TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO TIỂU LUẬN GIỮA KÌ
MÔN XỬ LÝ ẢNH SỐ

Người hướng dẫn: **TS TRỊNH HÙNG CƯỜNG**

Người thực hiện: **LÊ THÀNH ĐĂNG KHOA – 51900119**

NGUYỄN LÊ BẢO THY - 51900239

Lớp **: 19050202**

Khoá **: 23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO TIỂU LUẬN GIỮA KÌ
MÔN XỬ LÝ ẢNH SỐ

Người hướng dẫn: **TS TRỊNH HÙNG CƯỜNG**
Người thực hiện: **LÊ THÀNH ĐĂNG KHOA – 51900119**
NGUYỄN LÊ BẢO THY - 51900239
Lớp **: 19050202**
Khoá **: 23**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Trước hết em xin gửi lời cảm ơn chân thành đến Ts Trịnh Hùng Cường đã giúp đỡ chúng em trong suốt quá trình học tập và thực hiện bài tiểu luận. Thầy đã tận tâm hướng dẫn và chia sẽ cho chúng em những kiến thức hữu ích về môn học. Chúng em rất biết ơn thầy.

Xin cảm ơn khoa Công nghệ thông tin trường Đại học Tôn Đức Thắng đã đã tạo điều kiện để chúng em có thể được học bộ môn.

NHÓM XIN CHÂN THÀNH CẢM ƠN!

TIỂU LUẬN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của TS Trịnh Hùng Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung tiểu luận của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Lê Thành Đăng Khoa

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Lê Bảo Thy

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Tiểu luận nhóm chúng em nghiên cứu về các vấn đề liên quan đến thư viện open Cv bao gồm thay đổi màu sắc của ảnh, tạo các đường vector trên ảnh và chọng các vùng màu sắt trên ảnh và ứng dụng chúng giải quyết các bài tập được giao.

PHỤ LỤC

PHỤ LỤC HÌNH ẢNH

CHƯƠNG 1 – GIỚI THIỆU THUẬT TOÁN

1.1 Hàm inRange

1.1.1 Cú pháp hàm inRange trong OpenCV:

resultarray = inRange(sourcearray, upperboundarray, lowerboundarray)

- Sourcearray là mảng có các phần tử được so sánh với các mảng đại diện cho giới hạn trên và giới hạn dưới.
- Upperboundarray là mảng bao gồm các phần tử đại diện cho các giới hạn trên.
- Lowerboundarray là mảng bao gồm các phần tử đại diện cho các giới hạn dưới.
- Resultarray là mảng đại diện cho các phần tử bằng 255 hoặc 0 được trả về từ hàm inRange().

1.1.2 Hoạt động của hàm inRange() trong OpenCV

- Để xác định xem các phần tử của một mảng đã cho có nằm giữa các phần tử của hai mảng đại diện cho giới hạn trên và giới hạn dưới hay không, chúng tôi sử dụng một hàm gọi là hàm inRange() trong OpenCV.
- Hàm inRange() trong OpenCV nhận ba tham số cụ thể là mảng nguồn, dãy trên và dãy dưới.
- Tham số sourcearray là mảng có các phần tử được so sánh với hai mảng đại diện cho giới hạn trên và giới hạn dưới.
- Tham số upperboundsarray là mảng bao gồm các phần tử đại diện cho các cận trên.
- Tham số Lowerboundsarray là mảng bao gồm các phần tử đại diện cho các giới hạn dưới.
- Hàm inRange() trả về một mảng bao gồm các phần tử bằng 255 nếu các phần tử của mảng nguồn nằm giữa các phần tử của hai mảng đại diện cho giới hạn trên và giới hạn dưới.

- Hàm inRange() trả về một mảng bao gồm các phần tử bằng 0 nếu các phần tử của mảng nguồn nằm giữa các phần tử của hai mảng đại diện cho cận trên và cận dưới.

1.2 Hàm bitwise_and

1.2.1 Cú pháp hàm bitwise_and trong OpenCV

bitwise_and(source1_array, source2_array, destination_array, mask)

- source1_array là mảng tương ứng với hình ảnh đầu vào đầu tiên mà thao tác bit và thao tác sẽ được thực hiện trên đó,
- source2_array là mảng tương ứng với hình ảnh đầu vào thứ hai mà thao tác bit và thao tác sẽ được thực hiện trên đó,
- Destination_array là mảng kết quả bằng cách thực hiện thao tác bit trên mảng tương ứng với hình ảnh đầu vào đầu tiên và mảng tương ứng với hình ảnh đầu vào thứ hai và
- Mask là thao tác mặt nạ được thực hiện trên hình ảnh kết quả và nó là tùy chọn.

1.2.2 Hoạt động của hàm bitwise_and trong Opencv

- Để có thể thực hiện kết hợp bit khôn ngoan của hai mảng tương ứng với hai hình ảnh trong OpenCV, chúng tôi sử dụng toán tử bitwise_and.
- Để có thể sử dụng toán tử bitwise_and trong chương trình của chúng tôi, chúng tôi phải nhập mô-đun cv2.
- Các hình ảnh có mảng sẽ được kết hợp bằng toán tử bitwise_and được đọc bằng hàm imread().
- Sau đó, các mảng tương ứng của những hình ảnh đó được chuyển đến toán tử bitwise_and.
- Toán tử bitwise_and trả về một mảng tương ứng với hình ảnh kết quả từ việc hợp nhất hai hình ảnh đã cho.

- Hoạt động của bitwise_and chỉ có thể được thực hiện trên các hình ảnh có cùng kích thước.

1.3 Hàm cvtColor

1.3.1 Cú pháp hàm cvtColor trong OpenCV

cvtColor(image, code)

- Image dùng để truyền hình ảnh vào
- Code là mã chuyển đổi không gian màu

1.3.2 Hoạt động của hàm cvtColor trong Opencv

- Trong khi xử lý các vấn đề trong lĩnh vực thị giác máy tính, đôi khi cần phải chuyển đổi không gian màu của một không gian màu nhất định sang không gian màu khác.
- Bất cứ khi nào có nhu cầu chuyển đổi không gian màu của một hình ảnh nhất định sang không gian màu khác, chúng ta sử dụng một hàm gọi là hàm cvtColor() trong OpenCV.
- Hàm cvtColor() trong OpenCV nhận hai tham số là hình ảnh và mã trong đó hình ảnh là hình ảnh có không gian màu sẽ được chuyển đổi sang không gian màu khác và mã đại diện cho mã chuyển đổi màu.
- Có hơn 150 mã chuyển đổi không gian màu có sẵn trong OpenCV.
- Một số mã chuyển đổi không gian màu là cv2.COLOR_BGR2GRAY, cv2.COLOR_BGR2HSV, v.v.
- Hàm cvtColor() trả về hình ảnh với không gian màu đã thay đổi.

1.4 Hàm findContours

1.4.1 Cú pháp hàm findContours trong OpenCV

findContours (InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset = Point())

- Image dùng để truyền hình ảnh vào
- Contours là đường viền được lưu trữ dưới dạng vector
- Hierarchy phân tích cấp bật các đường viền trong hệ thống
- Mode kích hoạt đặt biệt để truy xuất đường viền
- Method dùng để mô tả phương pháp gần đúng cho đường viền hình ảnh
- Point offset tham số tùy chọn của đường viền

1.4.2 Hoạt động của hàm findContours trong Opencv

- Khi máy tính được tạo để phát hiện các cạnh của hình ảnh đầu vào, nó sẽ tìm các điểm cụ thể là có thông báo khác biệt đáng kể về cường độ màu, sau đó chỉ cần bật các pixel đó. Một sự khác biệt rõ rệt được nhận thấy khi hệ thống được hướng dẫn thực hiện đường viền.
- Đường viền về cơ bản là một tập hợp trừu tượng gồm các phân đoạn và điểm tương ứng với hình dạng phản chiếu của các đối tượng có trong ảnh đã được xử lý qua hệ thống. như là kết quả của việc này; chúng tôi có khả năng điều khiển đường viền trong các chương trình mà chúng đang được truy cập.
- Điều này có thể được thực hiện theo nhiều cách, chẳng hạn như đếm số lượng đường viền trong một hình ảnh và sau đó sử dụng số đó để phân loại hình dạng đối tượng, để phân đoạn hình ảnh hoặc cắt đối tượng khỏi hình ảnh đang được xử lý, v.v. chức năng.

1.5 Hàm drawContours

1.5.1 Cú pháp hàm drawContours trong OpenCV

drawContours(source_image, contours, contours_ID, contour_color, contour_thickness)

- cource_image là hình ảnh mà các đường viền phải được vẽ trên đó.
- contours là các đường viền được trích xuất từ một hình ảnh nhất định bằng cách sử dụng hàm findcontours().
- contours_ID là tham số xác định đường viền sẽ được vẽ và giá trị đường viền_ID âm cho biết rằng tất cả các đường viền phải được vẽ.
- contour_color đại diện cho màu đường viền.
- contour_thickness đại diện cho độ dày của các đường tạo thành đường viền.

1.5.2 Hoạt động của hàm drawContours trong Opencv

- Để thực hiện phân tích hình ảnh như phân tích hình dạng, phát hiện kích thước, phát hiện đối tượng, v.v., chúng tôi sử dụng các đường viền trong OpenCV.
- Đường bao là các điểm xung quanh ranh giới của một hình nhất định được tạo thành bằng cách nối chúng lại với nhau thành các đường.
- Các đường viền trong một hình ảnh nhất định có thể được trích xuất bằng cách sử dụng một hàm gọi là hàm findcontours() trong OpenCV.
- Hàm findcontours() trả về số lượng đường viền trong một hình ảnh nhất định.
- Để vẽ các đường viền trong một hình ảnh nhất định, chúng ta sử dụng một hàm gọi là hàm drawcontours().

- Hàm drawcontours() lấy các đường viền được trích xuất bằng cách sử dụng hàm findcontours() và vẽ các đường viền trong hình ảnh đã cho.
- Hàm drawcontours() trả về hình ảnh với các đường viền được vẽ trên đó.

1.6 cv2.filter2D(img, -1, kernel)

1.6.1 Cú pháp hàm filter2D trong OpenCV

filter2d(source_image, depth, kernel)

- source_image là hình ảnh mà thao tác lọc phải được thực hiện trên đó,
- depth là một biến số nguyên chỉ định độ sâu trong hình ảnh đầu ra,
- kernel là một ma trận đại diện cho hạt nhân tích chập.

1.6.2 Hoạt động của hàm filter2D trong Opencv

- Kỹ thuật cải thiện một số tính năng nhất định trong ảnh hoặc loại bỏ một số tính năng khỏi ảnh được gọi là lọc trong OpenCV.
- Quá trình lọc hình ảnh cho phép làm mờ hình ảnh, làm sắc nét hình ảnh, phát hiện các cạnh trong hình ảnh, v.v.
- OpenCV cung cấp một hàm gọi là hàm filter2d() để thực hiện lọc ảnh.
- Hàm filter2d() kết hợp một hạt nhân với một hình ảnh.
- Tích chập đang nhân hai mảng bao gồm các số có cùng kích thước để tạo ra một mảng thứ ba bao gồm các số có cùng kích thước.
- Nhân là một ma trận nhỏ được sử dụng để làm mờ ảnh, làm sắc nét ảnh, phát hiện các cạnh trong ảnh, v.v. trong xử lý ảnh.
- Hàm filter2d() hoạt động bằng cách đặt nhân trên tất cả các pixel, sau đó tìm giá trị trung bình của tất cả các pixel để biến nó thành giá trị pixel trung tâm và quá trình này lặp lại cho tất cả các pixel trong ảnh.

- Hàm filter2d() trả về một hình ảnh sau khi tăng cường một số tính năng nhất định trong hình ảnh hoặc sau khi xóa một số tính năng khỏi hình ảnh.

1.7 Hàm rectangle

1.7.1 Cú pháp hàm rectangle trong OpenCV

cv2 . rectangle (image, start _ point, end _ point, color, thickness)

- **Image :** Tham số đại diện cho hình ảnh gốc phải được xử lý và phải bao gồm một hình chữ nhật.
- **start_point:** Tham số start_point đại diện cho tọa độ bắt đầu có trong hình chữ nhật. Các bộ giá trị được trình bày đối với các tọa độ hình chữ nhật (nghĩa là các giá trị tọa độ X và Y).
- **end_point :** Tham số end_point biểu thị tọa độ kết thúc có trong hình chữ nhật. Các bộ giá trị được trình bày đối với các tọa độ hình chữ nhật (nghĩa là các giá trị tọa độ X và Y).
- **color :** Tham số biểu thị màu sẽ có trên đường viền của hình chữ nhật đang được vẽ. Ví dụ: nếu người dùng đang cố gắng tạo một hình vuông màu xanh thì bộ dữ liệu sau phải được chuyển: 255, 0, 0 để tạo màu - BLUE.
- **thickness:** Tham số thể hiện độ dày của đường viền đang được người dùng vẽ (hình chữ nhật). Độ dày của đường được xác định bởi kích thước pixel hoặc px. Độ dày của đường được tạo là -1 px sẽ lấp đầy đường viền hoặc hình chữ nhật bằng màu do người dùng chỉ định.

1.7.2 Hoạt động của hàm drawContours trong Opencv

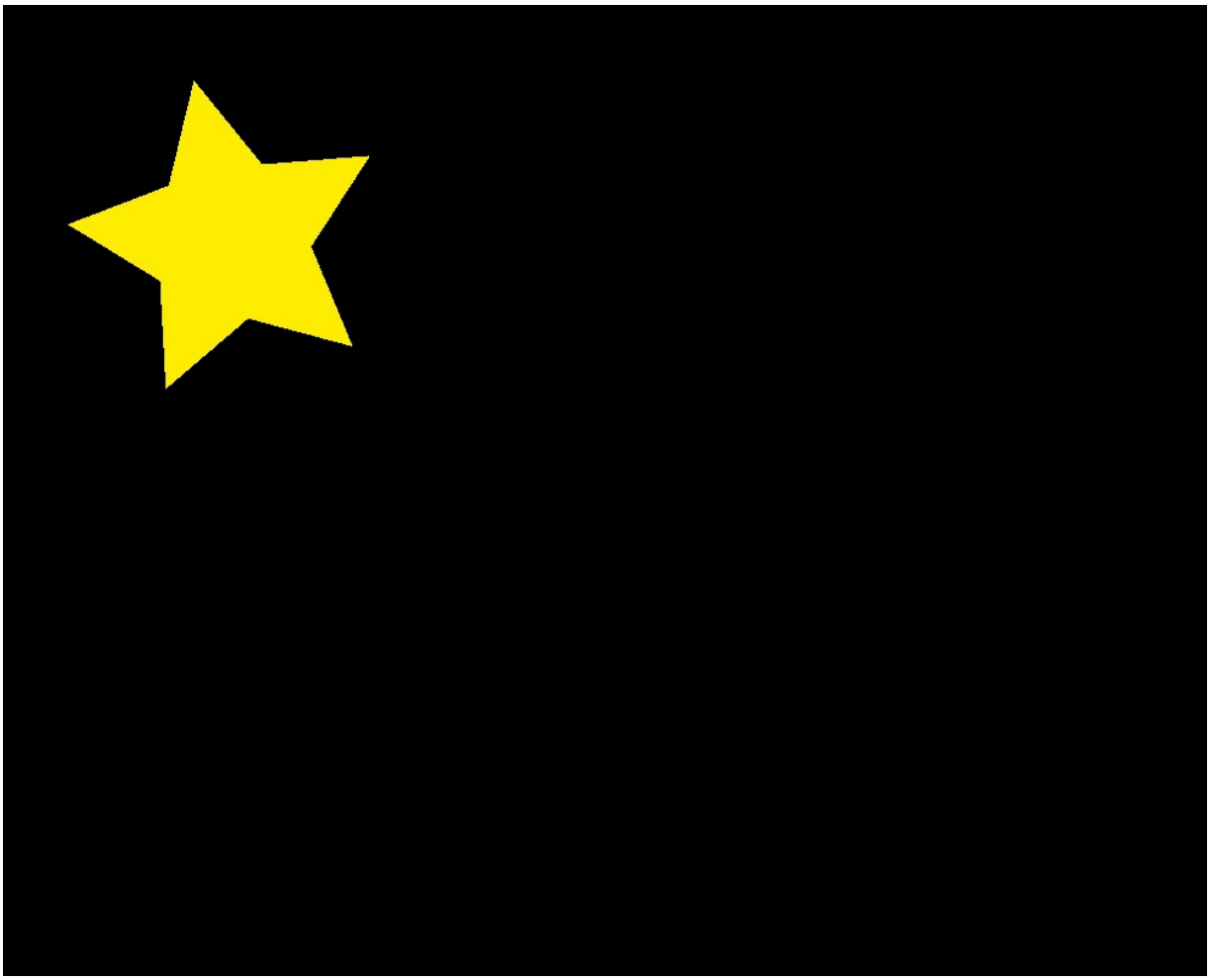
- Hình ảnh đầu ra đã được đưa ra một đường viền hoặc hình chữ nhật được đưa vào sau khi chức năng được thực thi trên hình ảnh gốc.

CHƯƠNG 2 – KẾT QUẢ

```
#frame
def saveimg(filename, image):
    directory = r'C:\Users\zhang\Desktop\New folder'
    os.chdir(directory)
    cv2.imwrite(filename, image)
```

Hình 1. Code lưu hình ảnh về thư mục có đường dẫn là "C:\Users\zhang\Desktop\New folder"

1.1. Câu a



Hình 2. Xuất ra ngôi sao vàng trong ảnh

```
def ay():
    lower_y = np.array([25, 155, 155])
    upper_y = np.array([50, 255, 255])
    masky = cv2.inRange(hsv, lower_y, upper_y)
    resulty = cv2.bitwise_and(img, img, mask = masky)
    filenamey = 'outputyellow.jpg'
    cv2.imshow('yellow', resulty)
    saveimg(filenamey, resulty)
```

Hình 3. Code xuất ra ngôi sao màu vàng trong hình ảnh

Hình 4. Xuất ra ngôi sao cam trong hình ảnh

```
def ao():
    lower_o = np.array([0, 155, 155])
    upper_o = np.array([25, 255, 255])
    masko = cv2.inRange(hsv, lower_o, upper_o)
    resulto = cv2.bitwise_and(img, img, mask = masko)
    filenameo = 'outputorange.jpg'
    cv2.imshow('orange', resulto)
    saveimg(filenameo, resulto)
```

Hình 5. Code xuất ra ngôi sao màu cam trong ảnh

Hình 6. Xuất ra ngôi sao hồng trong ảnh

```
def ar():
    lower_r = np.array([155, 155, 155])
    upper_r = np.array([180, 255, 255])
    maskr = cv2.inRange(hsv, lower_r, upper_r)
    resultr = cv2.bitwise_and(img, img, mask = maskr)
    os.chdir(directory)
    filenamer = 'outputred.jpg'
    cv2.imshow('red', resultr)
    saveimg(filenamer, resultr)
```

Hình 7. Code xuất ra ngôi sao màu hồng trong ảnh

Hình 8. Xuất ra ngôi sao xanh dương trong ảnh

```
def ab():
    lower_b = np.array([100, 155, 155])
    upper_b = np.array([155, 255, 255])
    maskb = cv2.inRange(hsv, lower_b, upper_b)
    resultb = cv2.bitwise_and(img, img, mask = maskb)
    filenameb = 'outputblue.jpg'
    cv2.imshow('blue', resultb)
    saveimg(filenameb, resultb)
```

Hình 9. Code xuất ra ngôi sao màu xanh dương trong ảnh

Hình 10. Xuất ra ngôi sao xanh lá trong ảnh

```
def ag():
    lower_g = np.array([55, 155, 100])
    upper_g = np.array([100, 255, 255])
    maskg = cv2.inRange(hsv, lower_g, upper_g)
    resultg = cv2.bitwise_and(img, img, mask = maskg)
    filenameg = 'outputgreen.jpg'
    cv2.imshow('green', resultg)
    saveimg(filenameg, resultg)
```

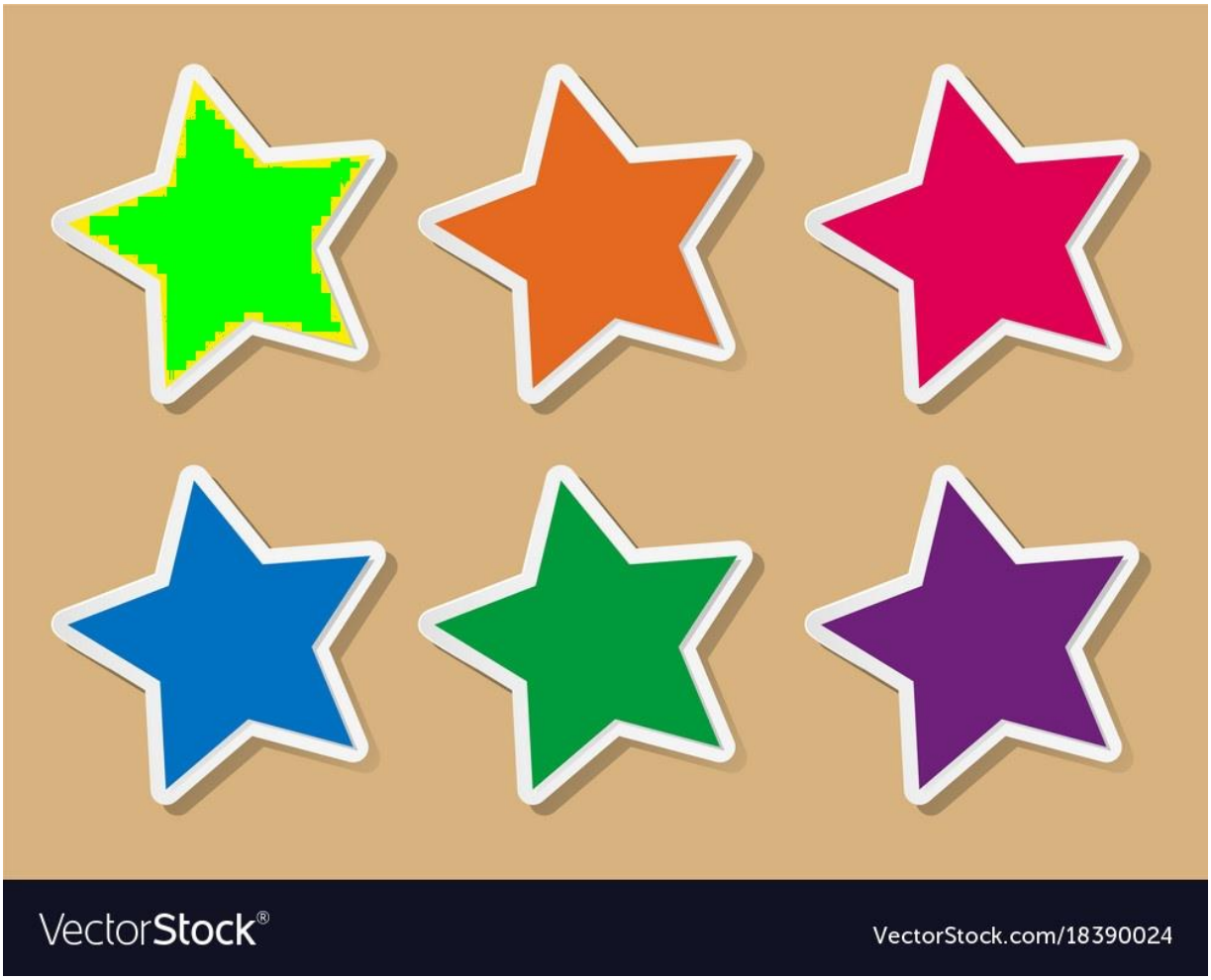
Hình 11. Code xuất ra ngôi sao màu xanh lá trong ảnh

Hình 12. Xuất ra ngôi sao tím trong ảnh

```
def ap():
    lower_v = np.array([100, 155, 100])
    upper_v = np.array([155, 255, 155])
    maskv = cv2.inRange(hsv, lower_v, upper_v)
    resultv = cv2.bitwise_and(img, img, mask = maskv)
    filenamev = 'outputviolet.jpg'
    cv2.imshow('violet', resultv)
    saveimg(filenamev, resultv)
```

Hình 13. Code xuất ra ngôi sao màu tím trong ảnh

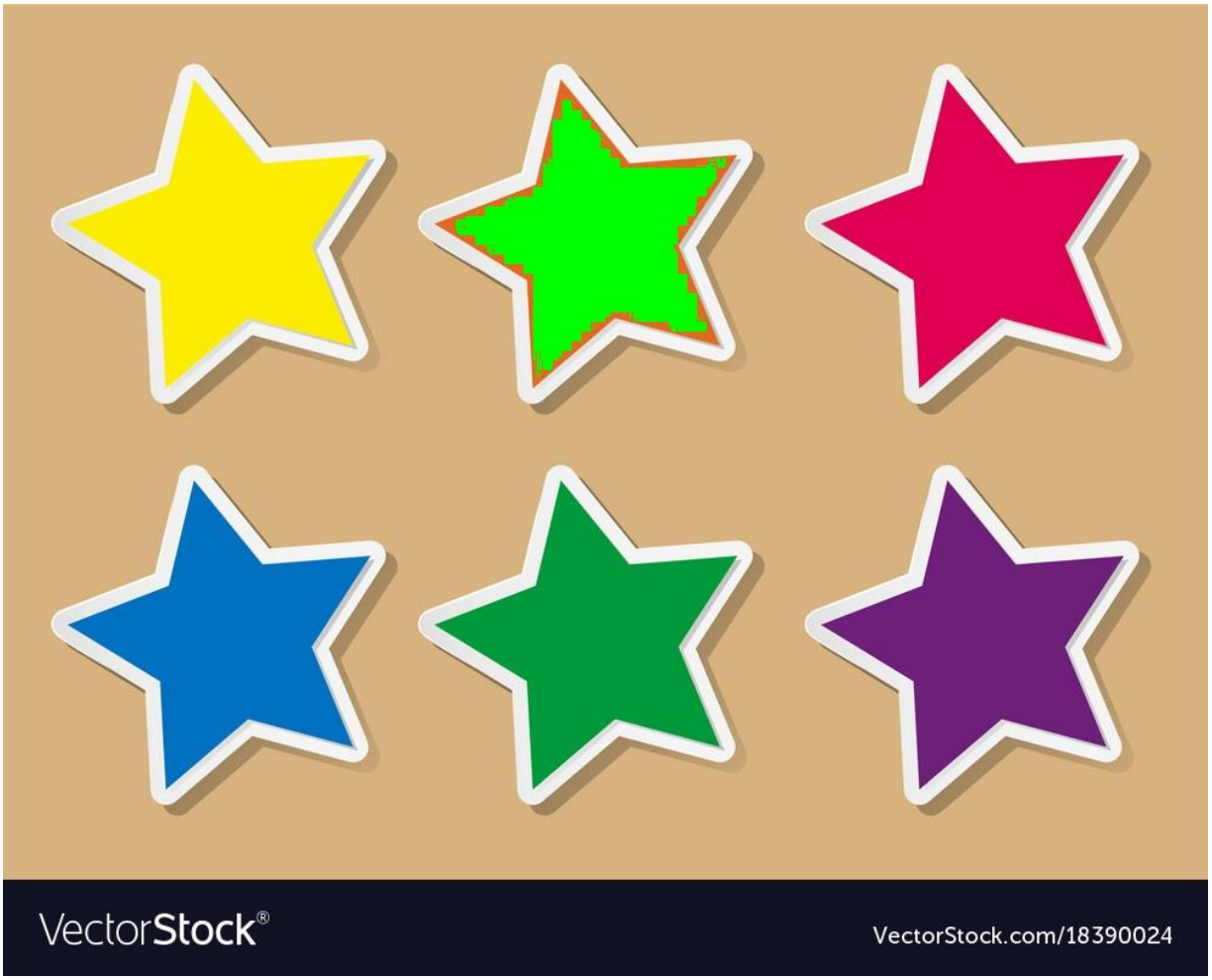
1.2. Câu b



Hình 14. Tô màu xanh lá cho ngôi sao màu vàng

```
def b1():
    img[np.where((img==[1,237,254]).all(axis=2))]=[0,255,0]
    filename1 ='Image1.jpg'
    cv2.imshow('Image1', img)
    saveimg(filename1, img)
```

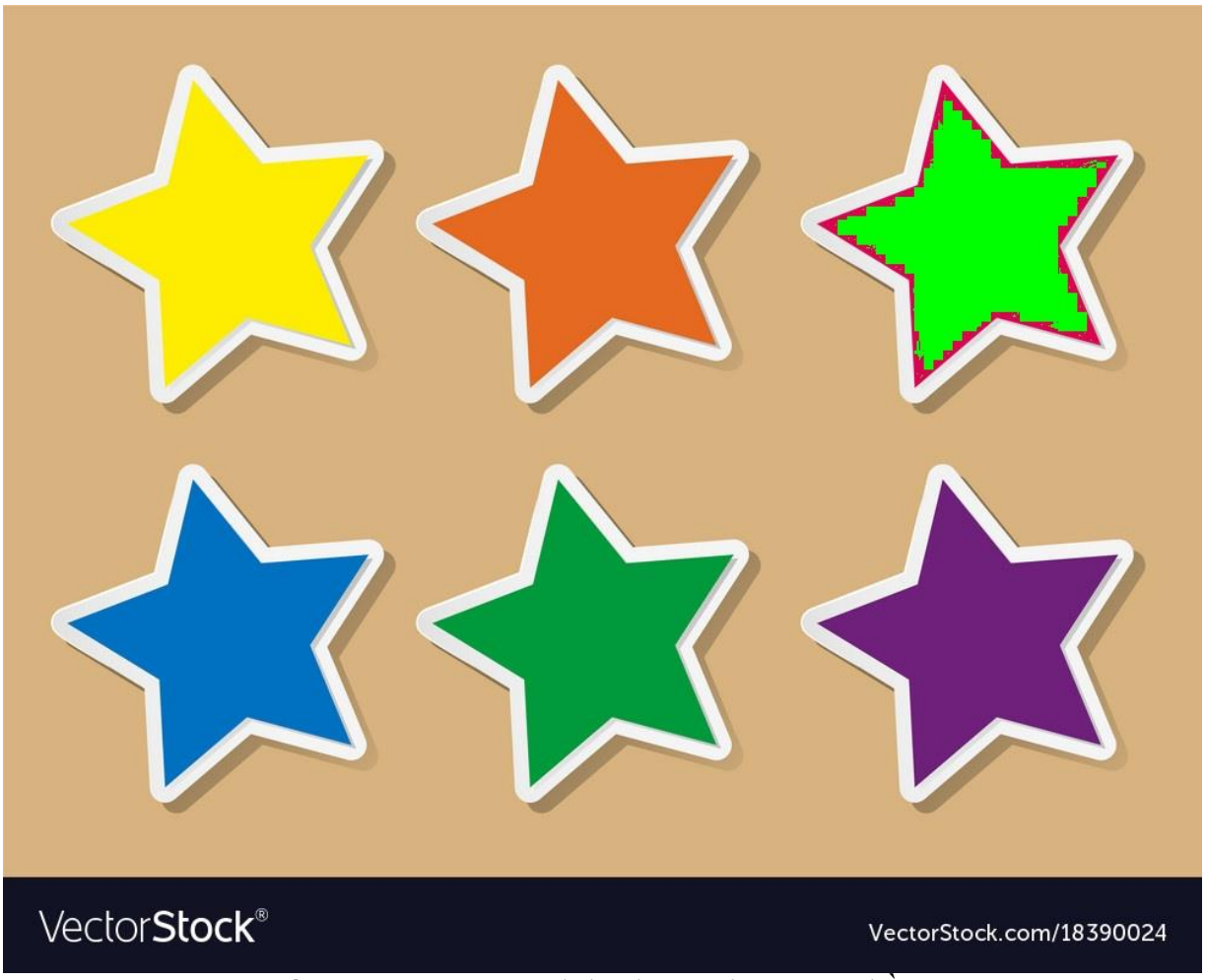
Hình 15. Code tô màu xanh lá cho ngôi sao màu vàng



Hình 16. Tô màu xanh lá cho ngôi sao màu cam

```
def b2():
    img[np.where((img==[33,105,229]).all(axis=2))]=[0,255,0]
    filename2 ='Image2.jpg'
    cv2.imshow('Image2', img)
    saveimg(filename2, img)
```

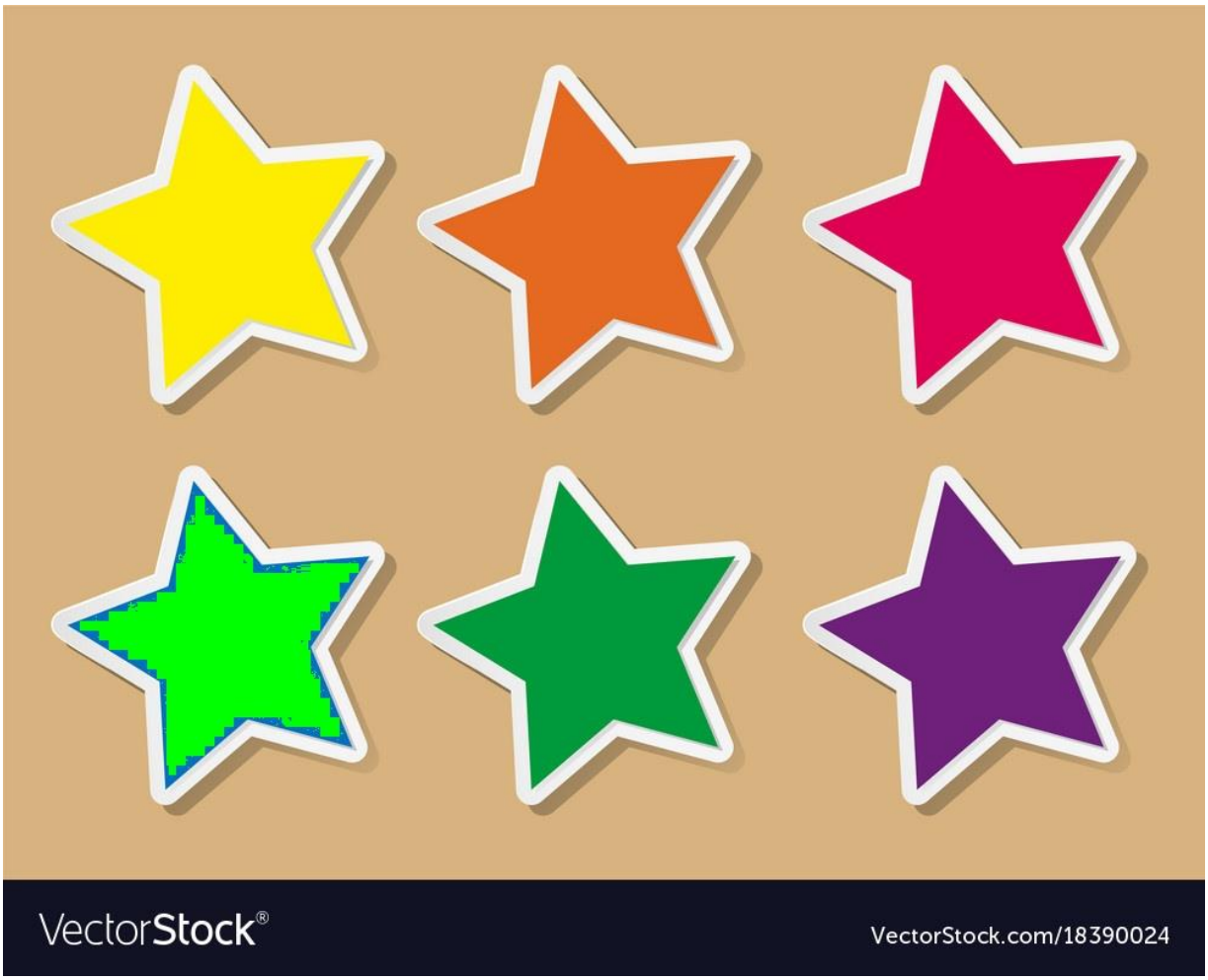
Hình 17. Code tô màu xanh lá cho ngôi sao màu cam



Hình 18. Tô màu xanh lá cho ngôi sao màu hồng

```
def b3():
    img[np.where((img==[83,0,223]).all(axis=2))]=[0,255,0]
    filename3 ='Image3.jpg'
    cv2.imshow('Image3', img)
    saveimg(filename3, img)
```

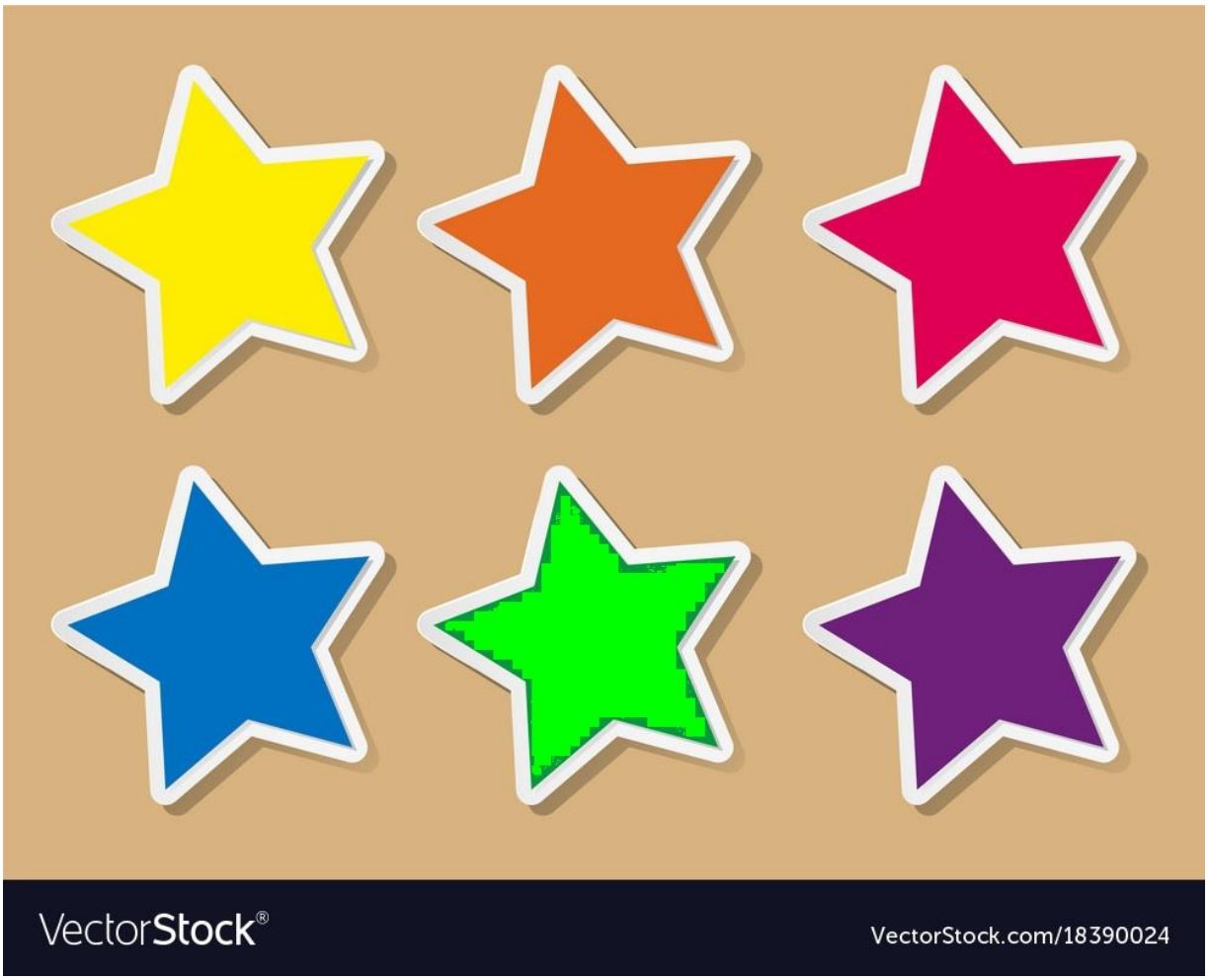
Hình 19. Code tô màu xanh lá cho ngôi sao màu hồng



Hình 20. Tô màu xanh lá cho ngôi sao màu xanh dương

```
def b4():
    img[np.where((img==[193,113,0]).all(axis=2))]=[0,255,0]
    filename4 ='Image4.jpg'
    cv2.imshow('Image4', img)
    saveimg(filename4, img)
```

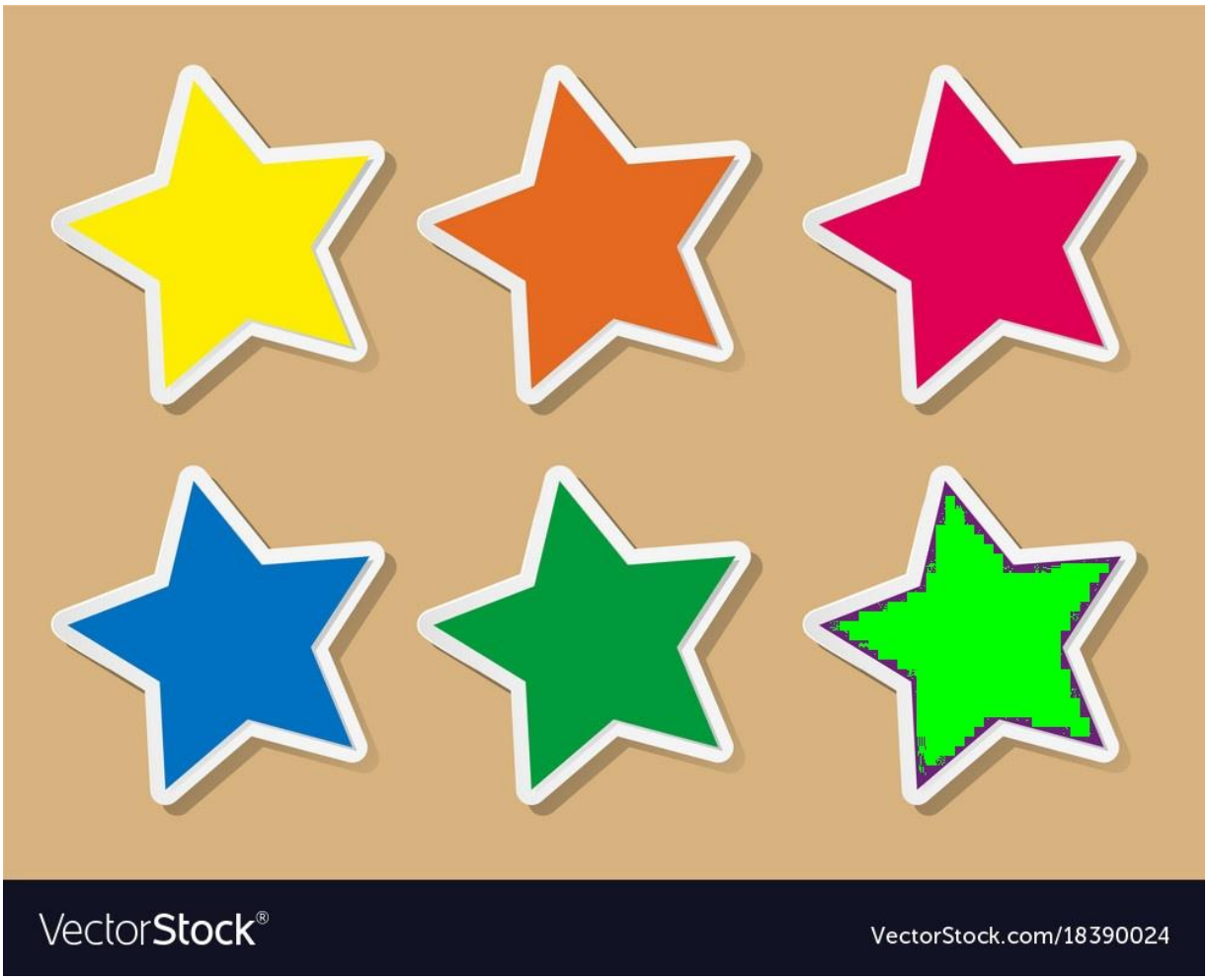
Hình 21. Code tô màu xanh lá cho ngôi sao màu xanh dương



Hình 22. Tô màu xanh lá cho ngôi sao màu xanh lá

```
def b5():
    img[np.where((img==[60,154,0]).all(axis=2))]=[0,255,0]
    filename5 ='Image5.jpg'
    cv2.imshow('Image5', img)
    saveimg(filename5, img)
```

Hình 23. Code tô màu xanh lá cho ngôi sao màu xanh lá

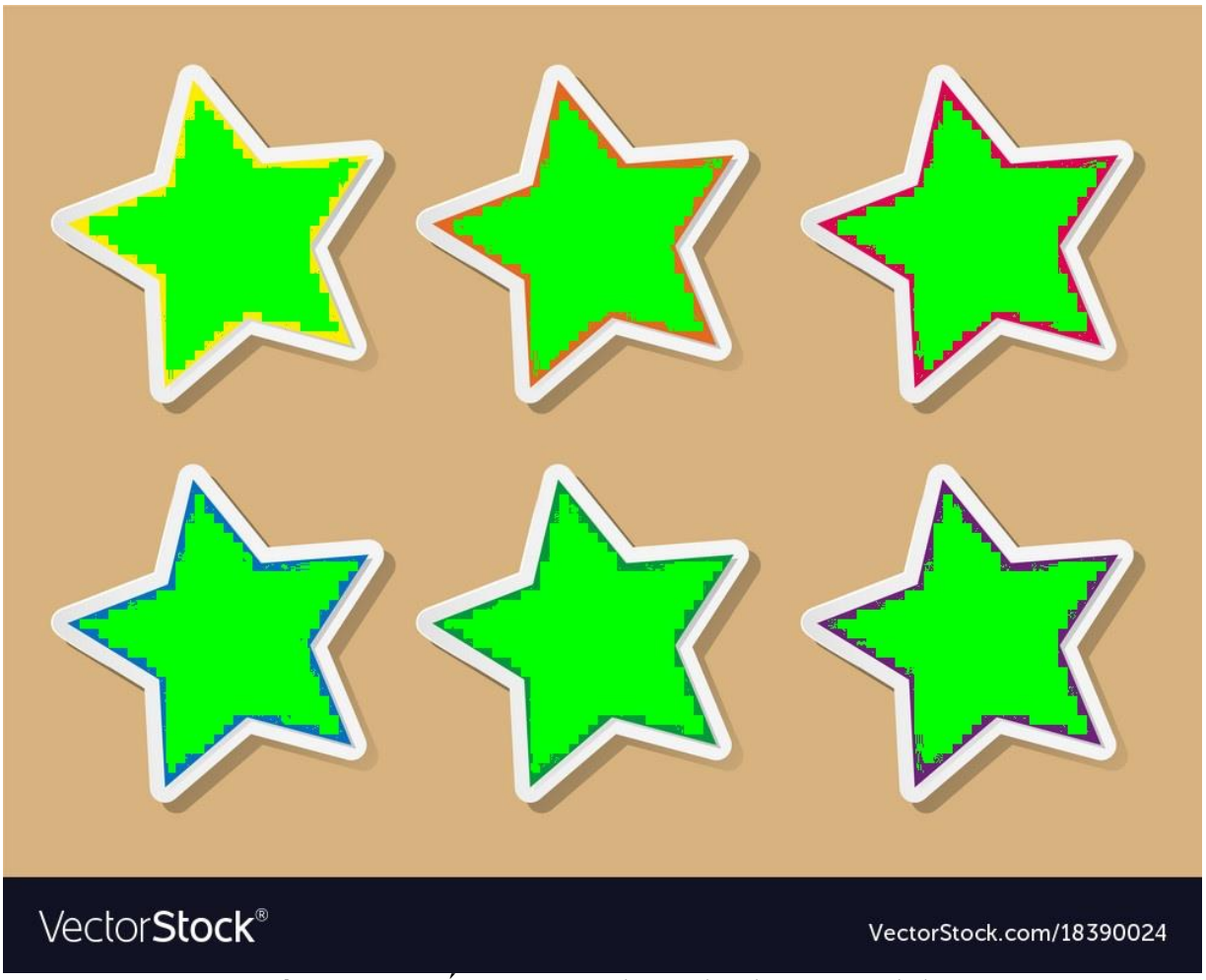


Hình 24. Tô màu xanh lá cho ngôi sao màu tím

```
def b6():
    img[np.where((img==[123,32,111]).all(axis=2))]=[0,255,0]
    filename6 ='Image6.jpg'
    cv2.imshow('Image6', img)
    saveimg(filename6, img)
```

Hình 25. Code tô màu xanh lá cho ngôi sao màu tím

1.3. Câu c



Hình 26. Tô tất cả các ngôi sao thành màu xanh lá

```
def c():
    img[np.where((img==[1,237,254]).all(axis=2))]=[0,255,0]
    img[np.where((img==[33,105,229]).all(axis=2))]=[0,255,0]
    img[np.where((img==[83,0,223]).all(axis=2))]=[0,255,0]
    img[np.where((img==[193,113,0]).all(axis=2))]=[0,255,0]
    img[np.where((img==[60,154,0]).all(axis=2))]=[0,255,0]
    img[np.where((img==[123,32,111]).all(axis=2))]=[0,255,0]
    filenamec ='Imagec.jpg'
    cv2.imshow('Imagec', img)
    saveimg(filenamec, img)
```

Hình 27. Code tô tất cả các ngôi sao thành màu xanh lá

1.4. Câu c

```
def c():
    img[np.where((img==[1,237,254]).all(axis=2))]=[0,255,0]
    img[np.where((img==[33,105,229]).all(axis=2))]=[0,255,0]
    img[np.where((img==[83,0,223]).all(axis=2))]=[0,255,0]
    img[np.where((img==[193,113,0]).all(axis=2))]=[0,255,0]
    img[np.where((img==[60,154,0]).all(axis=2))]=[0,255,0]
    img[np.where((img==[123,32,111]).all(axis=2))]=[0,255,0]
    filenamec ='Imagec.jpg'
    cv2.imshow('Imagec', img)
    cv2.imwrite(filenamec, img)
```

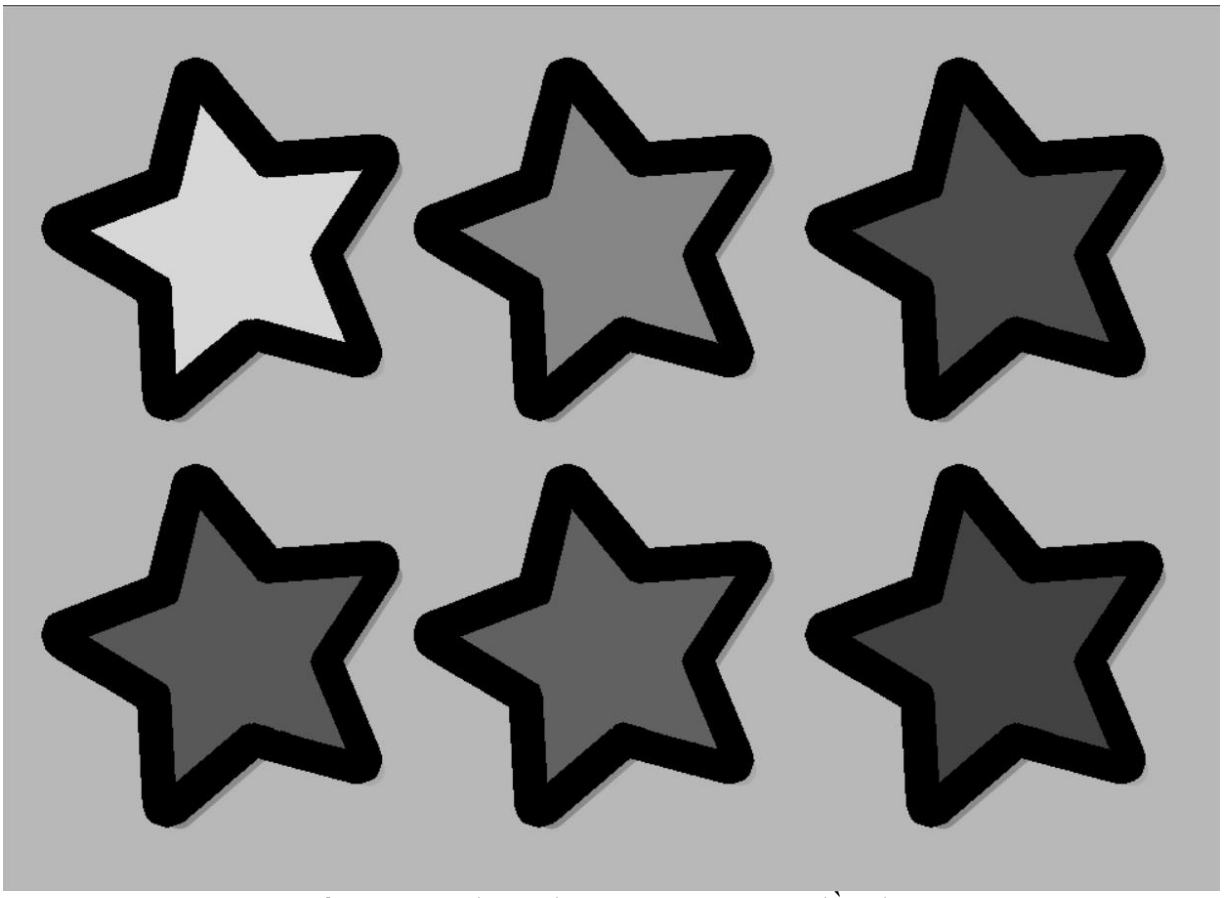
1.5. Câu d

```
def d():
    imageray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    ret,thresh=cv2.threshold(imageray,220,255, 0)
    contours,hierarchy = cv2.findContours(thresh,cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    print(""+str(len(contours)))
    print(contours[0])

    cv2.drawContours(img,contours, -1,(0,0,0),15)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cv2.imshow('gray', gray)
    filenamed = 'outputgray.jpg'
    cv2.imwrite(filenamed, gray)
```

Hình 28 Code xuất ra hình ngôi màu xám có viền đen



Hình 29 Hình ngôi sao màu xám có viền đen

1.6. Câu e

```
def e():
    imageray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    ret,thresh=cv2.threshold(imageray,220,255, 0)
    contours,hierarchy = cv2.findContours(thresh,cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    print(""+str(len(contours)))
    print(contours[0])
    cv2.drawContours(img,contours, -1,(0, 0, 0),2)
    h, w, num_c = img.shape
    segmask = np.zeros((h, w, num_c), np.uint8)
    stencil = np.zeros((h, w, num_c), np.uint8)
    for c in contours:
        cv2.drawContours(segmask, [c], 0, (255,255,255), -1)
        cv2.drawContours(stencil, [c], 0, (255, 0, 0), -1)
        stencil[np.where((stencil==[255,0,0]).all(axis=2))] = [0, 0, 0]
        stencil[np.where((stencil==[0,0,0]).all(axis=2))] = [255,255,255]
    mask = cv2.bitwise_xor(stencil, segmask)
    img = cv2.cvtColor(mask, cv2.COLOR_BGR2RGB)
    cv2.imshow('e', img)
    filenameh = 'e.jpg'
    saveimg(filenameh, img)
```

Hình 30 Code xuất ra ngôi sao màu đen nền trắng

Hình 31 Hình ngôi sao đen viền trắng

1.7. Câu f

```
def f():
    imageray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    ret,thresh=cv2.threshold(imageray,220,255, 0)
    contours,hierarchy = cv2.findContours(thresh,cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    print(""+str(len(contours)))
    print(contours[0])

    cv2.drawContours(img,contours, -1,(0,0,255),2)
    filenamef = 'contours.jpg'
    cv2.imshow('contours', img)
    cv2.imwrite(filenamef, img)
```

Hình 32Code tô màu đỏ cho viền ngôi sao

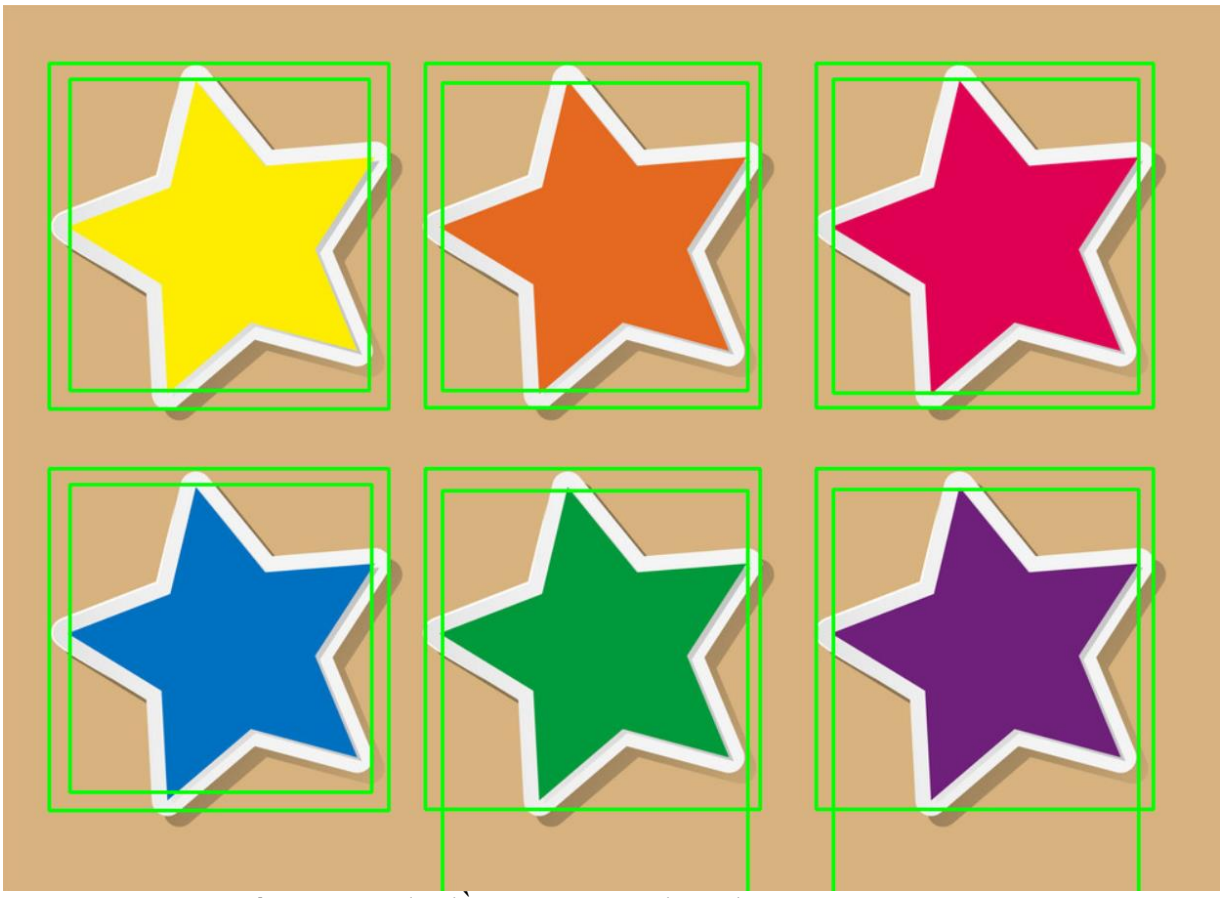
Hình 33 Hình Ngôi sao viền đỏ

1.8. Câu g

```
def g():
    start_point = (38, 48)
    end_point = (316, 331)
    color = (0,255, 0)
    thickness = 2
    imgf = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (55, 61)
    end_point = (300, 316)
    color = (0,255, 0)
    thickness = 2
    image3 = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (346, 48)
    end_point = (620, 330)
    color = (0,255, 0)
    thickness = 2
    image3 = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (360, 64)
    end_point = (610, 316)
    color = (0,255, 0)
    thickness = 2
    image4 = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (666, 48)
    end_point = (942, 330)
    color = (0,255, 0)
    thickness = 2
    image5 = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (680,61)
    end_point = (930, 318)
    color = (0,255, 0)
    thickness = 2
    image6 = cv2.rectangle(img, start_point, end_point, color, thickness)
    start_point = (38,380)
    end_point = (316, 660)
```

```
image7 = cv2.rectangle(img, start_point, end_point, color, thickness)
start_point = (55, 393)
end_point = (302, 645)
color = (0,255, 0)
thickness = 2
image8 = cv2.rectangle(img, start_point, end_point, color, thickness)
start_point = (346, 380)
end_point = (620, 659)
color = (0,255, 0)
thickness = 2
image9 = cv2.rectangle(img, start_point, end_point, color, thickness)
start_point = (360, 398)
end_point = (610, 961)
color = (0,255, 0)
thickness = 2
image10 = cv2.rectangle(img, start_point, end_point, color, thickness)
start_point = (666, 380)
end_point = (942, 659)
color = (0,255, 0)
thickness = 2
image11 = cv2.rectangle(img, start_point, end_point, color, thickness)
start_point = (680,397)
end_point = (930, 961)
color = (0,255, 0)
thickness = 2
image12 = cv2.rectangle(img, start_point, end_point, color, thickness)
filenameg = 'square.jpg'
cv2.imshow('square', img)
cv2.imwrite(filenameg, img)
```

Hình 34 Code viền vuông ngoài ngôi sao



Hình 35 Hình viền vuông ngoài ngôi sao

1.9. Câu h

```python
def h():
    kernel = np.array([[-1,-1,-1],
                       [-1, 9,-1],
                       [-1,-1,-1]])
    sharpened = cv2.filter2D(img, -1, kernel)
    font = cv2.FONT_HERSHEY_SIMPLEX
    org = (15, 700)
    fontScale = 1
    color = (0, 255, 255)
    thickness1 = 2
    sharpened = cv2.putText(img, '51900119-Le Thanh Dang Khoa', org, font, fontScale, color, thickness1, cv2.LINE_AA)
    cv2.imshow('Image Sharpening', sharpened)
    filenameh = 'Name.jpg'
    cv2.imwrite(filenameh, img)
```

Hình 36 Code nàm nét ảnh và hiện tên

Hình 37 Ảnh đã làm nét và tên sinh viên

TÀI LIỆU THAM KHẢO

Tiếng Anh

1. https://www.educba.com/opencv-inrange/
2. https://www.educba.com/opencv-bitwise_and/
3. https://www.educba.com/opencv-cvtcolor/
4. https://www.educba.com/opencv-findcontours/
5. https://www.educba.com/opencv-drawcontours/
6. https://www.educba.com/opencv-filter2d/
7. https://www.educba.com/opencv-rectangle/