

# Mì “Python”

---

Mì AI Training

Bài số 03



# Nội dung khóa học

- Bài 1. Python cơ bản A
- Bài 2. Python cơ bản B
- Bài 3. Python với OpenCV
- Bài 4. Python với Keras
- Bài 5. Python với Pandas
- Bài 6. Xây dựng Backend Server với Python

# Bài 3

- OpenCV là gì?
- Hàm đọc/ghi/hiển thị file ảnh
- Hàm convert màu file ảnh
- Hàm resize, rotate ảnh
- Hàm áp thresholding ảnh
- Hàm tìm cạnh trong ảnh
- Hàm làm mờ ảnh, giảm nhiễu ảnh
- Hàm tìm các contour trong ảnh
- Hàm face detection

# Hàm đọc ảnh

```
# import thư viện  
import cv2  
# Đọc ảnh từ file image.png  
image = cv2.imread("image.png")
```

True



# Hàm đọc ảnh

---

Có thể dùng thêm tham số để đặt chế độ đọc ảnh

```
# Đọc ảnh đen trắng  
image = cv2.imread("image.png", cv2.IMREAD_GRAYSCALE)
```



# Hàm đọc ảnh từ camera

---

```
# Mở camera  
cap = cv2.VideoCapture(camera_id)          (Camera_id = 0,1,...)  
# Đọc ảnh từ camera  
while(True):  
    ret, frame = cap.read()  
    if cv2.waitKey(1) & 0xFF == ord('q'):  
        break  
# Giải phóng camera  
cap.release()
```

# Hàm đọc ảnh từ video

---

```
# Mở video  
cap = cv2.VideoCapture(video_name)  
# Đọc ảnh từ camera  
while(cap.isOpened()):  
    ret, frame = cap.read()  
# Giải phóng camera  
cap.release()
```

(Tên file video.

Ví dụ: 'my\_video.mp4')

# Hàm lưu ảnh

---

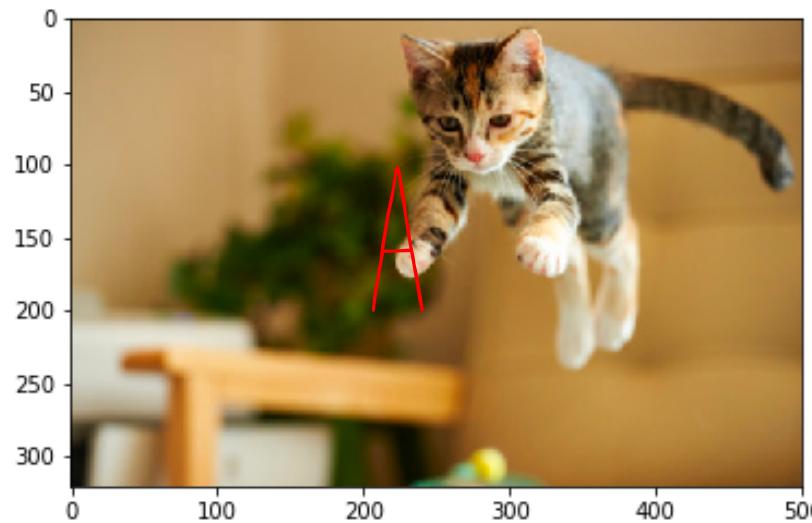
# Đọc ảnh từ file A (ảnh màu) -> đen trắng

```
image = cv2.imread("A.png", cv2.IMREAD_GRAYSCALE)
```

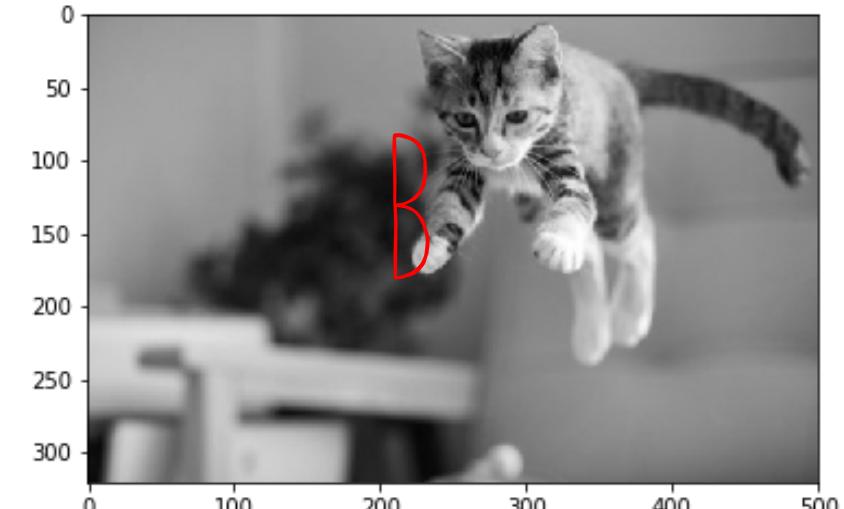
# Lưu ảnh vào file B

```
cv2.imwrite("B.png", image)
```

True



True



# Hệ màu trong OpenCV



BGR color profile



RGB color profile

- Mặc định OpenCV dùng hệ màu BGR
- Trong một số trường hợp cần convert sang các hệ màu khác để sử dụng.

# Hàm convert hệ màu

---

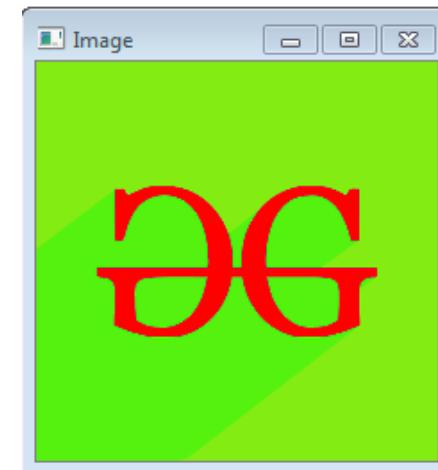
```
# Đọc ảnh từ file A  
image = cv2.imread("A.png")  
  
# Convert sang màu xám  
image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY )
```



# Hàm convert hệ màu

---

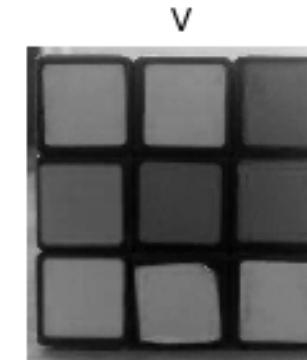
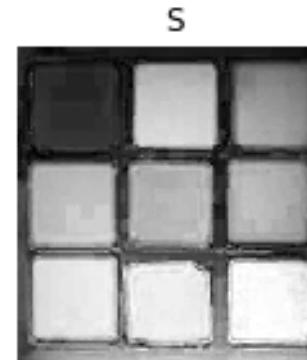
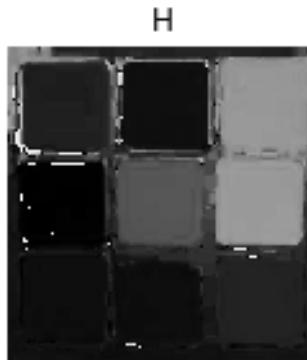
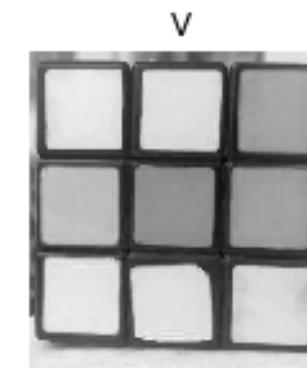
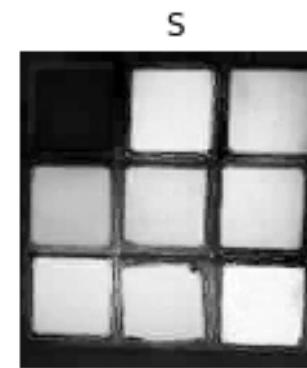
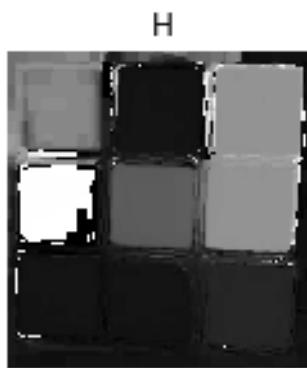
```
# Đọc ảnh từ file A  
image = cv2.imread("A.png")  
  
# Convert sang màu xám  
image = cv2.cvtColor(src, cv2.COLOR_BGR2HSV )
```



# Hàm convert hệ màu

---

Nói thêm về hệ màu HSV (Hue/Sat/Value)



# Hàm convert hệ màu

---

```
# Đọc ảnh từ file A
```

```
image = cv2.imread("A.png")
```

```
# Convert sang màu xám
```

```
image = cv2.cvtColor(src, cv2.COLOR_BGR2RGB )
```



BGR color profile

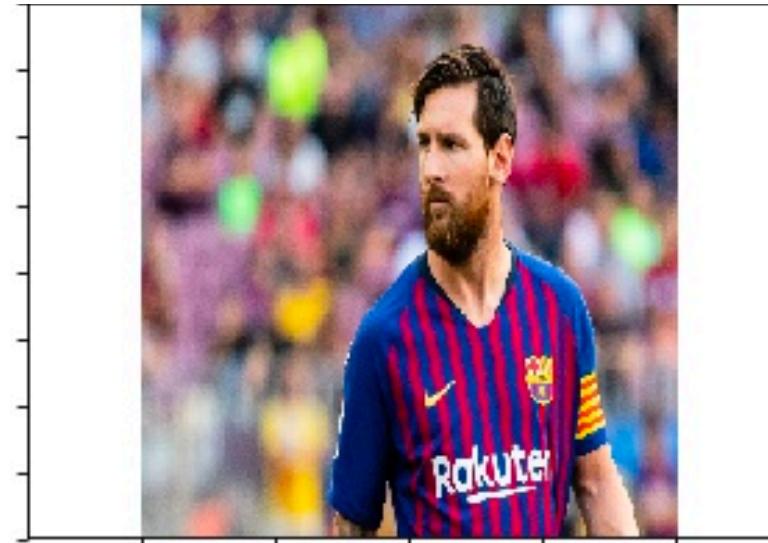
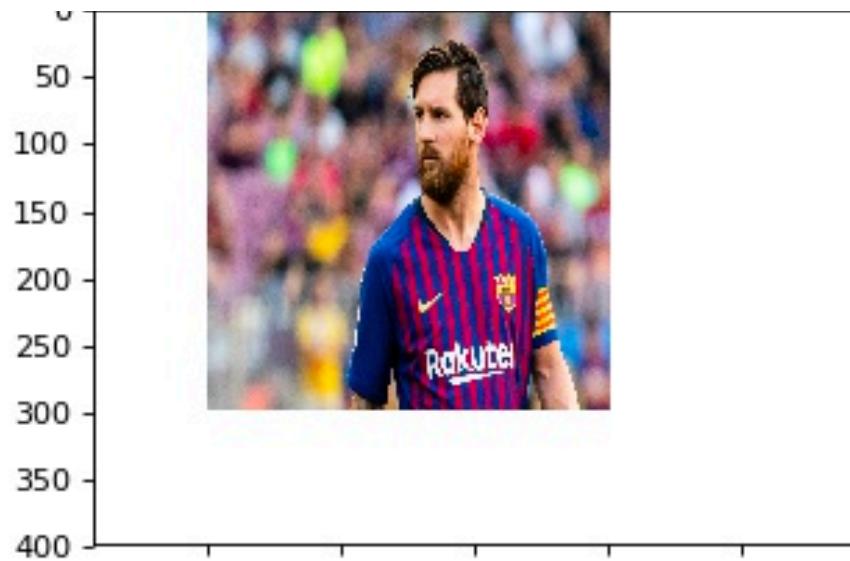


RGB color profile

# Hàm Resize ảnh

---

```
# Đọc ảnh  
image = cv2.imread('index.jpg')  
# Resize về size (100,100)  
smaller_image = cv2.resize(image,(100,100))
```



# Hàm Resize ảnh

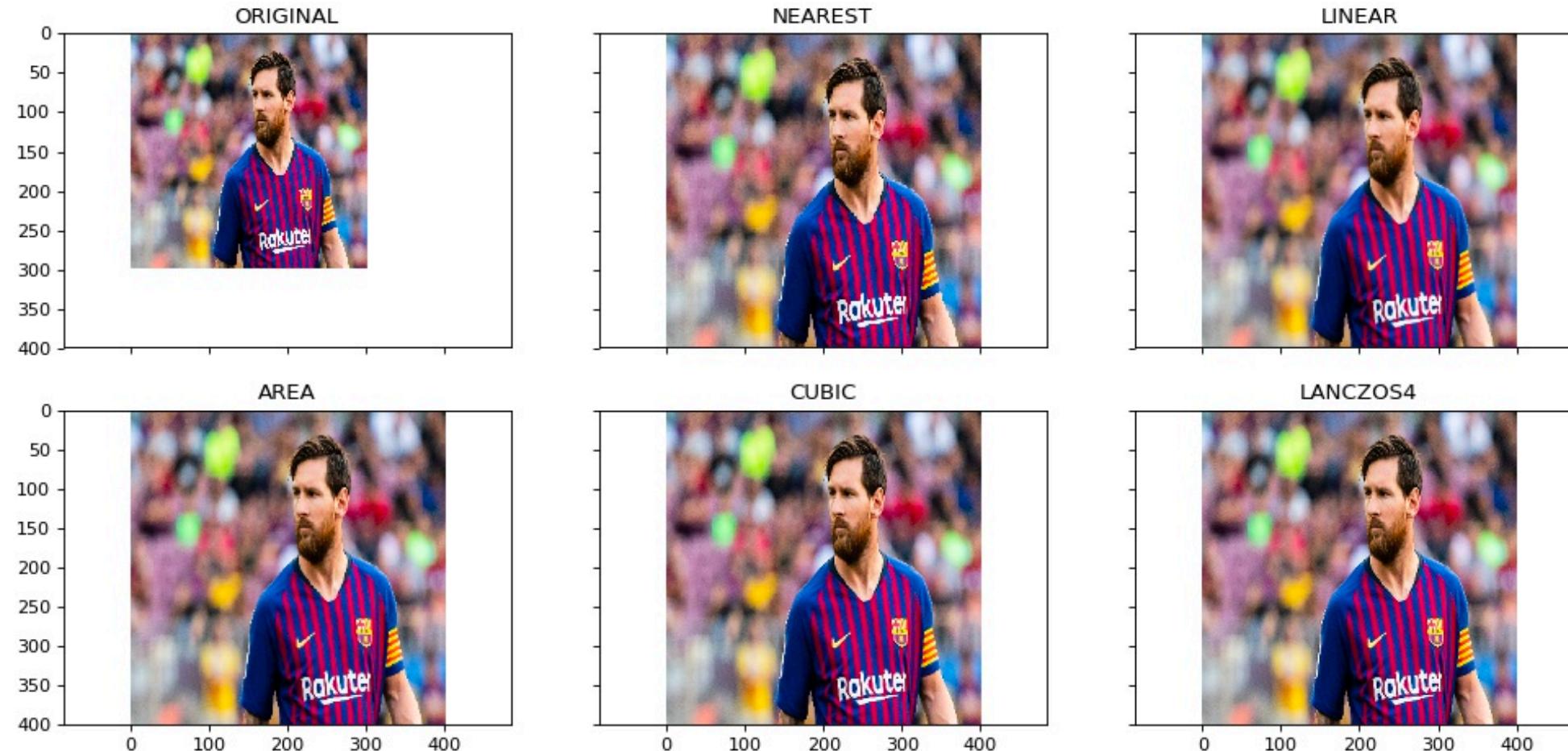
---

```
# Đọc ảnh  
image = cv2.imread('index.jpg')  
# Resize về size (100,100)  
smaller_image = cv2.resize(image, (100,100), interpolation='linear')
```

- **INTER\_NEAREST**: Nearest neighbor interpolation
- **INTER\_LINEAR**: Bilinear interpolation (mặc định)
- **INTER\_AREA**: Resampling using pixel area relation
- **INTER\_CUBIC**: Bicubic interpolation over  $4 \times 4$  pixel neighborhood
- **INTER\_LANCZOS4**: [Lanczos interpolation](#) over  $8 \times 8$  neighborhood

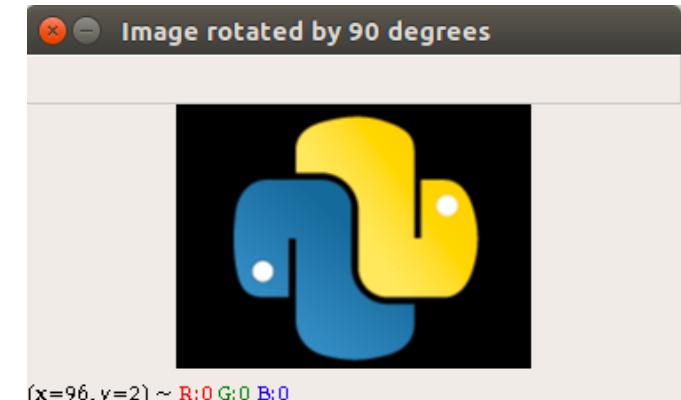
# Hàm Resize ảnh

QUESTION



# Hàm Rotate ảnh

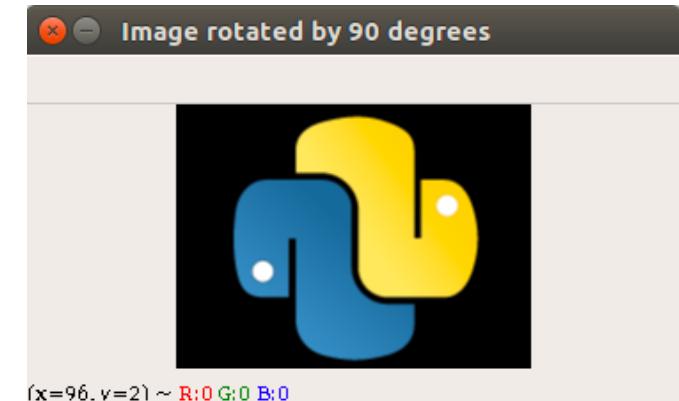
```
# Đọc ảnh  
img = cv2.imread('python.png')  
# Lấy cao, rộng  
(h, w) = img.shape[:2]  
# Tính tâm bức ảnh, góc quay và mức độ scale  
center = (w / 2, h / 2)  
angle90 = 90  
scale = 1.0  
# Quay ngược 90 độ  
M = cv2.getRotationMatrix2D(center, angle90, scale)  
rotated90 = cv2.warpAffine(img, M, (h, w))
```



# Hàm Rotate ảnh

---

```
# import thư viện  
import imutils  
# Xoay  
rotated = imutils.rotate(image, 90)
```



# Hàm Threshold

---

- Là một hàm phân ngưỡng ảnh nhằm đưa ảnh về dạng nhị phân (đen và trắng, 0 và 255)
- Ảnh nhị phân thường được dùng cho phân đoạn ảnh (segmentation)
- Nhị phân hóa ảnh là bước tiền xử lý rất hữu ích cho các giải thuật nhận dạng chữ viết / ký tự (OCR: Optical Character Recognition).

# Hàm Threshold

---

Nếu pixel có giá trị lớn hơn giá trị ngưỡng thì nó được gán 1 giá trị (thường là 1), ngược lại nhỏ hơn giá trị ngưỡng thì nó được gán 1 giá trị khác (thường là 0).

Original image



Result



# Hàm Threshold

---

## Các bước nhị phân hóa ảnh

- Biến đổi ảnh màu (color) sang ảnh xám (grayscale).
- Thiết lập ngưỡng (threshold) để nhị phân hóa ảnh.
- Áp dụng ngưỡng vào ảnh xám để tạo ảnh nhị phân hoặc áp dụng giải thuật nhị phân hóa ảnh. Các pixel có giá trị lớn hơn ngưỡng ta thiết lập bằng 255 (hoặc 1), nhỏ hơn ngưỡng ta thiết lập bằng 0.

# Hàm Threshold

---

```
# Đọc ảnh  
gray_image = cv2.imread('index.png',0)  
# Áp threshold  
ret,thresh_binary =  
cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY)
```

# Hàm Threshold ảnh

Original Image



BINARY



THRESH\_BINARY\_INV



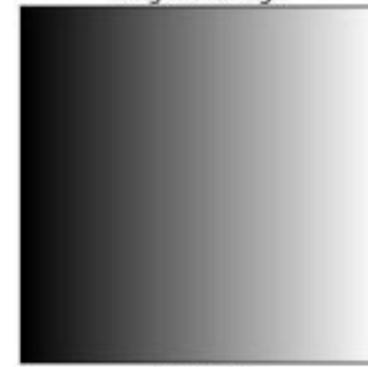
THRESH\_TRUNC



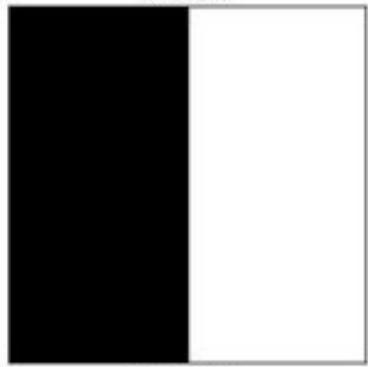
THRESH\_TOZERO



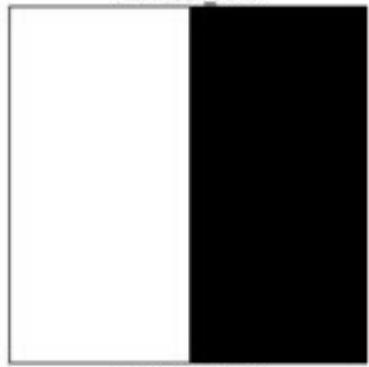
Original Image



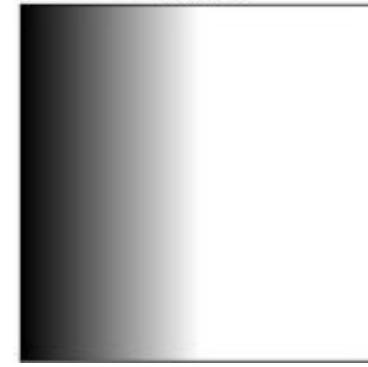
BINARY



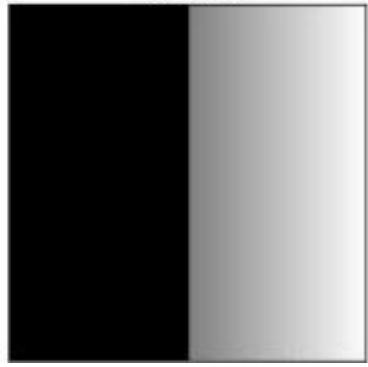
BINARY\_INV



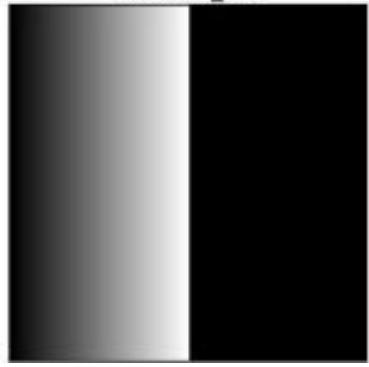
TRUNC



TOZERO



TOZERO\_INV



# Hàm Adaptive Threshold

---

Vậy chọn ngưỡng như thế nào là phù hợp?



# Hàm Adaptive Threshold

---

Adaptive Thresholding lựa chọn ngưỡng (threshold) động trong vùng lân cận:

- Đối với mỗi pixel ảnh, ta xét vùng ảnh con có  $B \times B$  ( $B$  là số lẻ) với pixel đang xét là trung tâm.
- Các mức sáng trong vùng ảnh con  $B \times B$  -> ta tính giá trị trung bình  $M$  (đối với phương pháp adaptive "cv2.ADAPTIVE\_THRESH\_MEAN\_C") -> threshold áp dụng cho pixel đang xét:  $\text{threshold} = M - C$  ( $C$  là hằng số xác định trước).

# Hàm Adaptive Threshold

---

Như vậy độ tốt của Adaptive Thresholding phụ thuộc vào 3 siêu tham số (hyper parameter):

- adaptiveMethod: cv2.ADAPTIVE\_THRESH\_MEAN\_C / cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C
- B: kích thước vùng lân cận
- C: hằng số mà ta sẽ trừ đi (C có thể là số dương, số âm hay bằng 0)

# Hàm Adaptive Threshold

---

Original Image



Global Thresholding ( $v = 127$ )



Adaptive Mean Thresholding



Adaptive Gaussian Thresholding



# Hàm tìm cạnh

---

Trong hình ảnh, thường tồn tại các thành phần như: vùng trơn, góc / cạnh và nhiễu. Cạnh trong ảnh mang đặc trưng quan trọng, thường là thuộc đối tượng trong ảnh (object). Do đó, để phát hiện cạnh trong ảnh, giải thuật Canny là một trong những giải thuật phổ biến / nổi tiếng nhất trong Xử lý ảnh.

# Hàm tìm cạnh

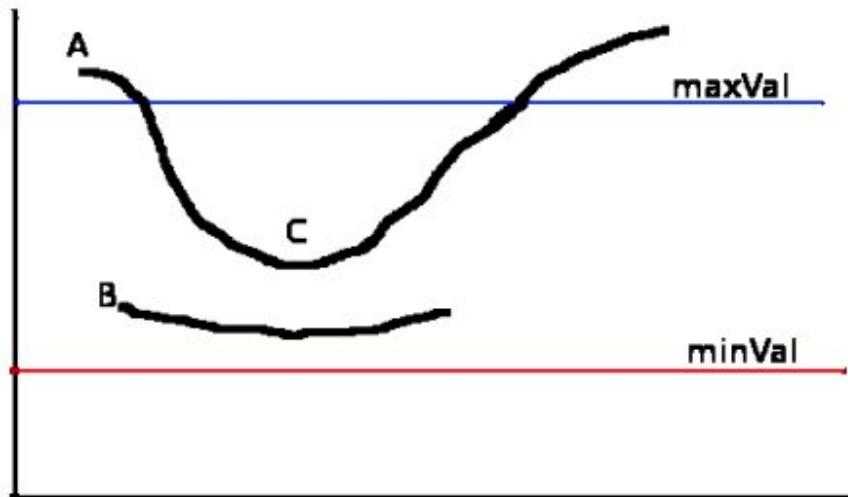
---

```
# Đọc ảnh
```

```
gray_image = cv2.imread('index.png',0)
```

```
# Áp threshold
```

```
edges = cv2.Canny(gray_image , min_val=100 , max_val=200)
```



# Hàm làm mờ, giảm nhiễu

# Đọc ảnh

```
gray_image = cv2.imread('index.png', 0)
```

# Làm mờ bằng Gaussian Blur

```
edges = cv2.GaussianBlur(img, ksize=(KERNEL_WIDTH, KERNEL_HEIGHT))
```

kernel\_width và  
kernel\_height là số lẻ và  
độ mờ càng cao khi số  
này càng lớn.



# Hàm tìm contours

---

- Contour là “tập các điểm-liên-tục tạo thành một đường cong (curve) (boundary), và không có khoảng hở trong đường cong đó.
- Trong opencv, việc tìm một contour là việc tìm một đối tượng có màu trắng trên nền đen
- Chi tiết:  
[https://docs.opencv.org/3.4/d4/d73/tutorial\\_py\\_contours\\_begin.html](https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html)

# Hàm tìm contours

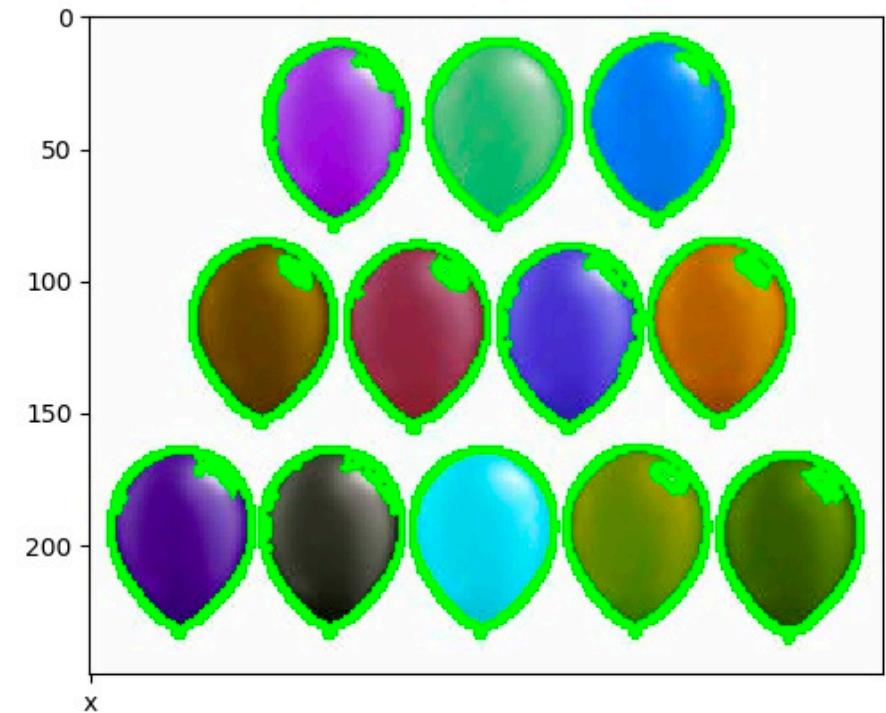
---

Bài toán: Có bao nhiêu quả bóng trong hình?



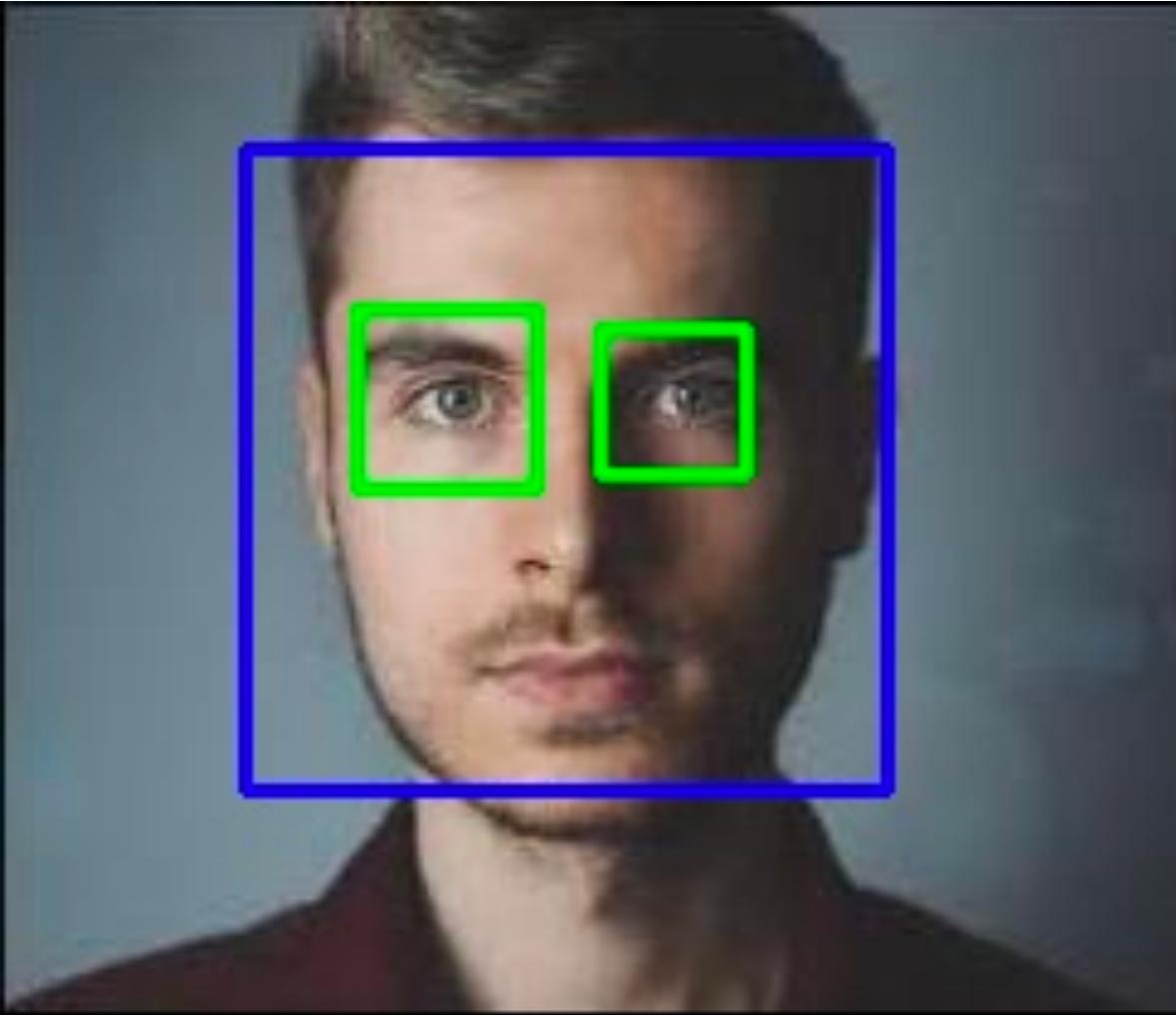
# Hàm tìm contours

```
_ , contours, _ =  
cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)  
  
cv2.drawContours(im, contours, -1, (0, 255, 0), 2) # vẽ lại ảnh  
contour vào ảnh gốc
```



# Hàm Face Detection

---



# Hàm Face Detection

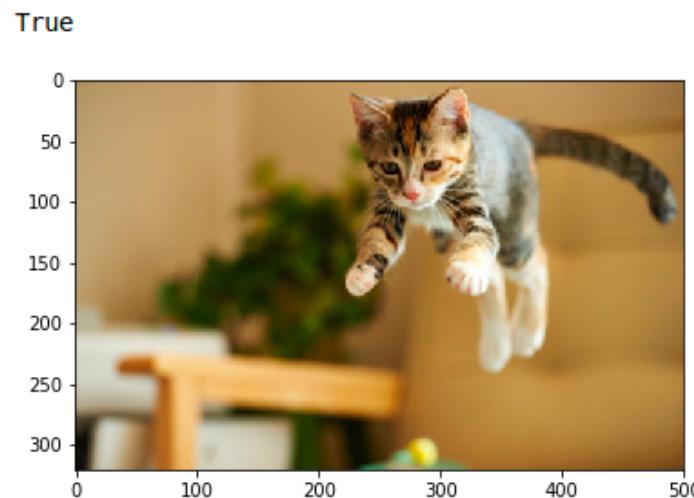
---

```
#load the classifiers downloaded  
face_cascade =  
cv.CascadeClassifier('haarcascade_frontalface_default.xml')  
  
#read the image and convert to grayscale format  
img = cv.imread('rotated_face.jpg', cv.COLOR_BGR2GRAY)  
  
#calculate coordinates  
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
```

# Bài thực hành

---

Bài 1: Đọc ảnh từ camera và chuyển sang màu xám, hiển thị lên màn hình.



## Bài thực hành

---

Bài 2: Đọc ảnh từ một file video, hiển thị lên màn hình. Nếu người dùng nhấn phím A thì video quay ngược chiều kim đồng hồ 90 độ, nếu nhảm phím D thì video quay theo chiều kim đồng hồ 90 độ.

## Bài thực hành

---

Bài 3: Đọc ảnh từ một webcam, nhận diện các khuôn mặt và vẽ một hình chữ nhật màu xanh Green quanh khuôn mặt. Đếm số khuôn mặt có trong hình. Nếu người dùng nhấn phím S thì lưu các khuôn mặt đó ra các file ảnh 0.png, 1.png, 2.png.... (tùy số lượng khuôn mặt)

# Bài thực hành

---

Bài 4: Đếm và in ra số bóng trong hình?



## Bài thực hành

---

Bài 5: Đọc ảnh biển số xe và crop các chữ số trên biển lưu ra file ảnh.

