TRƯỜNG TRUNG HỌC PHỔ THÔNG
CHUYÊN LÊ QUÝ ĐÔN

BÀI BÁO CÁO TỔNG KẾT

Đề tài

ỨNG DỤNG OPENCV ĐỂ XÁC ĐỊNH MÀU SẮC VÀ HÌNH DẠNG

NHÓM 4

Nhóm trưởng:
HOÀNG NGỌC DUNG

Thành viên:
PHẠM LÊ HỒNG PHONG

Thành viên:
PHAN THẾ THANH SƠN

Nha Trang, Tháng 10 năm 2022

Nội dung

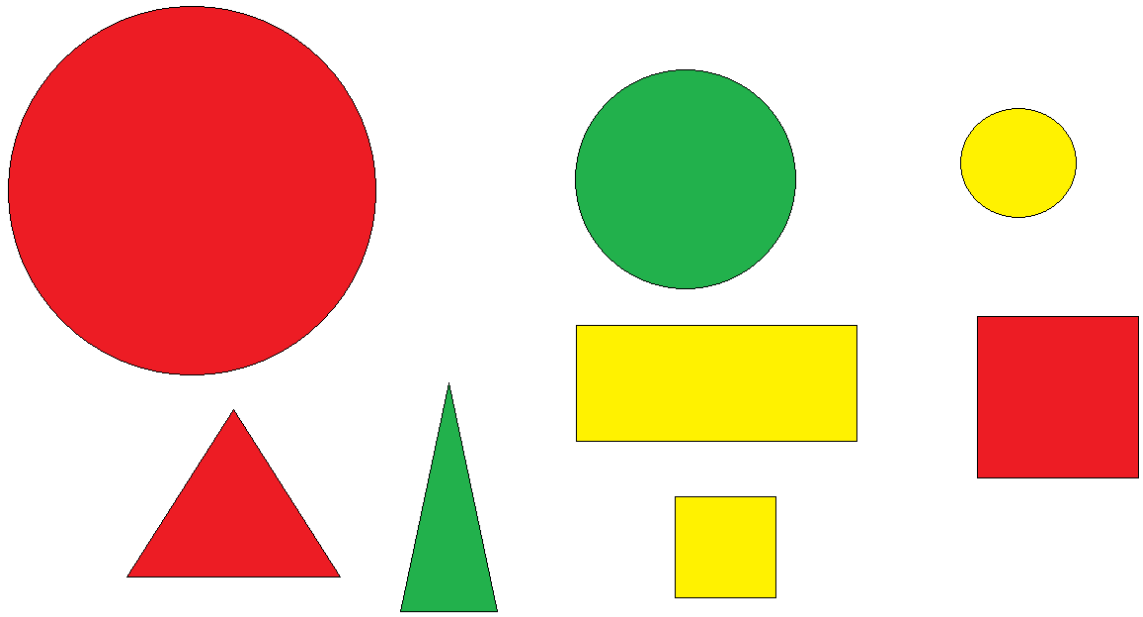
I- MỞ ĐẦU

OpenCV là một thư viện mở được dùng chủ yếu trong việc xử lý ảnh và video để xác định những hình dạng, đối tượng, chữ, …. Trong bài báo cáo này nhóm em sẽ ứng dụng Opencv để xác định có bao nhiêu hình vuông, hình chữ nhật (trừ hình vuông), hình tròn, hình tam giác và màu sắc mỗi loại.

II- CƠ SỞ LÝ THUYẾT

1. Dữ liệu

Đầu vào đảm bảo hình ảnh chuẩn rõ nét không cần trải qua bước tiền xử lý ảnh. Chỉ gồm hình vuông, hình chữ nhật, tam giác (các đa giác lồi) và hình tròn với các kích cỡ khác nhau. Mỗi đa giác có thể là màu đỏ, vàng hoặc xanh lá.
Ví dụ:



2. Xác định hình dạng (Shape Detection)

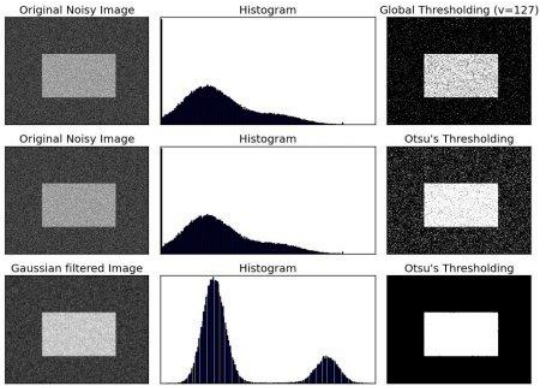
2.1. Tổng quan:

- Nhóm em xác định hình dạng và vị trí của 1 đối tượng bằng cách sử dụng thuật toán $Contours$ trong OpenCV.
- Nhận xét:
    + Có 4 cạnh bằng nhau, 4 đỉnh là hình vuông, góc 90^o
    + Có 4 cạnh, 4 đỉnh là hình chữ nhật

+ Có 3 cạnh là tam giác, nếu có góc 90^o là tam giác vuông, nếu có góc 60^o là tam giác cân, …
+ Còn lại là hình tròn

Tuy nhiên, với một đa giác ta chỉ cần quan tâm tới đỉnh và cạnh bởi khi sử dụng Contours ta không cần quan tâm tới góc. Vì vậy, phương pháp nhóm em tiếp cận để xác định hình dạng một đa giác bằng số lượng cạnh của đó giác đó. Ví dụ, nếu phát hiện có 3 cạnh thì là tam giác, 4 cạnh là tứ giác (hình vuông hoặc hình chữ nhật), …

2.2 Các kiến thức cần quan tâm



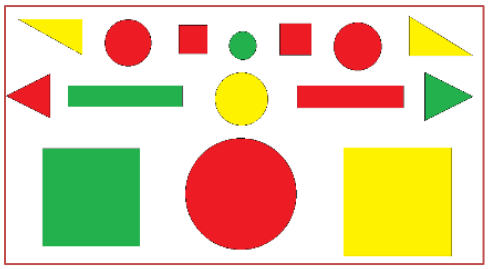
Thresholding(ngưỡng)[1]: một kỹ thuật trong OpenCV, nó là việc gán các giá trị đến ngưỡng được cung cấp. Tức mỗi giá trị pixel so sánh với giá trị ngưỡng. Nếu giá trị pixel nhỏ hơn ngưỡng nó được thành 0, ngược lại tối đa (thường là 255).

Giá trị thresholding: giá trị cố định vẽ đường ranh giới (boundary line) giữa 2 tập data (dữ liệu)

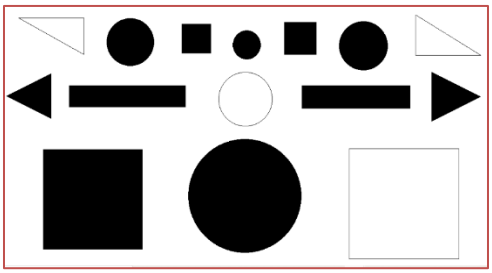
Hình ảnh nhị phân (Bi-value): chỉ có thể sử dụng hai giá trị cường độ bi hoặc hai giá trị cường độ để đại diện cho toàn bộ hình ảnh. Trong xử lý hình ảnh nói chung, chúng ta nói một nhị phân hình ảnh khi, nó chỉ bao gồm các pixel đen và trắng. Ngoài ra, giá trị ngưỡng có thể ảnh hưởng ít nhiều đến việc xử lý ảnh. Bởi khi để ngưỡng quá thấp, thì ảnh xám sẽ được xem như màu trắng.

[1] **OpenCV: Basic Thresholding Operations**

Ví dụ

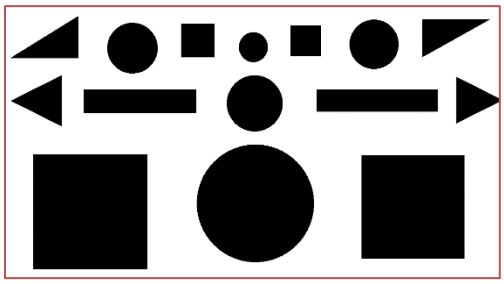


Ảnh gốc



Ảnh lọc 1

Ảnh lọc 2



Giải thích: Ta thấy hình 1 lọc với ngưỡng thấp hơn ảnh xám nên ảnh hưởng đến chát lượng hình ảnh. Nói đơn giản ảnh xám là 128 nhưng để ngưỡng 127 thì vật màu xám sẽ bị coi là màu nền. Vì vậy, khi ta biết màu trắng 255 là cao nhất. Ta cứ tăng ngưỡng cho đếm mức thì hợp.

Contours: đường viền có thể giải thích đơn giản là một đường cong nối tất cả các điểm liên tục(dọc theo đường biên). Chúng có cùng tính chất về màu sắc hoặc cường độ...). Đây là công cụ hữu để phát hiện, nhận dạng hình dạng.
Một số hàm liên quan Contours.

a. cv2.findContours(): tìm ra các điểm biên của hình dạng trong ảnh

Cú pháp***:***

$cv2.findContours(input, contour\_retrieval, contour\_approximation)$

Trong đó:

- input: hình ảnh đưa vào.
- contour_retrieval:
  - cv.RETR_EXTERNAL: truy xuất các đường viền ngoài cùng.
  - cv.RETR_LIST: lấy tất cả đường bao và đưa chúng vào danh sách.
  - cv.RETR_TREE: truy xuất đường viền và thiết lập mối quan hệ giữa các đường viền lồng nhau.

- cv.RETR_CCOMP: truy xuất tất cả các đường bao và sắp xếp chúng thành 2 cấp: cấp cao là ranh giới bên ngoài của các thành phần, cấp thấp hơn là ranh giới của các lỗ.

- contour_approximation:
  - cv.CHAIN_APPROX_NONE: lưu trữ các điểm ranh giới.
  - cv.CHAIN_APPROX_SIMPLE: lưu trữ số lượng điểm cuối(ví dụ tam giác sẽ lưu 3).

Trả về danh sách các điểm đường viền.

b. cv2.drawContours(): vẽ một đường viên hoặc vẽ hình dạng nếu có đủ các điểm biên

Cú pháp:

$$cv2.findContours(input, contour, contourlndex, colour, thickness)$$

Trong đó:

- input: hình ảnh đưa vào.
- contour: các điểm đường viền có thể được liệt kê.
- contourlndex: -1: vẽ tất cả các đường viền.
- colour: giá trị màu để vẽ đường bao riêng lẻ.
- thickness: kích thước để vẽ đường viền riêng lẻ

Hàm cv2.approxPolyDP(): thực hiện ước lượng, tính toán xấp xỉ hình dạng của một đường viên.

Hàm này được triển khai theo thuật toán Douglas-Peucker[2]

Cú pháp:

$$cv2.approxPolyDP(input, epsilon, ok)$$

Trong đó:

- input: đa giác đầu vào.
- Epsilon: khoảng cách tối đa giữa giá trị ước lượng của một đường bao hình dạng của đa giác đầu vào so với chính đa giác đầu vào đó.
- Ok: giá trị Boolean với 2 tình huống: đúng với đường cong gần đúng được đóng; sai với đường cong gần đúng không được viền.

Hàm cv2.moment()[3]: Tính toán khối tâm, diện tích của vật thể

[2] Ramer–Douglas–Peucker algorithm - Wikipedia
[3] Image moment - Wikipedia
The Maths behind Contour Moments from OpenCV | by Alex Yang Ph.D. | Medium (medium0.com)

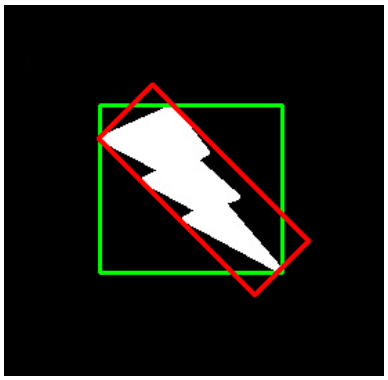
Khác với contour là một đường cong nối liền các điểm có cùng cường độ hoặc màu sắc. Bounding box là một đường biên đóng bao quanh vật thể. Một bounding box có thể được xác định thông qua một contour.

Hàm boundingRect (): Cung cấp các điểm biên của hình chữ nhật.

Tuy nhiên, hình chữ nhật là học phẳng nó không được coi là chuyển động quanh vật thể. Vì vậy kết quả hàm **boundingRect()** sẽ có thể không cho ra giới hạn nhỏ nhất. Nên để giảm thiểu sai sót ta sẽ sử dụng hàm **minAreaRect()** khi hình chữ nhật bị nghiêng/ xoay.

Hàm cv2.minAreaRect(): Vẽ bounding box hình chữ nhật xoay diện tích nhỏ nhất bao quanh một contour.

Hàm trả về cấu trúc Box2D(tâm(x,y), chiều rộng, chiều cao góc quay). Để lấy 4 góc hình chữ nhật ta sẽ sử dụng hàm **cv2.boxPoints()**



- Đường màu xanh khi ta sử dụng hàm boundingRect ()
- Đường màu đỏ khi ta sử dụng hàm minAreaRect ().

Hàm putText(): Viết chữ lên hình

Cú pháp:

$cv2.putText(input, text, org, font, fontScale, color\ [, thick\ [, lType[, BLO]]])$

Trong đó:

- input: hình ảnh đưa vào.
- text: đoạn văn bản sẽ được vẽ sau đó.
- org: tọa độ góc dưới bên trái của chuỗi văn bản trong hình ảnh, được biểu diễn dưới cặp gồm 2 giá trị là (X, Y).
- font: loại font chữ được dùng (FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN…).
- fontScale: (hệ số font chữ) *(kích thước của font chữ cụ thể).

- color: màu của chuỗi văn bản, với BGR, vượt qua 1 tuple.
- thick: độ dày (đơn vị px).
- lType: cung cấp loại đường được sử dụng.
- BLO: có 2 tình huống: nếu giá trị của nó là **true** thì nguồn gốc dữ liệu nằm góc dưới cùng bên trái; nếu **false** thì nó sẽ nằm ở góc trên cùng bên trái.

Ngoài ra, sau khi xác định có 4 cạnh để giảm thiểu sai số, chúng ta dùng khoảng cách trêm mặt phẳng tọa độ hai chiều và hàm cv2.minAreaRect() tìm ra bounding box diện tích nhỏ nhất bao quanh một contour xác định. Do nó giảm thiểu sai sot đối với cả bounding box là hình chữ nhật bị nghiêng.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (p_2 - p_2)^2}$$

3. Xác định màu sắc (Color Detection)

3.1. Tổng quan

Xác định màu là quá trình xác định tên của một màu bất kì. Nghe đơn giản nhưng thật ra việc này là cực kì đơn giản đối với con người nhưng đối với máy tính thì lại không dễ dàng như thế. Mắt và não của người hoạt động cùng nhau để phiên dịch ánh sáng thành màu sắc. Trong mắt chúng ta có cơ quan thụ cảm ánh sáng làm nhiệm vụ truyền tín hiệu tới não và sau đó não bộ sẽ nhận diện được màu sắc. Khi còn nhỏ chúng ta đã kết nối những loại ánh sáng nhất định với tên màu sắc của chúng và chúng ta cũng sẽ sử dụng cách thức tương tự để xác định tên màu sắc.
Tuy nhiên, ta thấy đầu vào hình ảnh các vật thể chỉ gồm duy nhất một màu. Nên sau khi thực hiện công việc xác định hình dạng ta tận dụng vị trí trọng tâm để xác định màu sắc của chúng.

3.2 Kiến thức liên quan

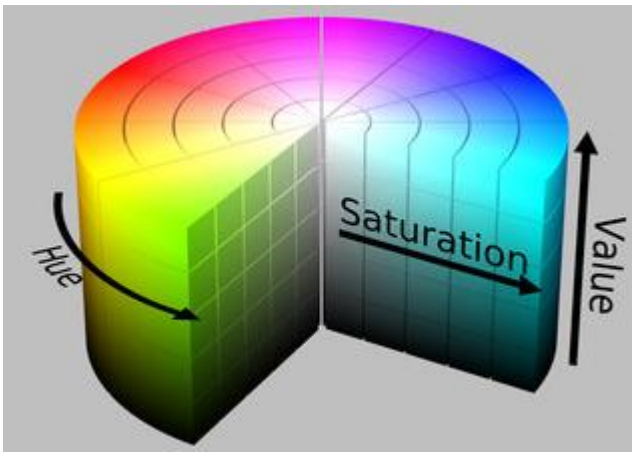
Tập dữ liệu (Dataset):

Màu sắc được tạo thành bởi 3 màu chính: đỏ, xanh lá cây và xanh nước biển. Trên máy tính, ta định nghĩa mỗi giá trị màu trong 1 khoảng từ 0 đến 255. Số cách định nghĩa 1 là 256*256*256 = 16 581 375. Có khoảng chừng 16,5 triệu cách để biểu diễn 1 màu. Trong tập dữ liệu cả chúng ta, ta cần ánh xạ mỗi giá trị màu với tên tương ứng của chúng. Nhưng chúng ta không cần phải làm với tất cả giá trị mà sẽ sử dụng 1 tập dữ liệu có chứa các giá trị RGB với tên tương

ứng. Tuy nhiên, để nhận diện màu sắc ta cần chuyển RGB sang không gian HSV để thực hiện.

HSV (hue, saturation, value)

Không gian màu HSV (sắc độ, độ bão hòa, giá trị) là một mô hình để biểu thị không gian màu tương tự như mô hình màu RGB. Vì kênh màu mô hình hóa kiểu màu nên nó rất hữu ích trong các tác vụ xử lý hình ảnh cần phân đoạn các đối tượng dựa trên màu sắc của nó. Sự thay đổi của độ bão hòa đi từ không bão hòa để biểu thị các sắc thái của màu xám và bão hòa hoàn toàn (không có thành phần màu trắng). Kênh giá trị mô tả độ sáng hoặc cường độ của màu. Hình ảnh tiếp theo cho thấy hình trụ HSV.Opencv giảm một nửa giá trị H để phù hợp với phạm vi [0,255], vì vậy giá trị H thay vì nằm trong phạm vi [0, 360], nằm trong phạm vi [0, 180]. S và V vẫn nằm trong khoảng [0, 255].



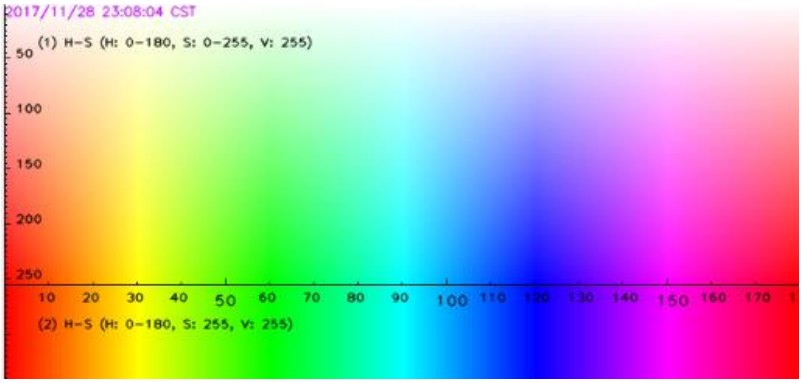
Hinh 1.1[4]

Trong OpenCV, Hue có các giá trị từ 0 đến 180, Độ bão hòa(Saturation) và Giá trị(Value) từ 0 đến 255. Do đó, OpenCV sử dụng HSV trong khoảng (0-180, 0-255, 0-255). Một số mã màu[5].

[4] python - Choosing the correct upper and lower HSV boundaries for color detection with`cv::inRange` (OpenCV) - Stack Overflow

[5] color-names/colors.csv at master · codebrainz/color-names (github.com)

HSV color map:



- Chuyển từ RGB sang HSV[6]
  - Giá trị màu đỏ (0, 255, 255) nhưng để phù hợp trong OpenCV các giá trị màu HSV xấp xỉ trong phạm vi từ và 160 đến 180.
  - Giá trị màu xanh (0,255,0) nhưng để phù hợp trong OpenCV thì các giá trị màu HSV xấp xỉ trong phạm vi từ và 40.
  - Giá trị màu vàng (255, 255, 0) nhưng để phù hợp trong OpenCV các giá trị màu HSV xấp xỉ trong phạm vi từ và 15 đến 35.

III- PHƯƠNG PHÁP THỰC HIỆN

1. Xác định hình dạng (Shape Dectection)

Bước 1: import thư viện

```
import cv2
import numpy as np
```

Bước 2: Đưa hình ảnh đầu vào

```
img = cv2.imread('Data1/pic 7.png')
```

Bước 3: Chuyển màu ảnh thành thang màu xám

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Bước 4:

- Áp dụng ngưỡng (thresholding) trên ảnh.
- Sử dụng hàm findContours () tìm các đường viền(contours)
- Lưu ý: bức ảnh phải được chuyển về dạng sau: các đối tượng trong đó phải là màu trắng và nền phải là màu đen hoặc các đối tưởng phải màu đen và nên phải màu trắng.

[6] Open CV xử lý ảnh bài 1 xử lý màu, chuyển hệ màu (viblo.asia)

```
_, threshold = cv2.threshold(gray, 240, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(threshold, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

Bước 5: Duyệt mảng contours. Với mỗi đường viền được phát hiện, hình dạng của chúng được ước lượng gần đúng bằng hàm approxPlolyDP().
Lưu ý: Loại vị trí đầu tiên (biên ảnh)

```
for contour in contours:
    if i == 0:
        i = 1
        continue
    approx = cv2.approxPolyDP(contour, 0.01 * cv2.arcLength(contour, True),
True)
```

Bước 6: Vẽ đường viên và tìm tâm của hình (có thể không cần hoặc lấy x, y là vị trí pixel đầu tiên của vật – linh hoạt)

- Vẽ đường viền của hình dạng, sử dụng drawContours() màu xanh lá
- Tìm tâm của hình dạng: $C_x = \frac{M_{10}}{M_{00}}$ and $C_y = \frac{M_{01}}{M_{00}}$

```
cv2.drawContours(img, [contour], 0, (0,255,0), 5)
M = cv2.moments(contour)
x = int(M['m10']/M['m00'])
y = int(M['m01']/M['m00'])
```

Bước 7: Phân loại hình dạng tìm được và đặt tên cho chúng. Trong quá trình đó ta kết hợp thực hiện đếm số lượng hình vuông, hình chữ nhật, hình tam giác, hình tròn.
Sau khi tính approx (ước lượng xấp xỉ hình dạng):

- Nếu có 3 cạnh là hình tam giác
- Nếu có 4 cạnh là tứ giác (hình chữ nhật hoặc hình vuông)
  + Nếu 2 cạnh bằng nhau thì hình vuông
  + Ngược lại là hình chữ nhật
  + Ngược lại hình chữ nh
- Trường hợp còn lại là hình tròn
- *Lưu ý: cv2.norm()[7]: tìm khoảng cách trên mặt phẳng tọa độ 2 chiều (euclide)*

[7] cv.norm - mexopencv (amroamroamro.github.io)

- Box là mảng toa độ 4 đỉnh của hình
- Sử dùng hàm putText ghi chữ ở giữa hình và chọn font FONT_HERSHEY_SIMPLEX để viết cỡ chữ 0.6 màu đen và độ đậm 2.

```
if len(approx) == 3:
        cv2.putText(img, 'Triangle', (x, y),cv2.FONT_HERSHEY_SIMPLEX,
0.6, (0), 2)
        cnt_tri += 1
    elif len(approx) == 4:
        rect = cv2.minAreaRect(contour)
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        tmp = cv2.norm(box[0], box[1])/ cv2.norm(box[0], box[-1])
        esp = 1e-2
        if (1 - esp < tmp < 1 + esp):
            cv2.putText(img, 'Square', (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0), 2)
            cnt_squ += 1
        else:
            cv2.putText(img, 'Rectangle', (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0), 2)
            cnt_rec += 1
```

2. Xác định màu sắc (Color Dectection)

Bước 1: import thư viện

```
import cv2
import numpy as np
```

Bước 2: Chuyển ảnh sang hệ màu HSV

```
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Bước 3: Vì đã có duyệt contour từ bước xác định hình dạng (shape dectection) ta sẽ kế thừa việc tính trọng tâm của vật.

- Tại vị trí trọng tâm ta sẽ xét màu của vị trị
- Lưu ý: y trước x (kích thước ảnh hsv)

```
M = cv2.moments(contour)
x = int(M['m10']/M['m00'])
y = int(M['m01']/M['m00'])
pi = hsv_img[y][x]
c = pi[0]
```

Bước 4: Phân loại màu sắc tìm được. Trong quá trình đó ta kết hợp thực hiện đếm số lượng màu.

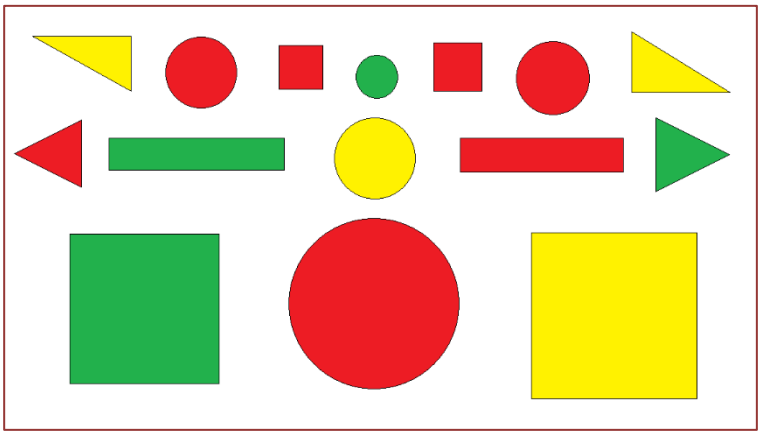
- Vì chỉ xét 1 điểm pixel nên chúng ta chỉ cần quan tâm đến Hue
- lower_red = np.array([160,100,100])
- upper_red = np.array([180,255,255])
- lower_yellow = np.array([15,150,150])
- upper_yellow = np.array([35,255,255])
- lower_green = np.array([40,50,50])
- upper_green = np.array([90,255,255])
- Sử dùng hàm putText ghi chữ ở giữa hình và chọn font FONT_HERSHEY_SIMPLEX để viết cỡ chữ 0.6 màu đen và độ đậm 2.

```
if c >= 160 and c <=180:
        name_color = "Red"
        cnt_re+=1
        cv2.putText(img, name_color, (x- 40, y + 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0), 2)
    elif c >= 15 and c <= 35:
        name_color = "Yellow"
        cnt_ye+=1
        cv2.putText(img, name_color, (x- 40, y + 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0), 2)
    elif c >= 40 and c <= 90:
        name_color = "Green"
        cnt_gr+=1
       cv2.putText(img, name_color, (x- 40, y + 20),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0), 2)
     cv2.putText(img, name_color, (x- 40, y + 20), cv2.FONT_HERSHEY_SIMPLEX,
0.6, (0), 2)
```

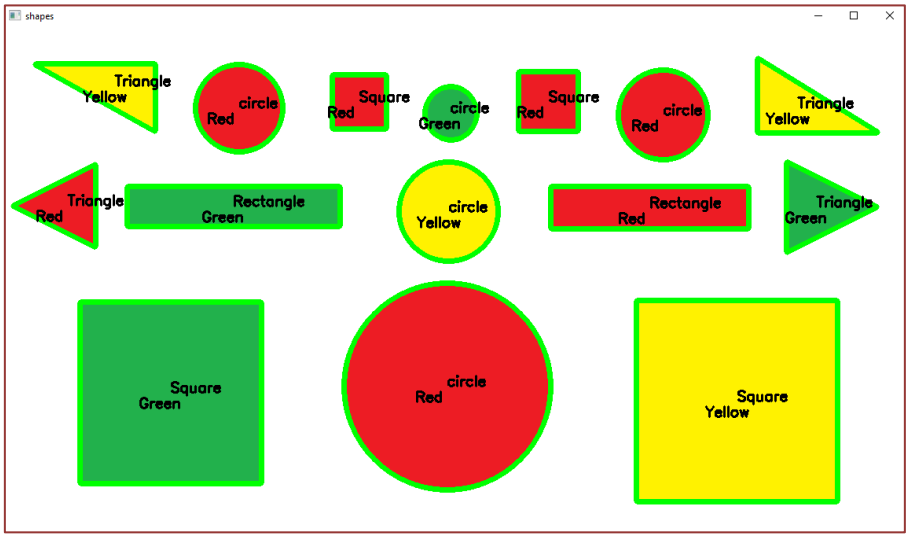
IV- KẾT QUẢ

- TH1: đối với ảnh gồm những hình thẳng (không nghiêng). Kết quả đầu ra khá chính xác.

Ảnh gốc



Ảnh kết quả

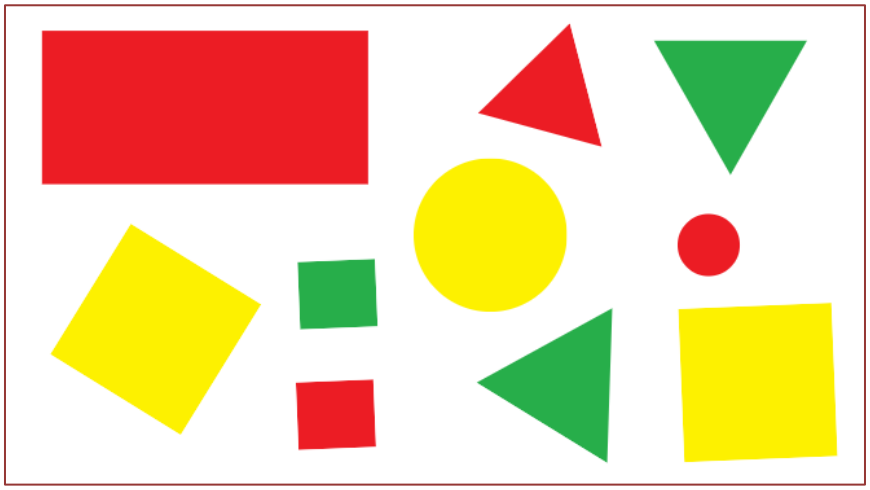


Terminal hiển thị

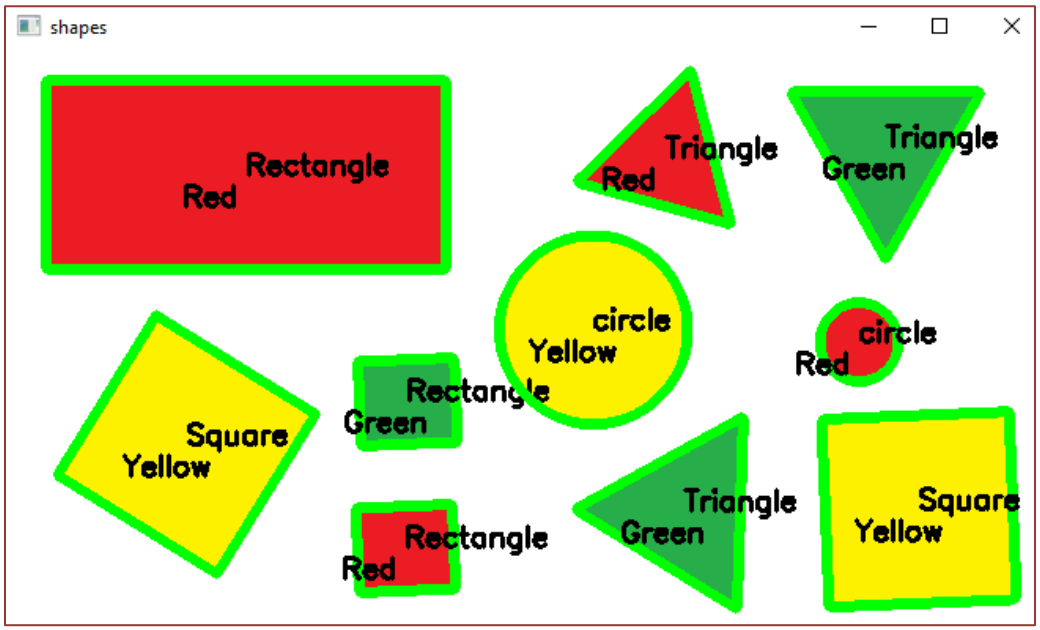
```
Kết quả
+ Có 5 hình tròn, 4 hình tam giác, 2 hình chữ nhật, 4 hình vuông.
+ Màu đỏ: 7 hình
+ Màu xanh: 4 hình
+ Màu vàng: 4 hình
```

- Th2: đối với ảnh gồm những hình nghiêng(xoay). Kết quả đầu ra không tốt bằng TH1.

Ảnh gốc



Ảnh kết quả



Terminal hiển thị

```
Kết quả
+ Có 2 hình tròn, 3 hình tam giác, 3 hình chữ nhật, 2 hình vuông.
+ Màu đỏ: 4 hình
+ Màu xanh: 3 hình
+ Màu vàng: 3 hình
```

V- TÀI LIỆU THAM KHẢO

Contour:
OpenCV: Contour Features
Tài liệu xử lý ảnh – Contour, moments – Univ.Tech hơn cả thực hành (wordpress.com)
Khoa học dữ liệu (phamdinhkhanh.github.io)
The Maths behind Contour Moments from OpenCV | by Alex Yang Ph.D. | Medium (medium0.com)
Shape Detection:
Detecting Geometrical Shapes in an image using OpenCV | by Simarpreet Singh | Simply Dev | Medium (medium0.com)
How to Detect Shapes in Images in Python using OpenCV? - GeeksforGeeks
OpenCV ApproxPolyDP | Learn the Examples of OpenCV ApproxPolyDP (educba.com)
Shape Detection & Tracking using Contours - OpenCV Tutorial C++ (opencv-srf.com)
Color Detection:
Color Detection & Object Tracking - OpenCV Tutorial C++ (opencv-srf.com)
Color Detection using Python - Beginner's Reference - AskPython
Color Detection using OpenCV Python | by Praveen | programming_fever | Medium (medium0.com)
HSV:
OpenCV: Thresholding Operations using inRange
OpenCV: Basic Thresholding Operations
Color Filtering/Segmentation/Detection – HSV | Computer Vision (wordpress.com)
Color data:
color-names/colors.csv at master · codebrainz/color-names (github.com)
Python OpenCV | cv2.putText() method - GeeksforGeeks