**Bài thực hành số 4**

**Các kỹ thuật phân đoạn ảnh trong OpenCV-Python**

**Nội dung:**

1. Giới thiệu.
2. Phân đoạn ảnh là gì?
   * 1. Các hướng tiếp cận phân đoạn ảnh.
3. Các khái niệm tiên quyết (pre-requisite concepts)
   * 1. Thuật toán k-mean
     2. Phát hiện đường viền (contourn detection)
     3. Masking
     4. Color detection
4. Phân đoạn ảnh trong OpenCV Python
5. Phân đoạn ảnh sử dung k-Mean
6. Phân đoạn ảnh sử dụng Contourn Detection
7. Phân đoạn ảnh sử dụng Masking
8. Phân đoạn ảnh sử dụng Color detection
9. Tổng kết.

**I. Giới thiệu:**

Trong bài học này, chúng ta tìm hiểu về cách thực hiện phân đoạn hình ảnh trong OpenCV Python với 4 kỹ thuật khác nhau. Trước tiên, tìm hiểu xử lý hình ảnh là gì và đề cập đến một số khái niệm tiên quyết. Sau đó chúng ta sẽ lần lượt đi qua các kỹ thuật và triển khai chúng bằng ngôn ngữ Python.

**II. Phân đoạn ảnh là gì?**

A collage of images of cars and buildings

Description automatically generated with low confidence

Phân đoạn ảnh là một tác vụ xử lý ảnh trong đó ảnh được phân đoạn hoặc phân vùng thành nhiều vùng sao cho các pixel trong cùng một vùng có chung các đặc điểm.

Có hai hình thức phân đoạn ảnh:

Phân đoạn cục bộ – Nó liên quan đến một khu vực hoặc vùng cụ thể của hình ảnh.

Phân đoạn toàn cục – Nó liên quan đến việc phân đoạn toàn bộ hình ảnh.

**III. Các loại phương pháp tiếp cận phân đoạn hình ảnh**

1. Phát hiện sự gián đoạn – Đây là phương pháp phân đoạn ảnh thành các khu vực dựa trên sự không liên tục. Đây là lúc tính năng phát hiện cạnh xuất hiện. Sự gián đoạn ở các cạnh được tạo ra do cường độ được nhận biết và sử dụng để thiết lập đường viền khu vực. Ví dụ như lọc biểu đồ và phát hiện đường viền.
2. Phát hiện sự giống nhau – Một phương pháp phân đoạn ảnh thành các phần dựa trên sự giống nhau. Ngưỡng, mở rộng khu vực, tách và hợp nhất khu vực đều được tổng hợp trong phương pháp này. Tất cả chúng đều chia hình ảnh thành các phần có số pixel tương đương nhau. Dựa trên các tiêu chí đã thiết lập, họ chia bức tranh thành một nhóm các cụm có các đặc điểm có thể so sánh được. Ví dụ như phân vùng k-Means, Phát hiện/phân loại màu sắc.
3. Phương pháp tiếp cận mạng nơ-ron – Đối với mục tiêu ra quyết định, các thuật toán phân đoạn dựa trên mạng nơ-ron tái hiện các kỹ thuật học tập của bộ não con người. Cách tiếp cận này được sử dụng rộng rãi trong việc phân đoạn hình ảnh y tế và tách chúng khỏi nền. Một mạng lưới thần kinh được tạo thành từ một số lượng lớn các nút được liên kết, mỗi nút có trọng số riêng.

**IV. Các khái niệm tiên quyết**

Trong phần này, chúng ta sẽ đề cập ngắn gọn một số khái niệm cần thiết trước đó sẽ hữu ích để hiểu các kỹ thuật phân đoạn hình ảnh trong Python.

*i) Thuật toán K-Means*

K-means là một thuật toán phân cụm được sử dụng để nhóm các điểm dữ liệu thành các cụm sao cho các điểm dữ liệu nằm trong cùng một nhóm rất giống nhau về các đặc điểm.

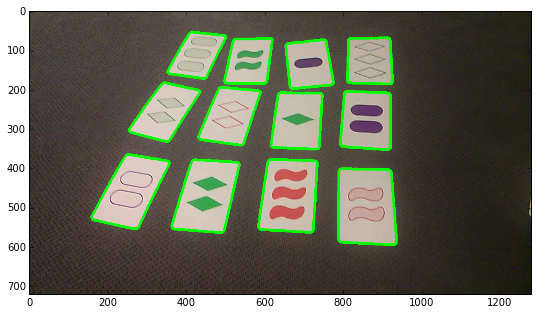
Thuật toán K-means có thể được sử dụng để tìm các nhóm con trong ảnh và gán pixel ảnh cho nhóm con đó, dẫn đến phân đoạn ảnh.



*ii) Phát hiện đường viền*

Đường viền có thể được định nghĩa đơn giản là các đường cong/đa giác được hình thành bằng cách nối các pixel được nhóm lại với nhau theo cường độ hoặc giá trị màu.

OpenCV cung cấp cho chúng ta các chức năng sẵn có để phát hiện các đường viền này trong ảnh. Phát hiện đường viền thường được áp dụng trên ảnh nhị phân (ảnh thang độ xám) sau khi phát hiện cạnh hoặc tạo ngưỡng (hoặc cả hai) đã được áp dụng cho chúng.



*iii) Áp mặt nạ (masking)*

Việc áp dụng các mặt nạ (là các hình ảnh nhị phân chỉ có 0 hoặc 1 làm giá trị pixel) để chuyển đổi một hình ảnh được gọi là tạo mặt nạ. Các pixel (của ảnh) trùng với số 0 trong mặt nạ sẽ bị tắt khi mặt nạ được áp dụng cho nó.



*iv) Phát hiện màu sắc*

Phát hiện và phân loại màu bằng cách sử dụng các giá trị không gian màu RGB của chúng được gọi là phát hiện màu.

Ví dụ :

R G B

Red = (255, 0, 0)

Green = (0, 255, 0)

Blue = (0, 0, 255)

Orange = (255, 165, 0)

Purple = (128, 0, 128)

**V. Phân đoạn hình ảnh trong OpenCV Python**

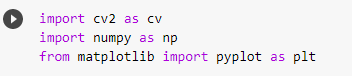
Chúng ta sẽ xem xét 4 kỹ thuật khác nhau sau đây để thực hiện phân đoạn hình ảnh trong OpenCV Python và Scikit Learn.

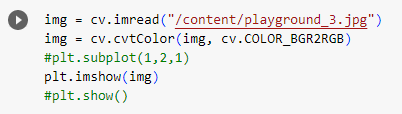
1. Phân đoạn ảnh sử dụng K-Means
2. Phân đoạn ảnh sử dụng phát hiện đường viền
3. Phân đoạn ảnh sử dụng ngưỡng.
4. Phân đoạn ảnh sử dụng mặt nạ màu

*1. Phân đoạn ảnh sử dụng k-mean*

B1. Nhập thư viện và hình ảnh

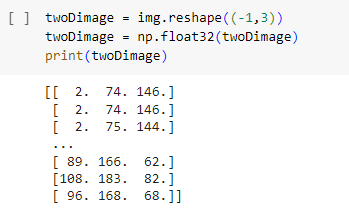
Chúng ta bắt đầu bằng cách nhập các thư viện cần thiết và tải hình ảnh mẫu. Vì OpenCV đọc hình ảnh ở định dạng BGR nên chúng tôi chuyển đổi nó thành RGB và hiển thị hình ảnh.





B2. Tiền xử lý hình ảnh

Tiếp theo, chúng ta định hình lại hình ảnh thành một vectơ 2D, tức là nếu hình ảnh có hình dạng (100,100,3) (chiều rộng, chiều cao, kênh) thì nó sẽ được chuyển đổi thành (10000,3). Tiếp theo, chuyển đổi nó thành kiểu dữ liệu float.



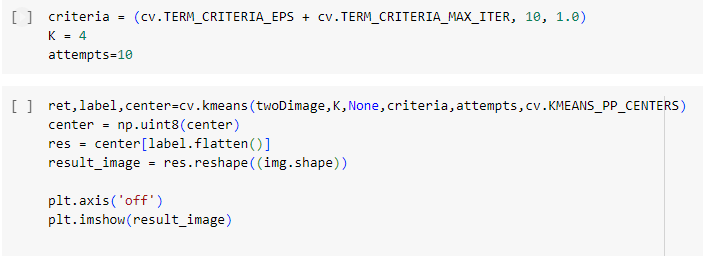
B3. Xác định tham số

Bây giờ chúng tôi xác định các tiêu chí theo đó thuật toán K-means được cho là phân cụm pixel.

Biến 'K' xác định số cụm/nhóm mà pixel có thể thuộc về (Bạn có thể tăng giá trị này để tăng mức độ phân đoạn).

B4. Áp dụng K-Means cho phân đoạn hình ảnh

Biến K khởi tạo ngẫu nhiên K cụm khác nhau và biến 'trung tâm' xác định trung tâm của các cụm này. Khoảng cách của mỗi điểm từ các trung tâm này được tính toán và sau đó chúng được gán cho một trong các cụm. Sau đó, chúng được chia thành các phân đoạn khác nhau tùy theo giá trị của 'biến nhãn' của chúng.



B5. Kết quả phân đoạn ảnh cho các giá trị khác nhau của K

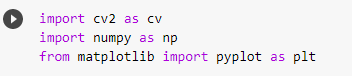
Bằng cách chạy cùng một mã ở trên trong các bước iii) và iv) cho các giá trị khác nhau của K, chúng tôi sẽ thu được kết quả phân đoạn hình ảnh bên dưới -

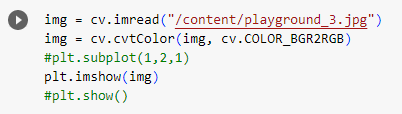


2. Phân đoạn hình ảnh bằng phát hiện đường viền

i) Nhập thư viện và hình ảnh

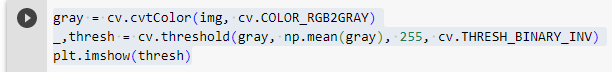
Nhập các thư viện cần thiết và tải hình ảnh mẫu. Vì OpenCV đọc hình ảnh ở định dạng BGR nên chúng tôi chuyển đổi nó thành RGB và hiển thị hình ảnh. Để thuận tiện, chúng tôi cũng thay đổi kích thước hình ảnh thành 256×256 vì chúng tôi sẽ tạo mặt nạ có cùng kích thước trong các bước tiếp theo.





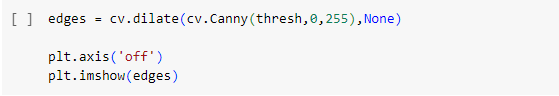
ii) Áp dụng ngưỡng hình ảnh

Bây giờ, chúng ta chuyển đổi hình ảnh thành thang độ xám và sau đó áp dụng ngưỡng, sao cho pixel trên ngưỡng được gán 255 nếu không thì 0. Giá trị ngưỡng được giữ làm giá trị trung bình của tất cả các giá trị pixel của hình ảnh màu xám. Hình ảnh đầu ra cho thấy kết quả của bước này.



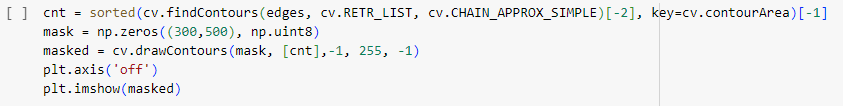
iii) Phát hiện các cạnh

Tiếp theo, chúng ta áp dụng tính năng phát hiện cạnh khôn ngoan cho hình ảnh có ngưỡng trước khi sử dụng chức năng 'cv2.dilate' để làm giãn các cạnh được phát hiện.



iv) Phát hiện các đường viền để tạo mặt nạ

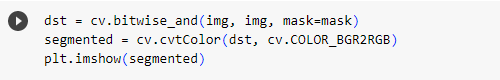
* Sử dụng chức năng tìm đường viền OpenCV để tìm tất cả các vùng mở/đóng trong ảnh và lưu trữ (cnt). Sử dụng chỉ số -1 vì hàm trả về một bộ hai phần tử.
* Chuyển chúng qua hàm đã sắp xếp để truy cập các đường viền lớn nhất trước.
* Tạo mặt nạ không pixel có hình dạng và kích thước bằng hình ảnh gốc.
* Vẽ các đường viền được phát hiện để tạo mặt nạ.



v) Phân vùng

Để chỉ hiển thị các phần được phân đoạn của hình ảnh, chúng tôi thực hiện thao tác AND theo chiều bit trên hình ảnh gốc (img) và mặt nạ (chứa đường viền của các đường viền được phát hiện).

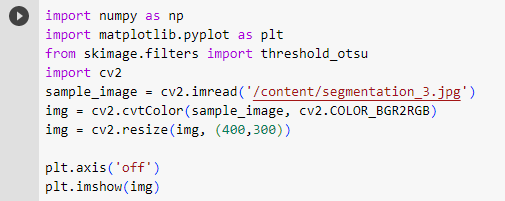
Cuối cùng, Chuyển đổi hình ảnh trở lại RGB để xem nó được phân đoạn (đồng thời có thể so sánh với hình ảnh gốc).



3. **Phân vùng hình ảnh bằng Ngưỡng Otsu**

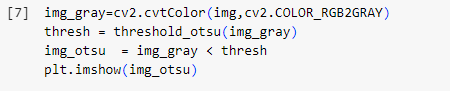
i) Nhập thư viện và hình ảnh

Nhập các thư viện cần thiết và tải hình ảnh mẫu.



ii) Áp dụng Ngưỡng Otsu trên Hình ảnh

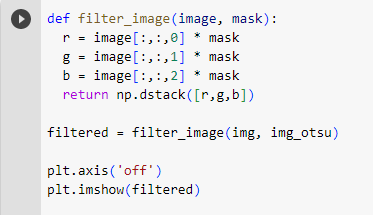
Ngưỡng Otsu là một kỹ thuật trong đó giá trị ngưỡng được xác định tự động để chuyển đổi hình ảnh thành hình ảnh nhị phân. Trước tiên, chúng ta chuyển đổi hình ảnh thành thang độ xám và sau đó sử dụng hàm Ngưỡng\_otsu () của thư viện lướt để tìm giá trị ngưỡng. Sử dụng điều này, chúng tôi tạo ra hình ảnh nhị phân.



iii) Quá trình phân đoạn

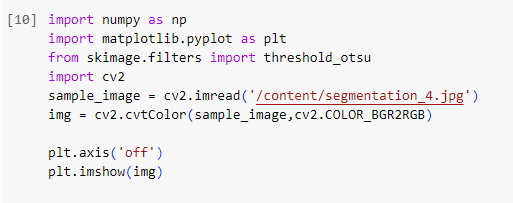
Ở đây, trước tiên chúng ta tạo một hàm “filter\_image” để nhân mặt nạ (được tạo trong phần trước) với các kênh RGB của hình ảnh của chúng ta. Hơn nữa, chúng được ghép nối để tạo thành một hình ảnh bình thường.

Cuối cùng, áp dụng chức năng “filter\_image” trên ảnh gốc (img) và mặt nạ được tạo bằng cách sử dụng ngưỡng (img\_otsu)



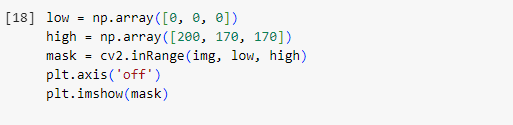
4. **Phân đoạn hình ảnh bằng cách sử dụng Mặt nạ màu**

i) Nhập thư viện và hình ảnh



ii) Tạo mặt nạ bằng cách phát hiện màu

Chúng ta sử dụng chức năng OpenCV inRange() và cung cấp dải màu RGB thấp và cao sẽ được phát hiện trong hình ảnh để tạo mặt nạ. Việc thay đổi dải màu thấp và cao sẽ ảnh hưởng đến kết quả phân đoạn. Do đó, để đưa ra phạm vi RGB, nó đòi hỏi sự hiểu biết của chúng ta về hình ảnh. Nên phương pháp này có thể không hữu ích trong các hình ảnh nhiều màu phức tạp.



iii) Áp mặt nạ

Cuối cùng, chúng ta sử dụng thao tác AND theo bit để áp dụng mặt nạ của vừa tạo ra lên ảnh và thu được phân đoạn hình ảnh.

