webcam USB

IP = 'Video.mp4'
Đọc Video

IP = 'Địa chỉ IP'
Đọc Camera IP

2.1 Đọc, hiển thị và lưu hình ảnh (video)

IP = 'Địa chỉ IP'
Đọc Camera IP




```
cap = cv2.VideoCapture('http://192.168.1.4:4747/video')
```

2.1 Đọc, hiển thị và lưu hình ảnh (video)

Kết nối camera công nghiệp Basler ace3800 – 14uc

```
from pypylon import pylon
import cv2
import imutils as im

camera = pylon.InstantCamera(pylon.TlFactory.GetInstance().CreateFirstDevice())
camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)

converter = pylon.ImageFormatConverter()
converter.OutputPixelFormat = pylon.PixelType_BGR8packed
converter.OutputBitAlignment = pylon.OutputBitAlignment_MsbAligned
```

2.1 Đọc, hiển thị và lưu hình ảnh (video)

Kết nối camera công nghiệp Basler ace3800 – 14uc

```
while True:
    grabResult = camera.RetrieveResult(5000, pylon.TimeoutHandling_ThrowException)
    if grabResult.GrabSucceeded():
        image = converter.Convert(grabResult)
        fra_cam = image.GetArray()
        #print(fra_cam.shape) (2748, 3840, 3)
        frame = im.resize(fra_cam, width=640, height=480)
        ##################################################
        #code here
        ##################################################
        cv2.imshow('Frame',frame)
    k = cv2.waitKey(1)
    if k == ord('q'):
        break
        camera.close()
cv2.destroyAllWindows()
```

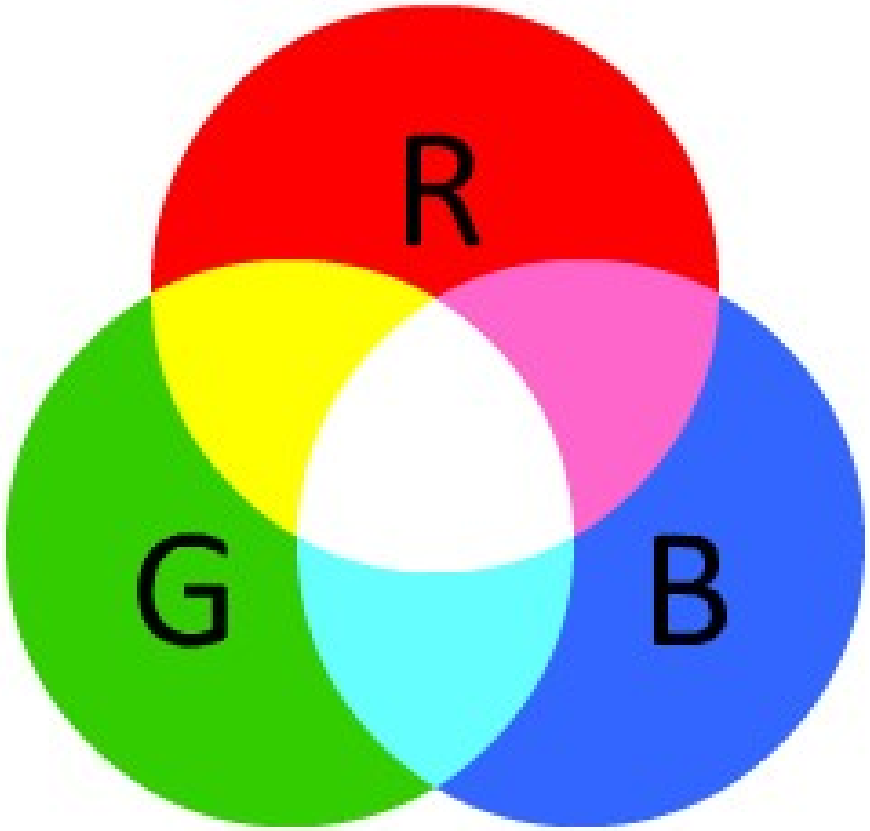
2.2 Không gian màu và chuyển đổi

- Không gian màu là một mô hình toán học dùng để mô tả các màu trong thực tế được biểu diễn dưới dạng số học.
- Trong thực tế của nhiều không gian màu khác nhau để sử dụng cho các mục đích khác nhau.
- Có 3 không gian màu được ứng dụng phổ biến là:
  - RGB
  - HSV
  - CMY

2.2 Không gian màu và chuyển đổi

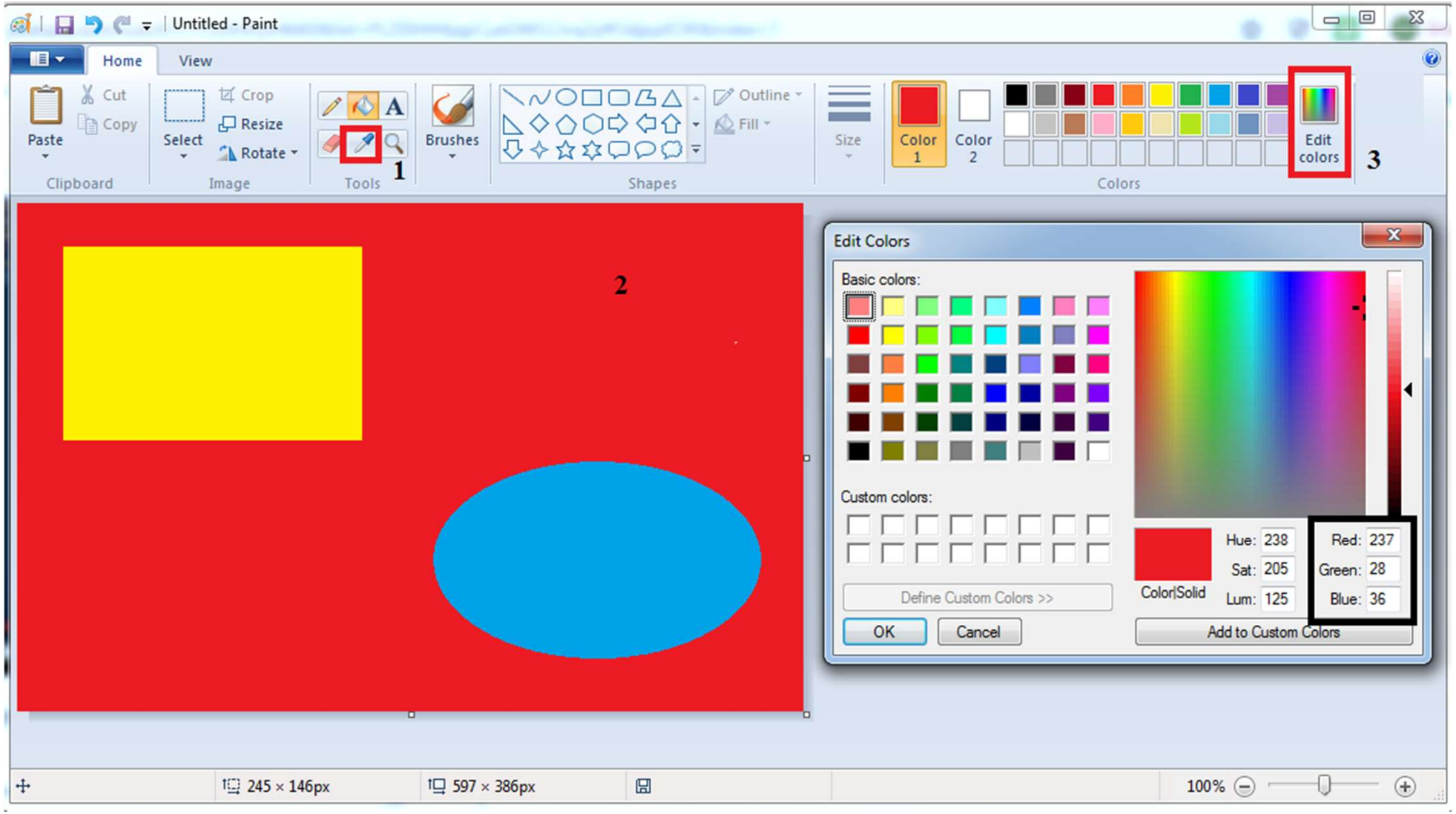
Không gian màu RGB

- RGB là không gian màu rất phổ biến được dùng trong đồ họa máy tính và nhiều thiết bị kĩ thuật số khác.
- Ý tưởng chính của không gian màu này là sự **kết hợp** của 3 màu sắc cơ bản: màu đỏ (R, Red), xanh lục (G, Green) và xanh dương (B, Blue) để mô tả tất cả các màu sắc khác.



2.2 Không gian màu và chuyển đổi

Xem thông số của không gian màu RGB bằng Paint

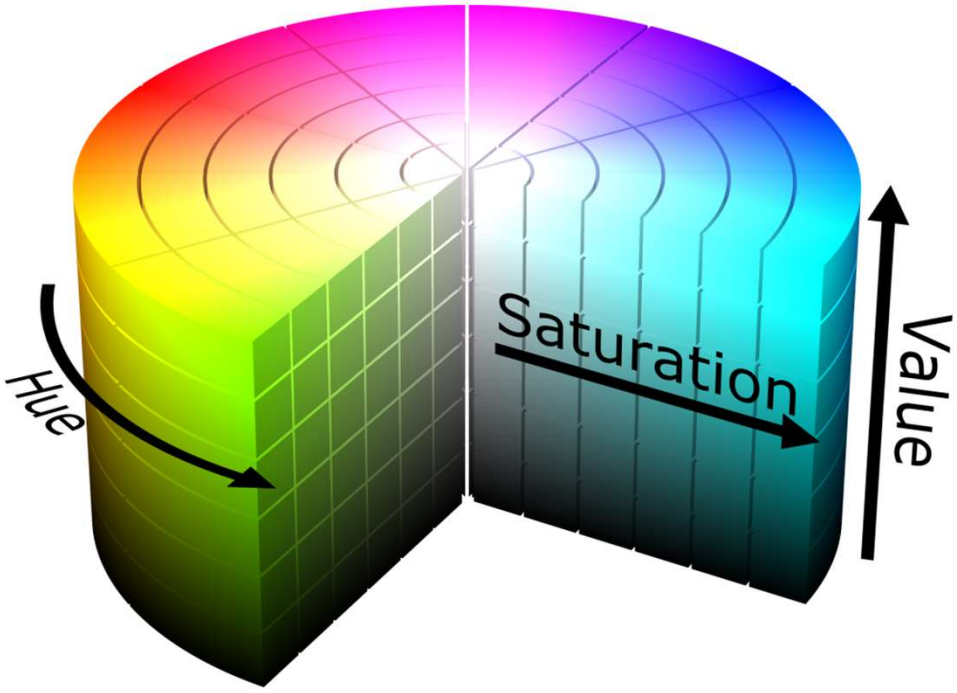


2.2 Không gian màu và chuyển đổi

Không gian màu HSV

HSV là một không gian màu dựa trên 3 thông số chính của không gian màu:

- **H**UE có nghĩa là vùng màu.
- **S**ATURATION có nghĩa là độ bảo hòa màu (độ đậm đặc).
- **V**ALUE có nghĩ là giá trị hay độ sáng của màu sắc.

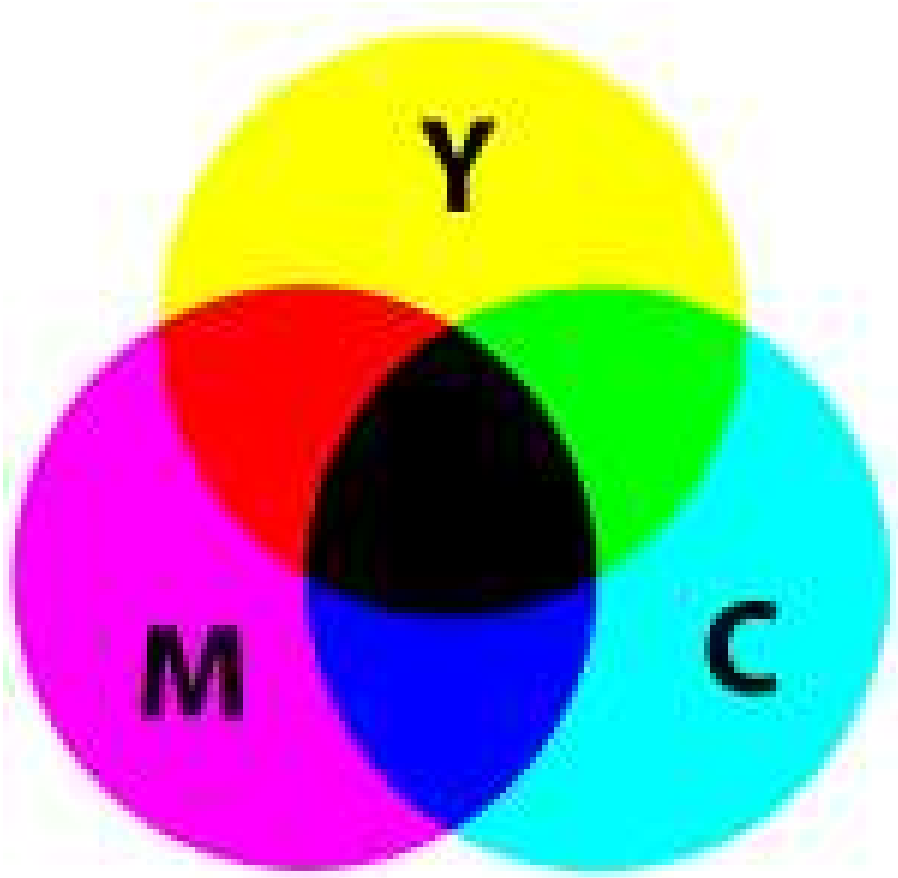


2.2 Không gian màu và chuyển đổi

Không gian màu CMY

CMY được tạo ra từ việc **trừ ánh sáng trắng** với 3 màu: xanh lam (C, Cyan), hồng (M, Magenta) và vàng (Y, Yellow).

- Được sử dụng phổ biến trong ngành in ấn.
- Trong máy in thường sử dụng thêm thành phần mực đen (K) do màu đen được tạo bằng cách phối hợp các màu CMY thực sự không được quá đậm.
- Như vậy, có thể gọi là không gian màu CMYK.



2.2 Không gian màu và chuyển đổi

Chuyển đổi từ không gian màu BGR sang HSV

$$V \leftarrow max(R, G, B)$$

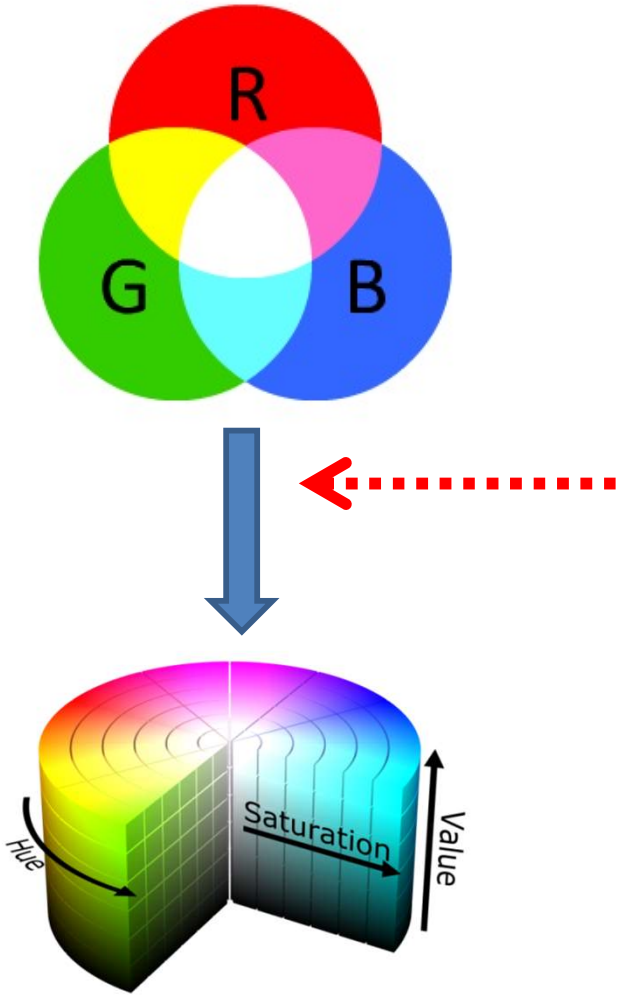
$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - min(R, G, B)) & \text{if } V = B \end{cases}$$

If $H < 0$ then $H \leftarrow H + 360$. On output $0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 360$.

2.2 Không gian màu và chuyển đổi

Chuyển đổi từ không gian màu BGR sang HSV




```
import cv2

img = cv2.imread('Color.png')
print('Gia tri RGB')
print(img)

##chuyen sang khong gian mau HSV
img_hsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

print('Gia tri HSV')
print(img_hsv)

cv2.imshow('Color RGB',img) #hien thi img
cv2.imshow('Color HSV',img_hsv) #hien thi img hsv

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.2 Không gian màu và chuyển đổi

Chuyển đổi từ không gian màu BGR sang Gray

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

```
import cv2

img1 = cv2.imread('Lighthouse1.jpg')

img2=cv2.cvtColor(img1,cv2.COLOR_RGB2GRAY)

cv2.imshow('Image1', img1) #hien thi img1
cv2.imshow('Image2', img2) #hien thi img1

k = cv2.waitKey()

if k == 27: # bam ESC de thoat
   cv2.destroyAllWindows()
```

2.2 Không gian màu và chuyển đổi

Tách đối tượng có màu quy định (Phân đoạn ảnh màu)

- Xác định ma trận đối tượng màu cần tách trong không gian BGR.
- Chuyển ma trận màu của đối tượng đó sang không gian HSV.

```
import cv2
import numpy as np

img = np.uint8([[[0, 242, 255]]])

img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

print (img_hsv)
#[[[28, 255, 255]]]
```

2.2 Không gian màu và chuyển đổi

Tách đối tượng có màu quy định (Phân đoạn ảnh màu)

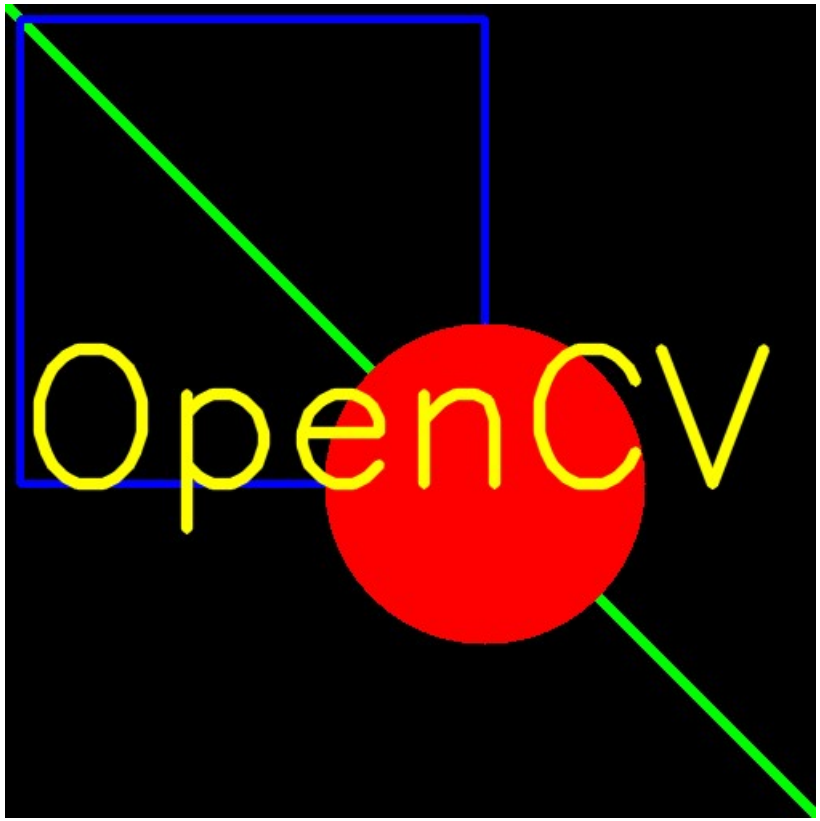
- Tạo mặt nạ để tách đối tượng màu quanh giá trị ma trận đối tượng màu cần tách trong không gian màu HSV.
- Cộng giá trị mặt nạ với ảnh trong không gian màu BGR.

```
import cv2
import numpy as np
img = cv2.imread('Color.png')
img_hsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
#chuyen sang KGM HSV
cv2.imshow('Color RGB',img) #hien thi img
cv2.imshow('Color HSV',img_hsv) #hien thi img hsv
#Gia  min, max xung quanh HSV
min_color = np.array([26, 250, 250])
max_color = np.array([30, 255, 255])

mask = cv2.inRange(img_hsv, min_color, max_color)
cv2.imshow('Mask', mask)
final = cv2.bitwise_and(img, img, mask = mask)
cv2.imshow('Final', final)
cv2.waitKey()
cv2.destroyAllWindows()
```

2.3 Các hàm vẽ trong OpenCV

Một số plot function hay sử dụng trong OpenCV:

cv2.line(), cv2.circle() , cv2.rectangle(), cv2.putText() etc.




```
import numpy as np
import cv2
img = np.zeros((512,512,3), np.uint8)
img = cv2.line(img,(0,0),(511,511),(0,255,0),5)
img = cv2.rectangle(img,(10,10),(300,300),(255,0,0),3)
img = cv2.circle(img,(300,300), 100, (0,0,255), -1)
#ve chu OpenCV
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(10,300), font,
\4,(0,255,255),5,cv2.LINE_AA)
cv2.imshow('Plot', img)
cv2.imwrite('OutputPlot.jpg', img)
cv2.waitKey()
```

2.4 Các thao tác cơ bản trên ảnh

Resize (Scale)

Resize ảnh là chỉnh kích thước ảnh về kích thước mới (có thể giữ tỉ lệ ảnh ban đầu hoặc không).

```
import cv2
img = cv2.imread('ChimCanhCut.jpg')
print(img.shape)
#kich thuoc tuong doi
new_width = 800
new_height = 400
img_resized1 = cv2.resize(src=img, dsize=(new_width, new_height))
cv2.imwrite('OutputResize1.jpg', img_resized1)
print(img_resized1.shape)
#kich thuoc tuyet doi
fx = 0.5
fy = 1.0
img_resized2 = cv2.resize(src=img, dsize=None, fx=fx, fy=fy)
cv2.imwrite('OutputResize2.jpg', img_resized2)
print(img_resized2.shape)
```

2.4 Các thao tác cơ bản trên ảnh

Rotation (Xoay ảnh)

Rotation ảnh là xoay ảnh theo một góc nhất định.

Phép xoay ảnh theo một góc θ được thực hiện bằng phép biến đổi ma trận R:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Tuy nhiên, trong OpenCV cho phép ta có thể xoay với tỉ lệ và tâm xoay bất kì, khi đó ma trận xoay được biến đổi là:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

với:

$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

2.4 Các thao tác cơ bản trên ảnh

Rotation (Xoay ảnh)

Để việc xác định ma trận xoay ta sử dụng hàm **cv2getRotationMatrix2D(center, angle, scale)**. Hàm này nhận 3 tham số đầu vào: tâm xoay, góc xoay (ngược chiều kim đồng hồ) và hệ số tỷ lệ (phóng to, thu nhỏ).

```
import cv2
img = cv2.imread('ChimCanhCut.jpg')
num_width, num_height = img.shape[:2]
print(img.shape[:2])
print(num_width)
print(num_height)
rotation_matrix = cv2.getRotationMatrix2D((num_width/2, num_height/2), 45, 1)
img_rotation = cv2.warpAffine(img, rotation_matrix, (num_width, num_height))
cv2.imshow('Rotation', img_rotation)
cv2.waitKey()
```

2.4 Các thao tác cơ bản trên ảnh

Crop

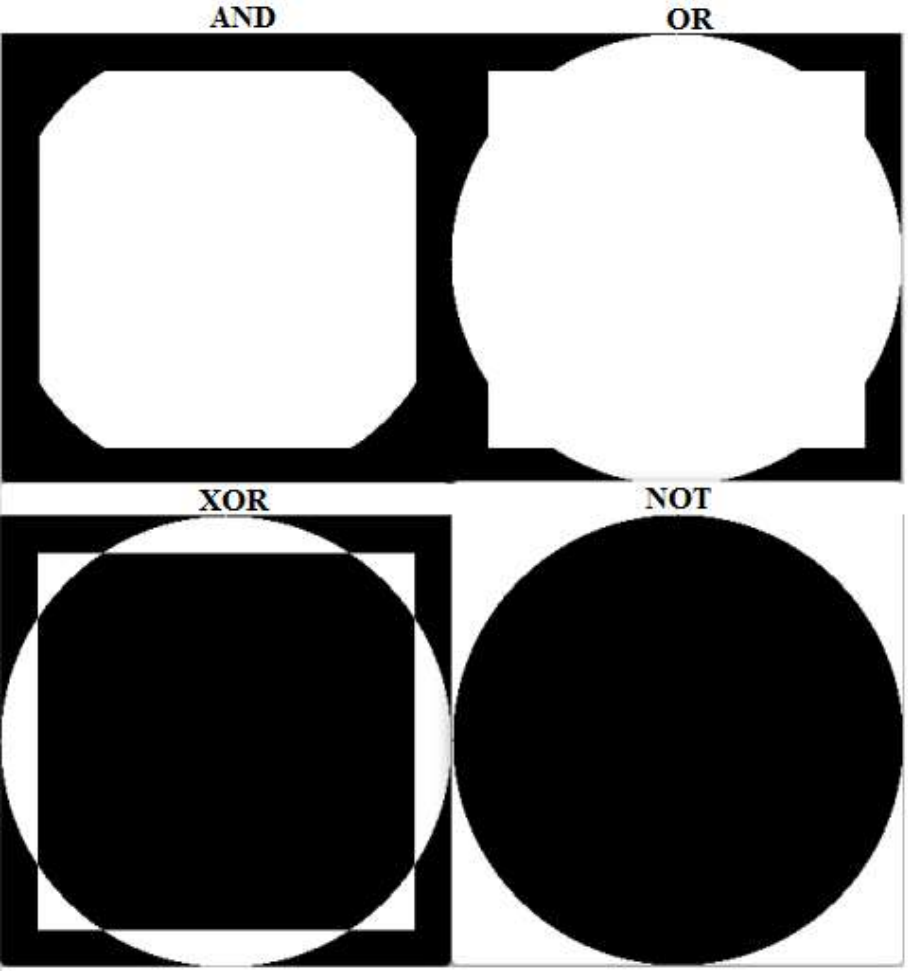
Crop ảnh là cắt một vùng ảnh theo kích thước nhất định.

```
import cv2
img = cv2.imread('ChimCanhCut.jpg')
print('ChimCanhCut.jpg', img.shape)
x = 100
y = 200
h =  200
w =  400
img_crop = img[y : y+h, x : x+w]
cv2.imwrite('OutCrop.jpg', img_crop)
print('OutCrop.jpg', img_crop.shape)
```

2.4 Các thao tác cơ bản trên ảnh

Bitwise

AND, OR, XOR, NOT

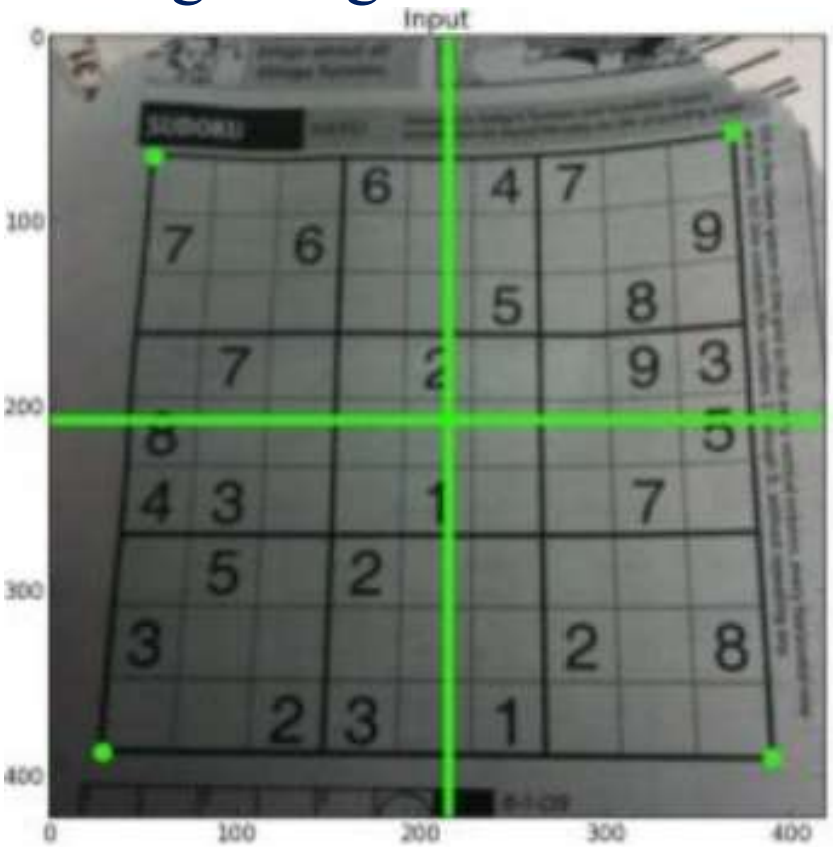


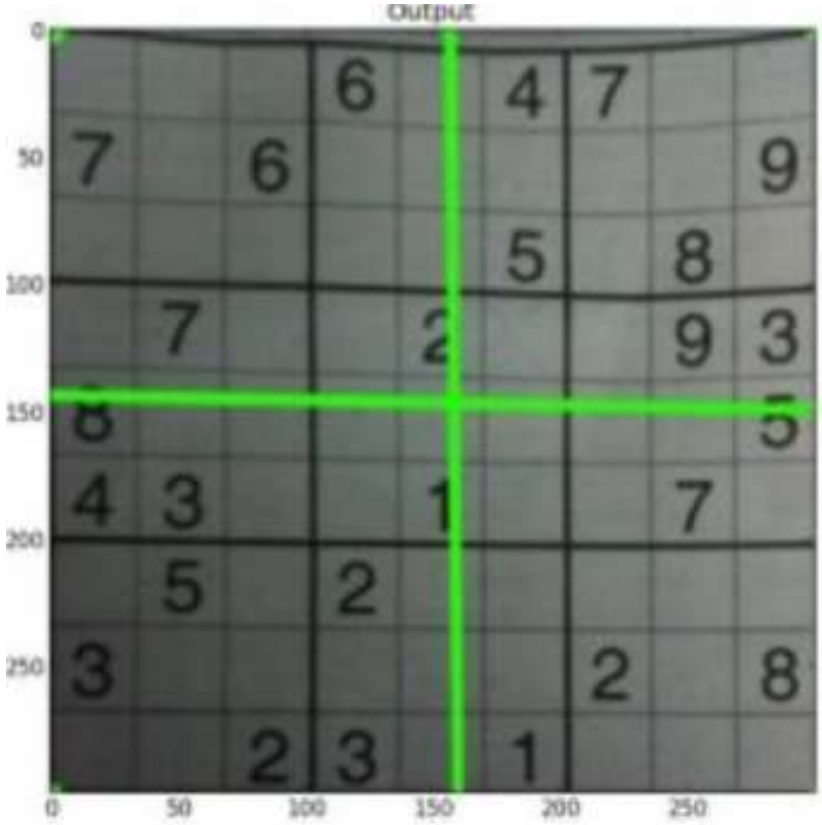

```
import numpy as np
import cv2
rectangle = np.zeros((300, 300), dtype = "uint8")
cv2.rectangle(rectangle, (25, 25), (275, 275), 255, -1)
circle = np.zeros((300, 300), dtype = "uint8")
cv2.circle(circle, (150, 150), 150, 255, -1)

bitwiseAnd = cv2.bitwise_and(rectangle, circle)
cv2.imshow("AND", bitwiseAnd)
cv2.waitKey(0)
bitwiseOr = cv2.bitwise_or(rectangle, circle)
cv2.imshow("OR", bitwiseOr)
cv2.waitKey(0)
bitwiseXor = cv2.bitwise_xor(rectangle, circle)
cv2.imshow("XOR", bitwiseXor)
cv2.waitKey(0)
bitwiseNot = cv2.bitwise_not(circle)
cv2.imshow("NOT", bitwiseNot)
cv2.waitKey(0)
```

2.4 Các thao tác cơ bản trên ảnh

Phép phối cảnh (Perspective)

Perspective transformation là phép biến đổi sao cho các điểm nằm trên một đường thẳng ở ảnh đầu vào vẫn nằm trên cùng một đường thẳng ở ảnh đầu ra.





2.4 Các thao tác cơ bản trên ảnh

Phép phối cảnh (Perspective)

Để xác định ma trận biến đổi này ta sử dụng hàm **cv2.getPerspectiveTransform**.

Hàm này nhận tham số đầu vào là vị trí của 4 điểm ở bức ảnh đầu vào và vị trí tương ứng của chúng ở bức ảnh đầu ra. Bốn điểm này được chọn sao cho bộ 3 điểm bất kỳ không nằm trên cùng một đường thẳng.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
img = cv2.imread('Sudoku.jpg')
pts1 = np.float32([[56, 65], [368, 52], [28, 387], [389, 390]])
pts2 = np.float32([[0, 0], [300, 0], [0, 300], [300, 300]])
M = cv2.getPerspectiveTransform(pts1, pts2)
dst = cv2.warpPerspective(img, M, (300, 300))
plt.subplot(121), plt.imshow(img), plt.title('Input')
plt.subplot(122), plt.imshow(dst), plt.title('Output')
plt.show()
```

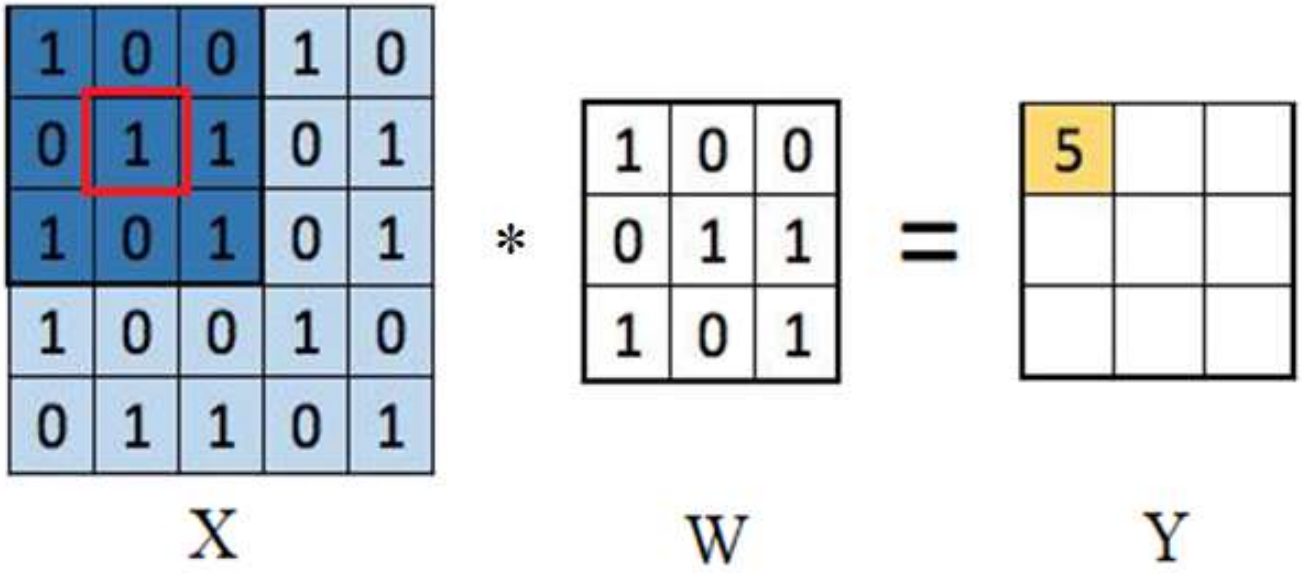
2.6 Phép nhân tích chập (convolution)

- Convolution là kỹ thuật quan trọng trong xử lý ảnh, được sử dụng chính yếu trong các phép toán trên ảnh như: đạo hàm ảnh, làm trơn ảnh, trích xuất biên cạnh trong ảnh.
- Định nghĩa convolution: Theo toán học, tích chập là phép toán tuyến tính, cho ra kết quả là một hàm bằng việc tính toán dựa trên hai hàm đã có.
- Định nghĩa kernel là một ma trận vuông kích thước (k x k) trong đó k là số lẻ (k có thể bằng 1, 3, 5, 7, 9,...).
- Ví dụ kernel kích thước 3x3:

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

2.6 Phép nhân tích chập (convolution)

- Kí hiệu phép tích chập *, kí hiệu Y = X * W

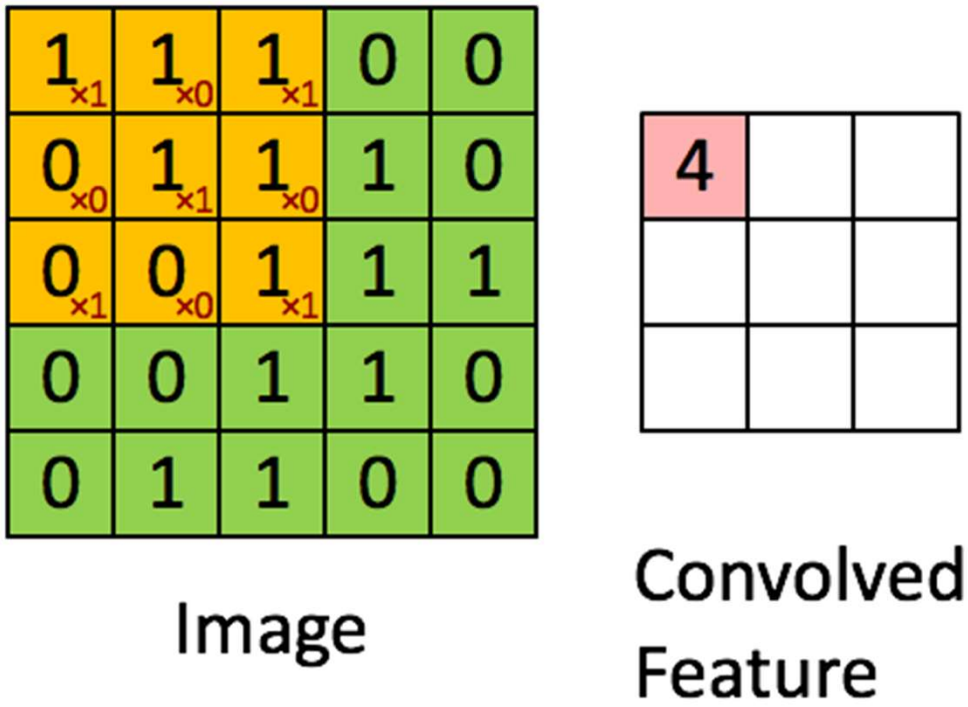


- Với mỗi phần tử xij trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử xij làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A (A ∈ X). Sau đó tính tổng các phần tử của phép tính nhân từng phần tử của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y.

2.6 Phép nhân tích chập (convolution)

- Ví dụ về phép convolution với ma trận W là:

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



TỔNG HỢP MỘT SỐ HÀM QUAN TRỌNG

cv.rectangle(img, pt1, pt2, color, thickness, lineType, shift)

Parameters

img	Image.
pt1	Vertex of the rectangle.
pt2	Vertex of the rectangle opposite to pt1 .
color	Rectangle color or brightness (grayscale image).
thickness	Thickness of lines that make up the rectangle.
lineType	Type of the line. See **LineTypes**
shift	Number of fractional bits in the point coordinates.

TỔNG HỢP MỘT SỐ HÀM QUAN TRỌNG

cv.circle(img, center, radius, color, thickness, lineType, shift)

Parameters

img	Image where the circle is drawn.
center	Center of the circle.
radius	Radius of the circle.
color	Circle color.
thickness	Thickness of the circle outline, if positive. Negative values, like **FILLED**, mean that a filled circle is to be drawn.
lineType	Type of the circle boundary. See **LineTypes**
shift	Number of fractional bits in the coordinates of the center and in the radius value.

TỔNG HỢP MỘT SỐ HÀM QUAN TRỌNG

cv.putText(img, text, org, fontFace, fontScale, color, thickness, lineType, bottomLeftOrigin)

Parameters

img	Image.
text	Text string to be drawn.
org	Bottom-left corner of the text string in the image.
fontFace	Font type, see **HersheyFonts**.
fontScale	Font scale factor that is multiplied by the font-specific base size.
color	Text color.
thickness	Thickness of the lines used to draw a text.
lineType	Line type. See **LineTypes**
bottomLeftOrigin	When true, the image data origin is at the bottom-left corner. Otherwise, it is at the top-left corner.

TỔNG HỢP MỘT SỐ HÀM QUAN TRỌNG

cv.getTrackbarPos(trackbarname, winname, value, count, onChange)

Parameters

trackbarname	Name of the created trackbar.
winname	Name of the window that will be used as a parent of the created trackbar.
value	Optional pointer to an integer variable whose value reflects the position of the slider.
count	Maximal position of the slider. The minimal position is always 0.
onChange	Pointer to the function to be called every time the slider changes position.

cv.getTrackbarPos(trackbarname, winname)

Parameters

trackbarname	Name of the trackbar.
winname	Name of the window that is the parent of the trackbar.

TỔNG HỢP MỘT SỐ HÀM QUAN TRỌNG

cv.inRange(src, lowerb, upperb)

Parameters

src first input array.

lowerb inclusive lower boundary array or a scalar.

upperb inclusive upper boundary array or a scalar.

cv.bitwise_and(src1, src2, mask)

Parameters

src1 first input array or a scalar.

src2 second input array or a scalar.

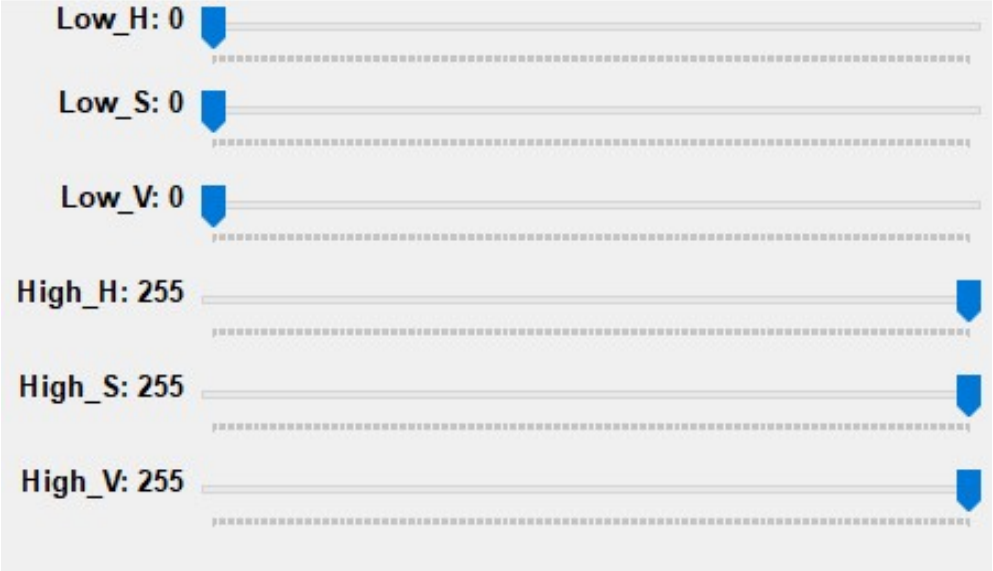
mask optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed.

CASE STUDY 2: TẠO CỬA SỔ TRƯỢT

Mô tả: Tạo cửa sổ trượt để xác định khoảng giá trị màu sắc của đối tượng trong không gian màu HSV.

Yêu cầu:

- Tạo cửa sổ trượt để xác định khoảng giá trị không gian màu HSV của đối tượng;
- Áp dụng khoảng giá trị đó để tạo mặt nạ phát hiện đối tượng.



Thank you !!!