Bài giảng môn học:

Thị giác máy tính (7080518)

CHƯƠNG 4:
PHÂN VÙNG ẢNH
(Image Segmentation)

Đặng Văn Nam

dangvannam@humg.edu.vn

Nội dung chương 4

1. Bài toán phân vùng ảnh (image segmentation)
2. Ứng dụng của phân vùng ảnh
3. Một số kỹ thuật phân vùng ảnh
    1. Phân vùng ảnh dựa trên ngưỡng
    2. Phân vùng ảnh dựa trên cạnh
    3. Phân vùng ảnh dựa trên kỹ thuật phân cụm
4. Tìm kiếm vùng trong ảnh

1. Bài toán phân vùng ảnh

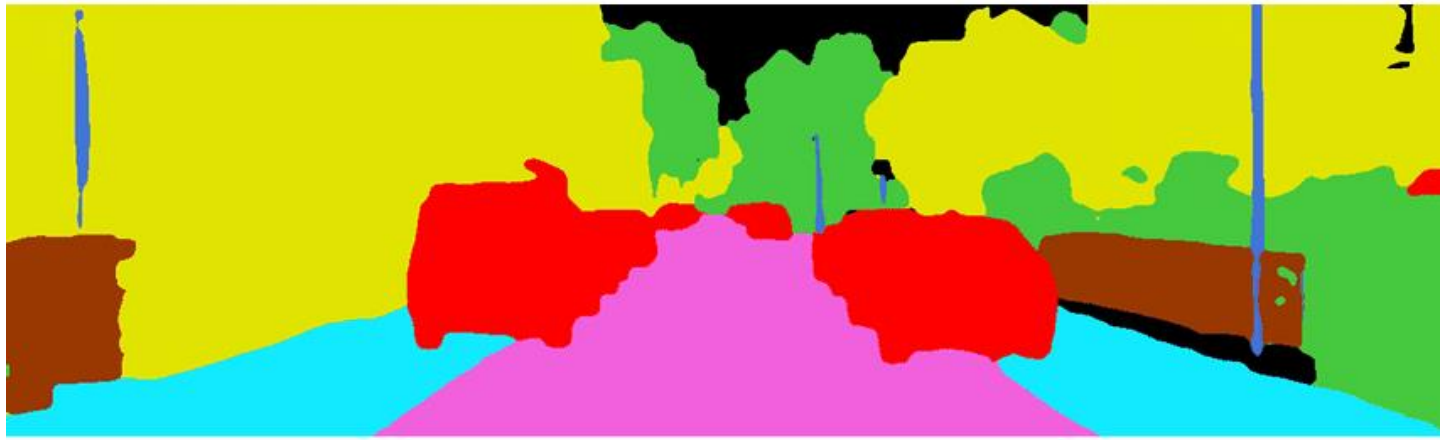
Giới thiệu

- Phân vùng ảnh (image segmentation) là một kỹ thuật quan trọng trong thị giác máy tính. Đây là tiền đề của của quá trình xử lý dữ liệu hình ảnh.
- Phân vùng ảnh là được ứng dụng trong nhiều lĩnh vực khác nhau như lĩnh vực hình ảnh y tế (medical imaging), phát hiện và nhận dạng đối tượng, hệ thống camera giám sát, hệ thống điều khiển giao thông…
- Kết quả phân vùng tốt sẽ tạo điều kiện thuận lợi cho các khâu xử lý về sau, đảm bảo tính hiệu quả cao, gia tăng mức độ chính xác, đồng thời giảm thiểu nguồn lực tính toán.

Phân vùng ảnh là gì?

- **Phân vùng ảnh** là một phương pháp mà trong đó, hình ảnh kỹ thuật số được chia thành nhiều nhóm con khác nhau được gọi là segments.

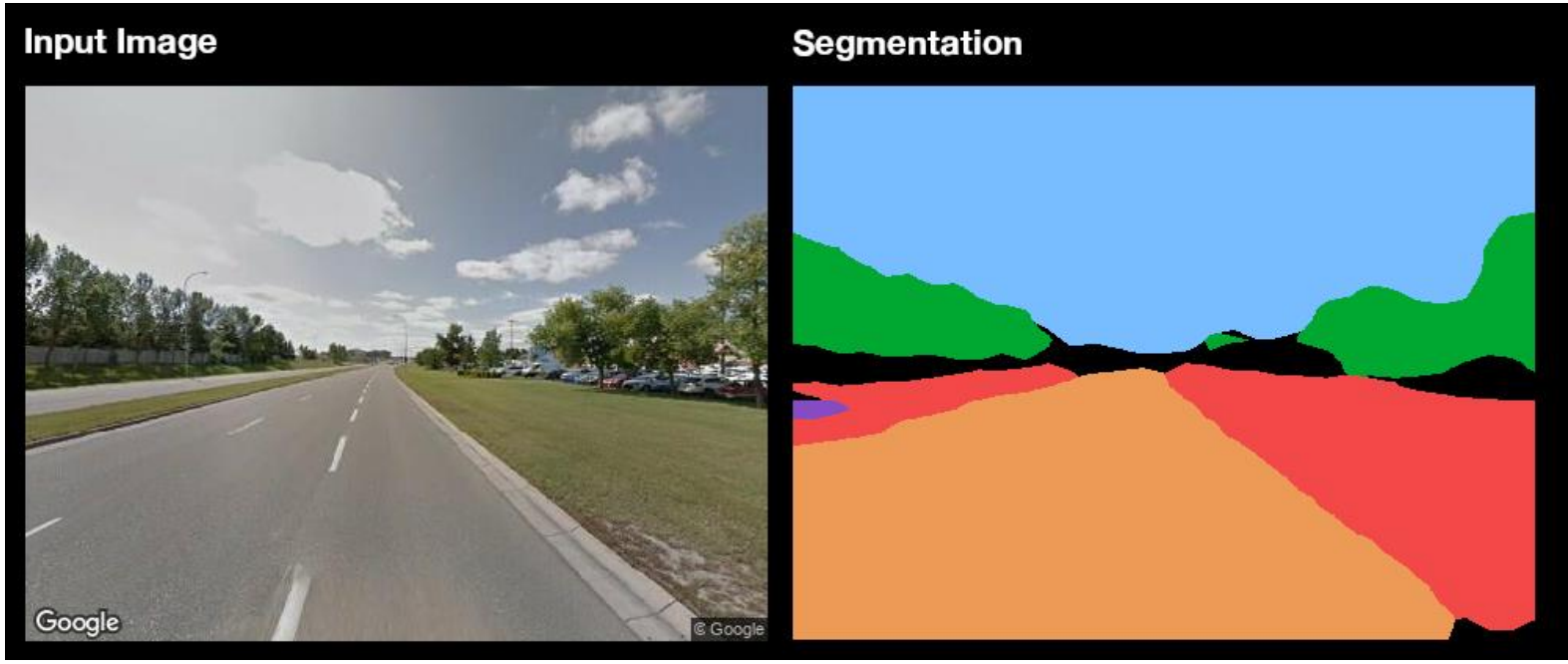
- Một cách dễ hiểu, phân vùng ảnh là một quá trình gán nhãn (assigning a lablel) cho mỗi điểm ảnh trong một bức ảnh, các điểm ảnh trong cùng một nhãn sẽ có những đặc tính giống nhau về màu sắc, cường độ hoặc kết cấu của ảnh



Phân vùng ảnh là gì?

- **Mục tiêu của phân vùng ảnh:** Làm giảm độ phức tạp của hình ảnh, giúp cho quá trình xử lý hoặc phân tích hình ảnh sau đó trở nên đơn giản hơn.

- **Kết quả của việc phân vùng ảnh là tập hợp các phân đoạn** (segments) bao gồm có thể là toàn bộ bức ảnh hoặc tập hợp các đường biên chiết xuất từ hình ảnh. Các điểm ảnh trong cùng một vùng có đặc tính tương tự nhau về màu sắc, cường độ hoặc kết cấu. Các vùng lân cận thì khác nhau đáng kể về các đặc trưng trên.



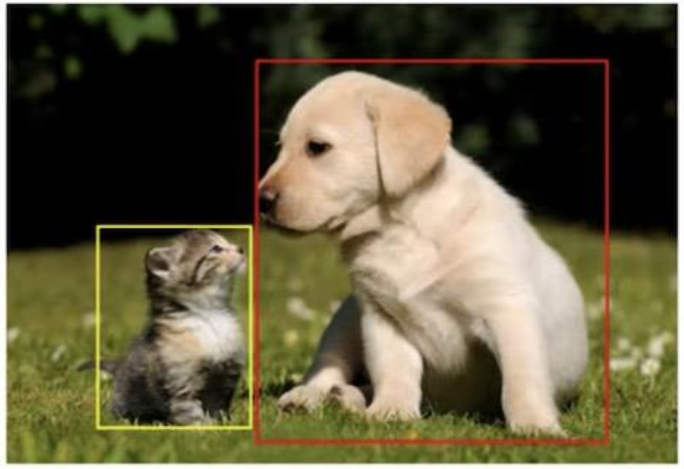
Phân vùng ảnh là gì?

- **Phân biệt một số bài toán trong thị giác máy tính**

Is this a dog?

Image Classification

What is there in image and where?



Object Detection

Which pixels belong to which object?

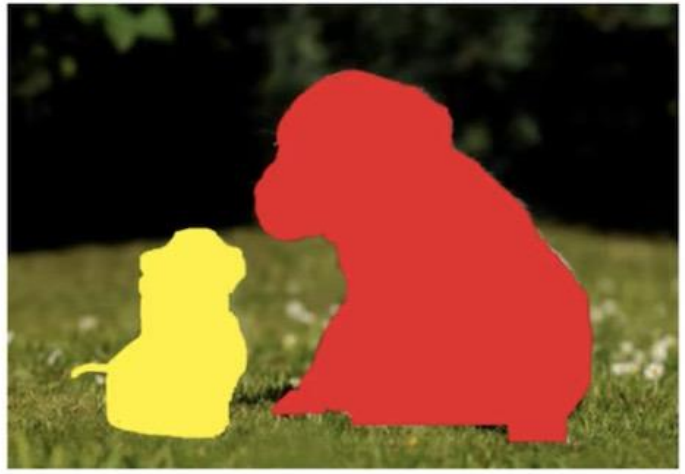


Image Segmentation

Các dạng phân vùng ảnh

- **Có 2 dạng bài toán phân vùng ảnh:**

Semantic Segmentation

Instance Segmentation

• Chúng ta phân đoạn (segment) các vùng ảnh theo những nhãn khác nhau mà không phân biệt sự khác nhau giữa các đối tượng trong từng nhãn. Ví dụ trong hình ảnh bên trái chúng ta phân biệt được pixel nào thuộc về người và pixel nào thuộc về background. Tuy nhiên trong bức ảnh xuất hiện 5 người, mức độ phân chia sẽ không xác định từng pixel thuộc về người nào.

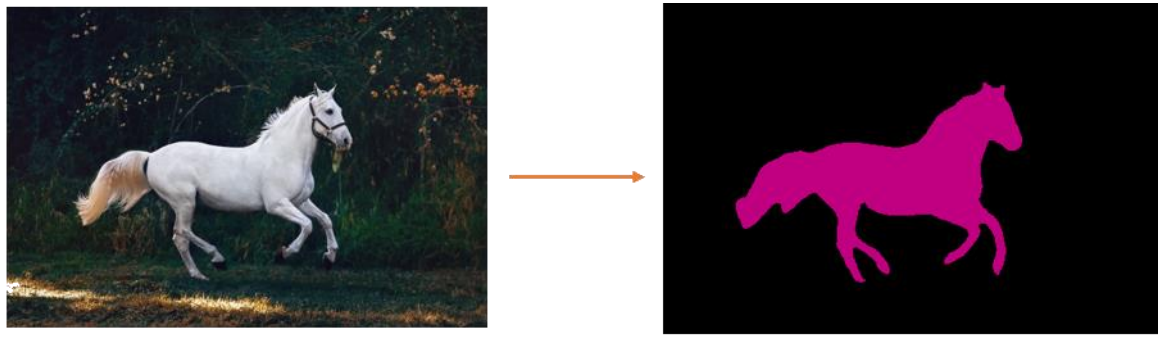
• Chúng ta phân đoạn các vùng ảnh chi tiết đến từng đối tượng trong mỗi nhãn. Ví dụ: ở hình ảnh bên phải đối với nhãn người sẽ được phân chia chi tiết tới từng người 1, 2, … , 5.

2. Ứng dụng của phân vùng ảnh

Ứng dụng của phân vùng ảnh

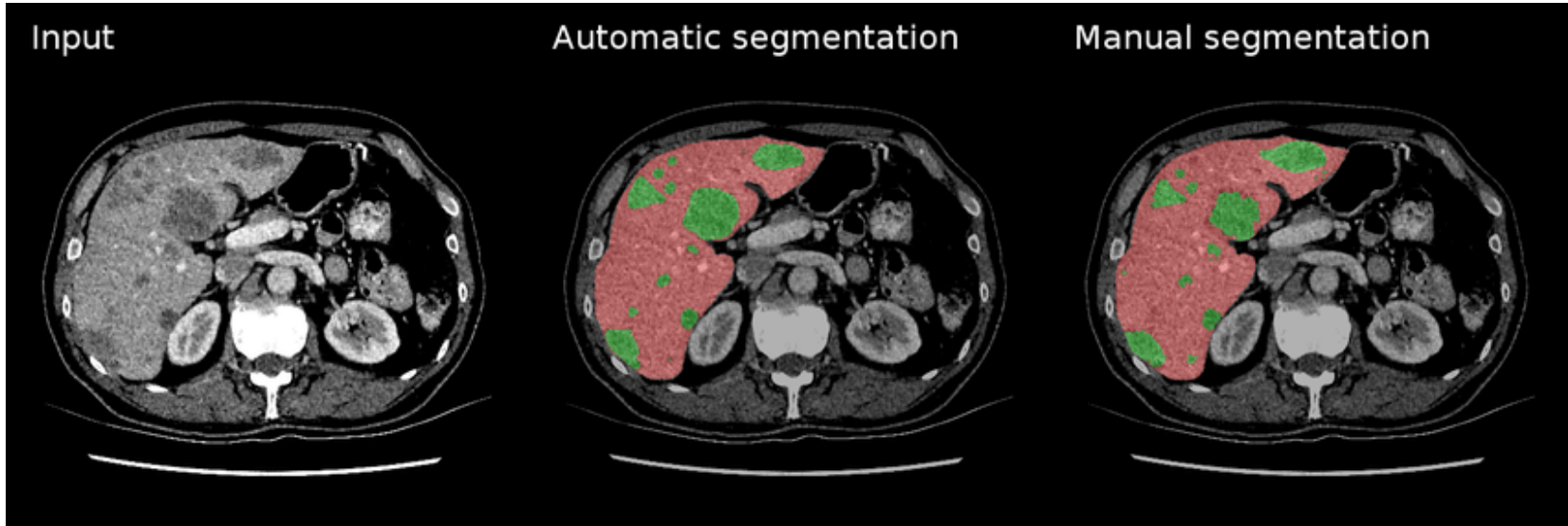
- **Ứng dụng trong lĩnh vực nhận dạng đối tượng,** quá trình phân vùng ảnh sẽ tách đối tượng ra khỏi vùng nền. Đối tượng ở đây có thể là con người hoặc một vật gì đó chuyển động thuộc vùng tiền cảnh (Fg).
- Đối tượng sau khi được tách ra trong quá trình phân vùng có thể được xử lý trong các hệ thống như đếm số lượng người ra vào, nhận dạng cử chỉ tay, nhận dạng khuôn mặt.

- **Nhận dạng đối tượng:**
  - Phát hiện đi bộ
  - Phát hiện khuôn mặt
  - Phát hiện đèn dừng xe
  - Xác định vị trí đối tượng trong ảnh vệ tinh



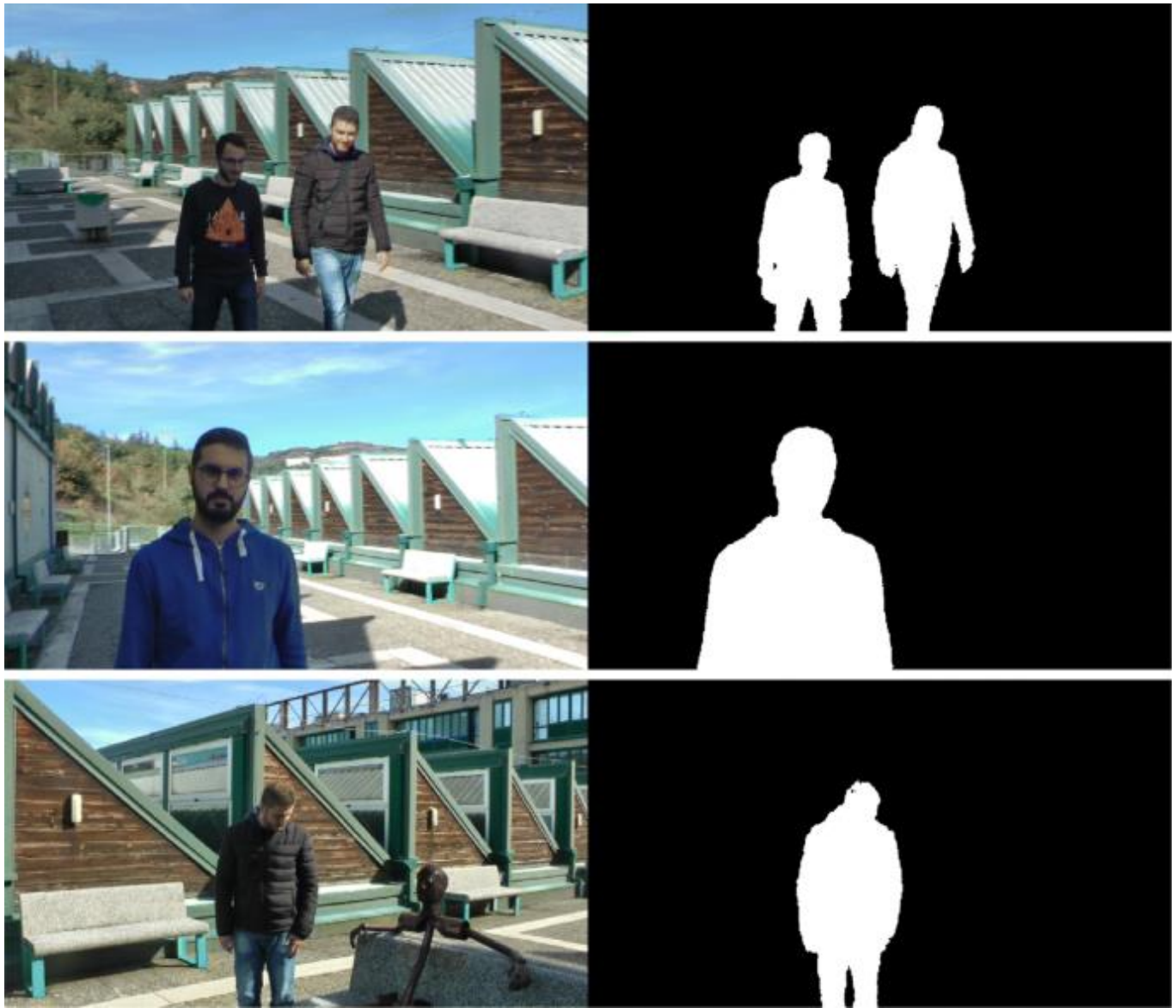
Ứng dụng của phân vùng ảnh

- **Ứng dụng trong lĩnh vực hình ảnh y tế**, các kỹ thuật hình ảnh y tế như chụp CT (Computer Tomography), chụp MRI (Magnetic Resonance Imaging), chụp X-Quang, USG (Ultrasound) không thể thiếu để có thể phân tích chính xác nhiều bệnh lý khác nhau, qua đó đã hỗ trợ đáng kể bác sĩ trong việc chẩn đoán bệnh.
- Trong quá trình phân tích, người chẩn đoán cần trích xuất các đường biên cần thiết, các bề mặt hoặc các bộ phận cơ thể ra khỏi bức hình, kỹ thuật này được gọi là phân vùng (segmentation). Tuy nhiên, quá trình phân vùng thủ công là rất tốn thời gian và có thể không cho kết quả tốt. Các phân vùng và đường biên này là rất quan trọng đối với các bác sĩ.
- Chính vì vậy, trong vài thập kỷ qua, nhiều thuật toán phân vùng ảnh y tế được đề xuất nhằm tăng độ chính xác trong quá trình phân vùng ảnh.



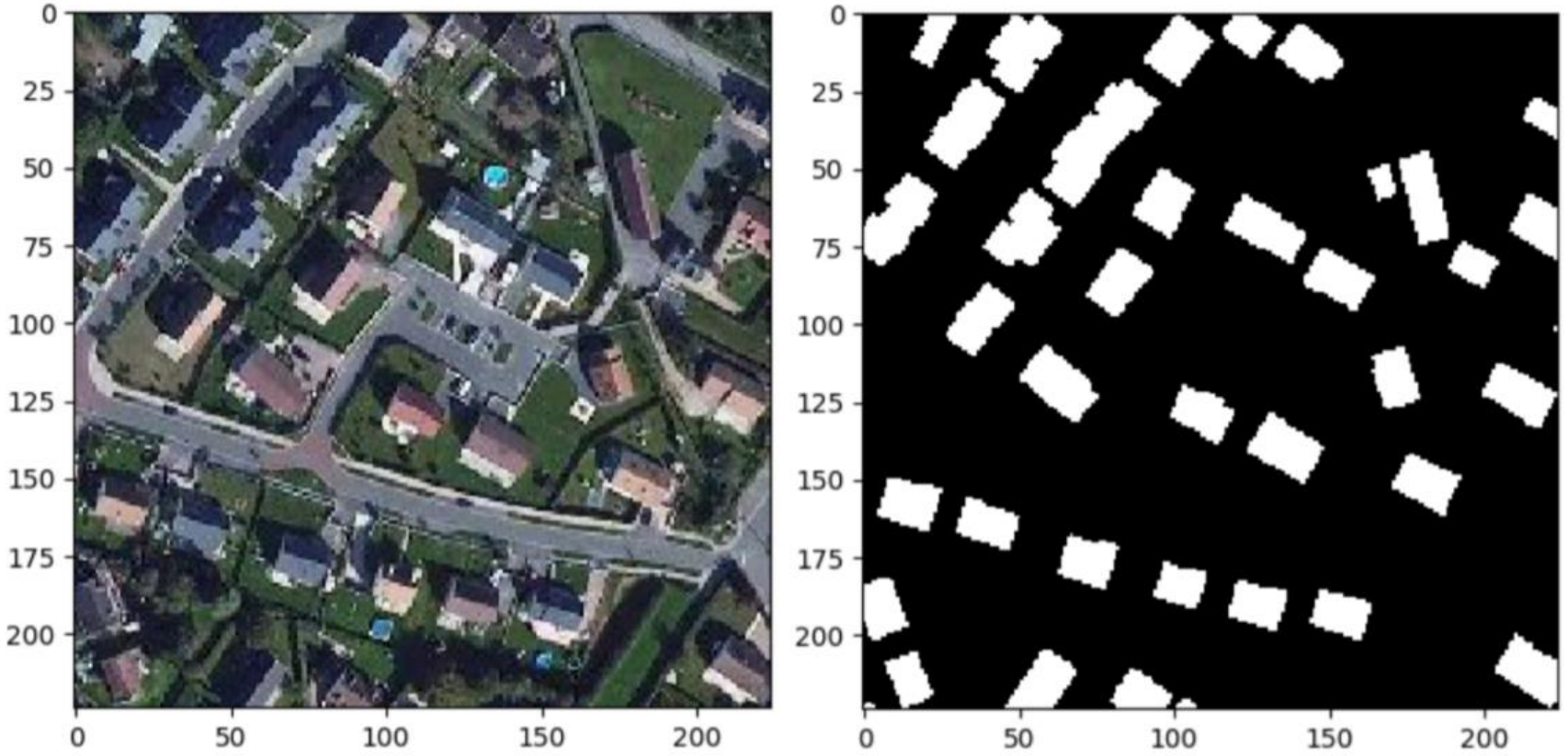
Ứng dụng của phân vùng ảnh

- **Ứng dụng trong lĩnh vực camera giám sát,** quá trình phân vùng ảnh có thể ứng dụng trong việc xác định, giám sát đối tượng đi vào vùng giám sát, cảnh báo chuyển động khi đối tượng di chuyển vào vùng giám sát.



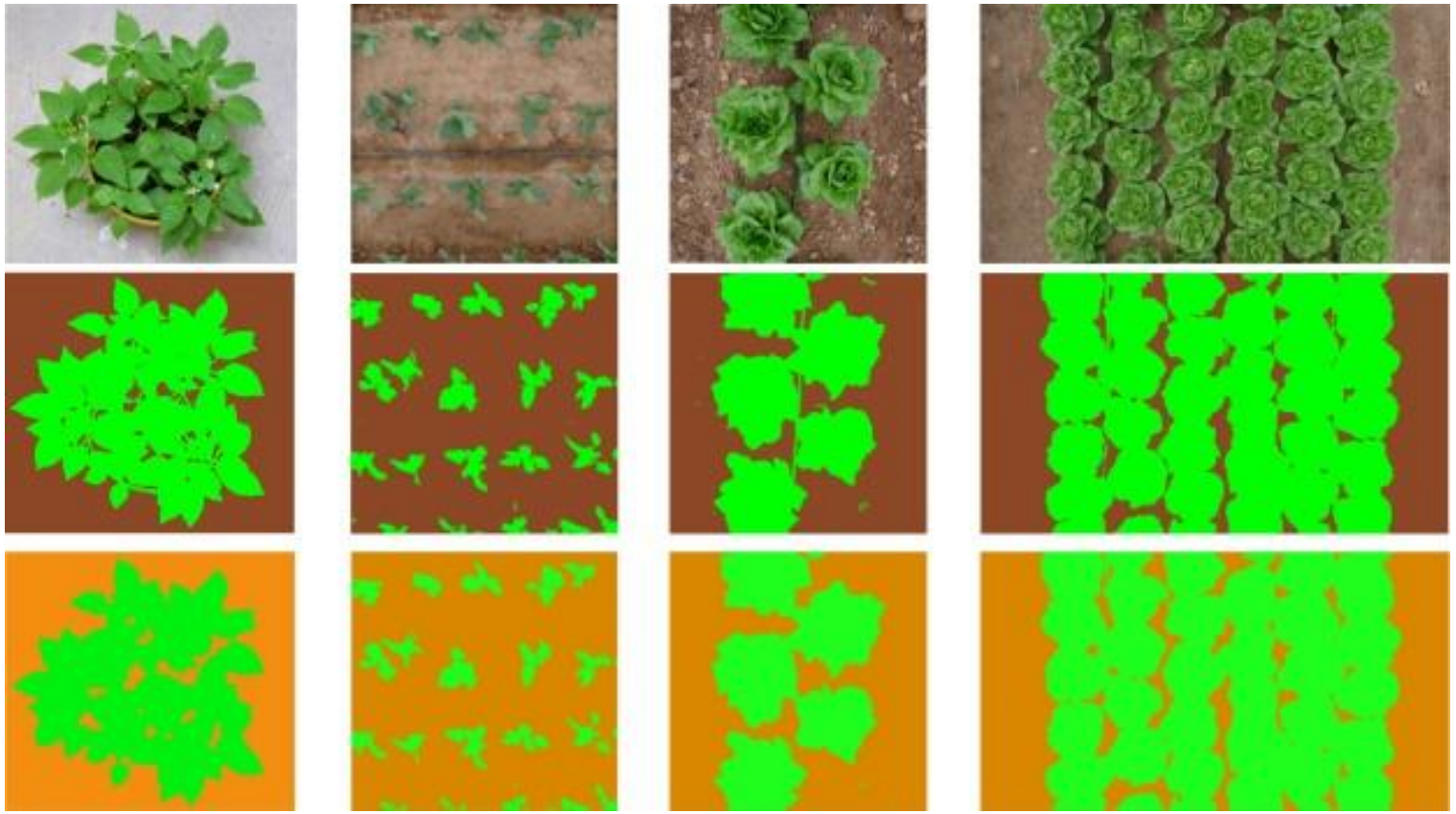
Ứng dụng của phân vùng ảnh

- **Ứng dụng trong xử lý ảnh vệ tinh,** Các vệ tinh quay quanh trái đất sẽ liên tục thu thập hình ảnh bề mặt trái đất ở những vùng khác nhau. Từ các bức ảnh vệ tinh, mô hình phân vùng ảnh sẽ phân hình ảnh thành các thuyến đường, khu phố, biển cả, cây cối.



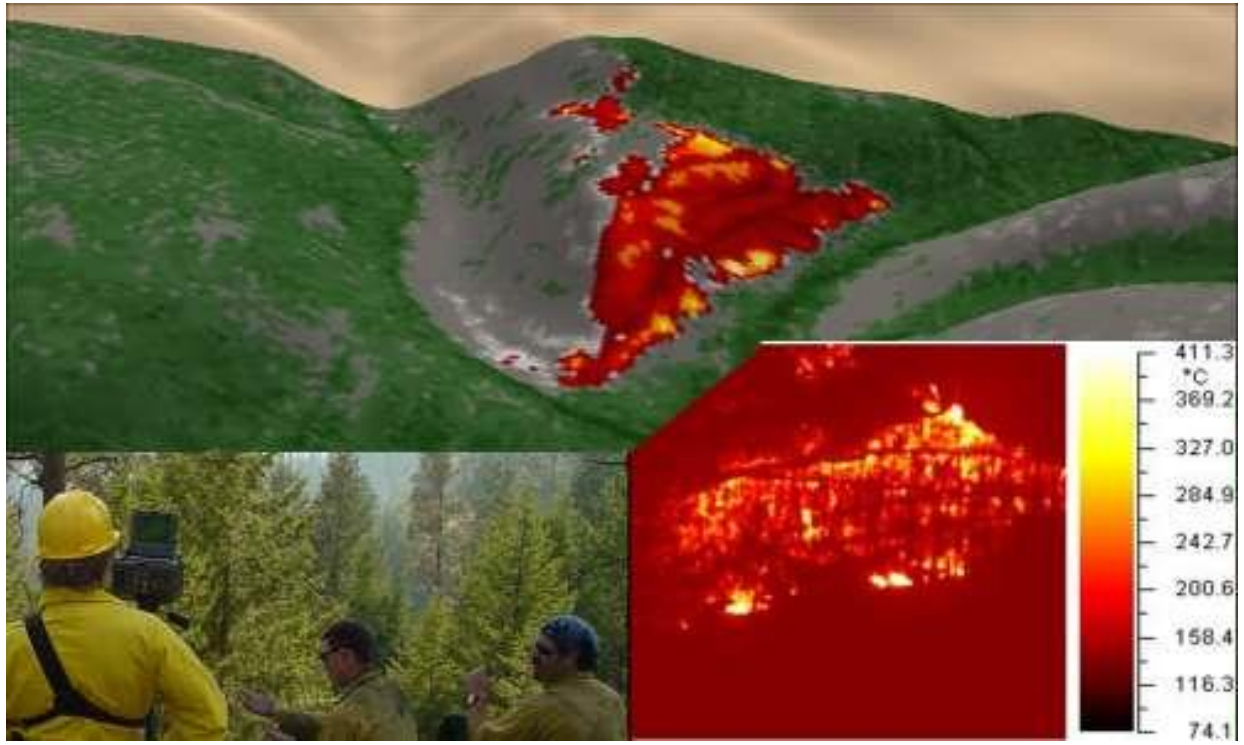
Ứng dụng của phân vùng ảnh

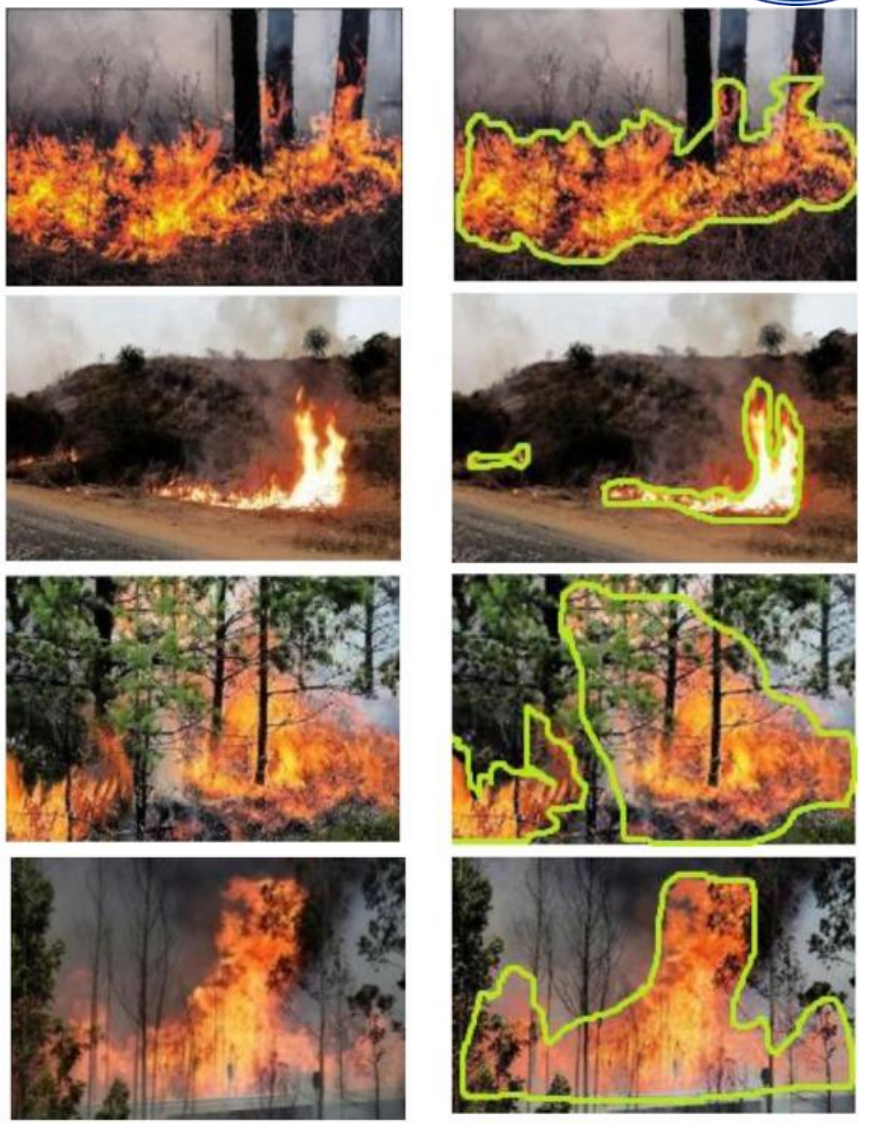
- **Ứng dụng trong nông nghiệp,** Chúng ta có thể tiết kiệm được một lượng lớn thuốc trừ sâu, nước tưới trong nông nghiệp nhờ sử dụng hệ thống phun thuốc trừ sâu, tưới nước tự động có khả năng phân biệt được diện tích cỏ và cây trồng, đất trống dựa trên thuật toán Image Segmentation.



Ứng dụng của phân vùng ảnh

- **Ứng dụng trong cảnh báo cháy rừng**, những hệ thống cảnh báo cháy rừng có thể phân vùng được chính xác vị trí phát sinh đám cháy từ ảnh chụp vệ tinh. Từ đó đưa ra cảnh báo về qui mô và mức độ lây lan của các đám cháy trên diện rộng.





Các cách tiếp cận phân vùng ảnh

- **Cách tiếp cận tương đồng (Similarity approach),** có nghĩa là phát hiện sự tương đồng giữa các pixel hình ảnh để tạo thành một phân đoạn, dựa trên một ngưỡng. Các thuật toán học máy như phân cụm thường dựa trên kiểu tiếp cận này để phân vùng một hình ảnh.
- **Cách tiếp cận gián đoạn (Discontinuity approach):** Cách tiếp cận này dựa trên sự gián đoạn của các giá trị cường độ pixel trong hình ảnh. Các kỹ thuật phát hiện đường, điểm và cạnh sử dụng kiểu tiếp cận gián đoạn để thu được các kết quả phân vùng trung gian. Kết quả này sau đó có thể được xử lý để cho ra hình ảnh được phân vùng cuối cùng.

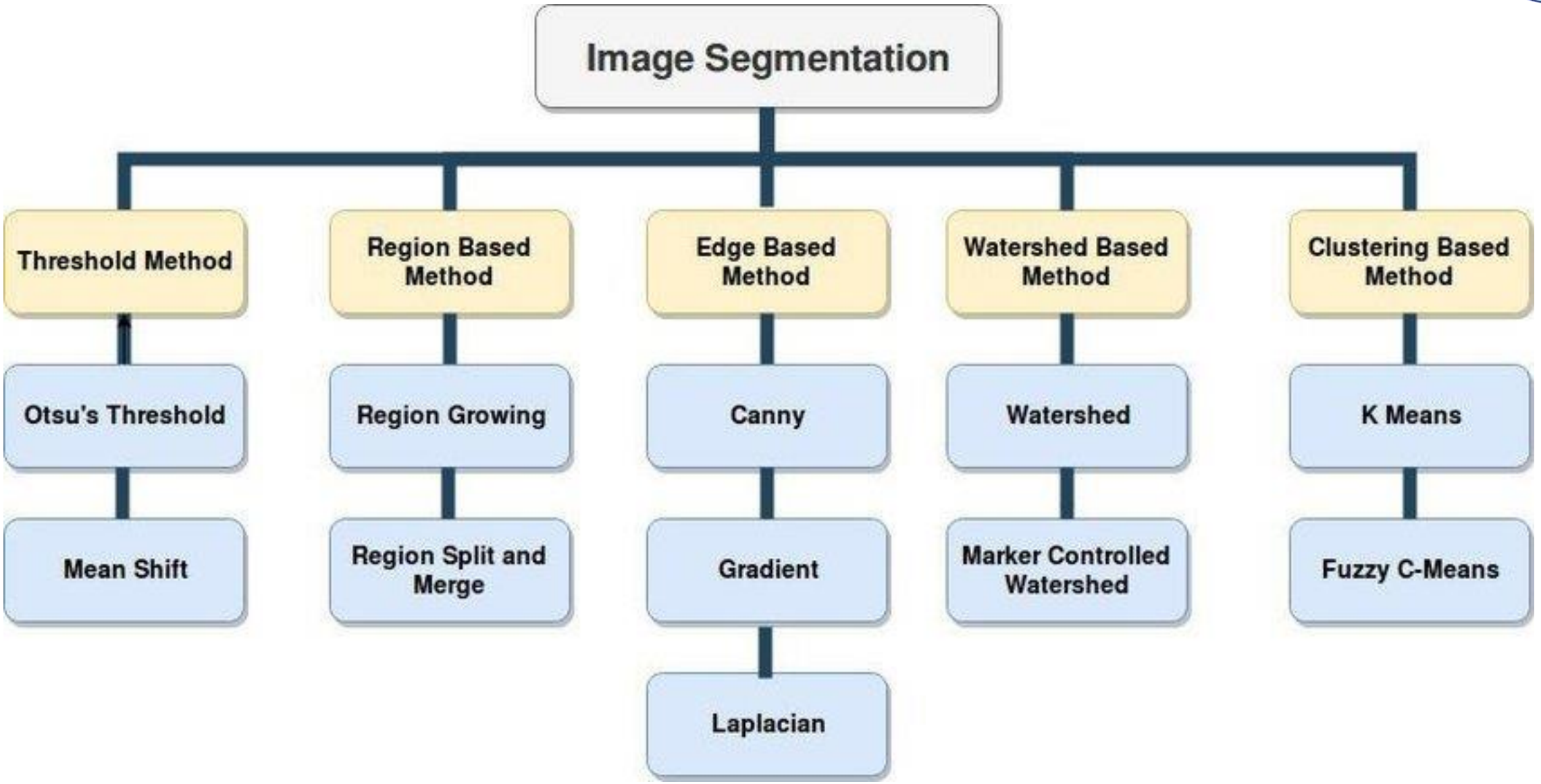
3. Phương pháp phân vùng ảnh

Phương pháp phân vùng ảnh

Có nhiều phương pháp phân vùng ảnh, một số phương pháp phổ biến bao gồm:

1. Phân vùng dựa trên ngưỡng (Threshold Based Method)
2. Phân vùng dựa trên khu vực (Region Based Method)
3. Phân vùng dựa trên cạnh (Edge Based Method)
4. phân vùng dựa trên lưu vực (Watershed Based Method)
5. Phân vùng dựa trên kỹ thuật phân cụm (Clustering Based Method)
6. Phân vùng dựa trên mạng neural nhân tạo (Artificial Neural Nework Based Method)

Phương pháp phân vùng ảnh

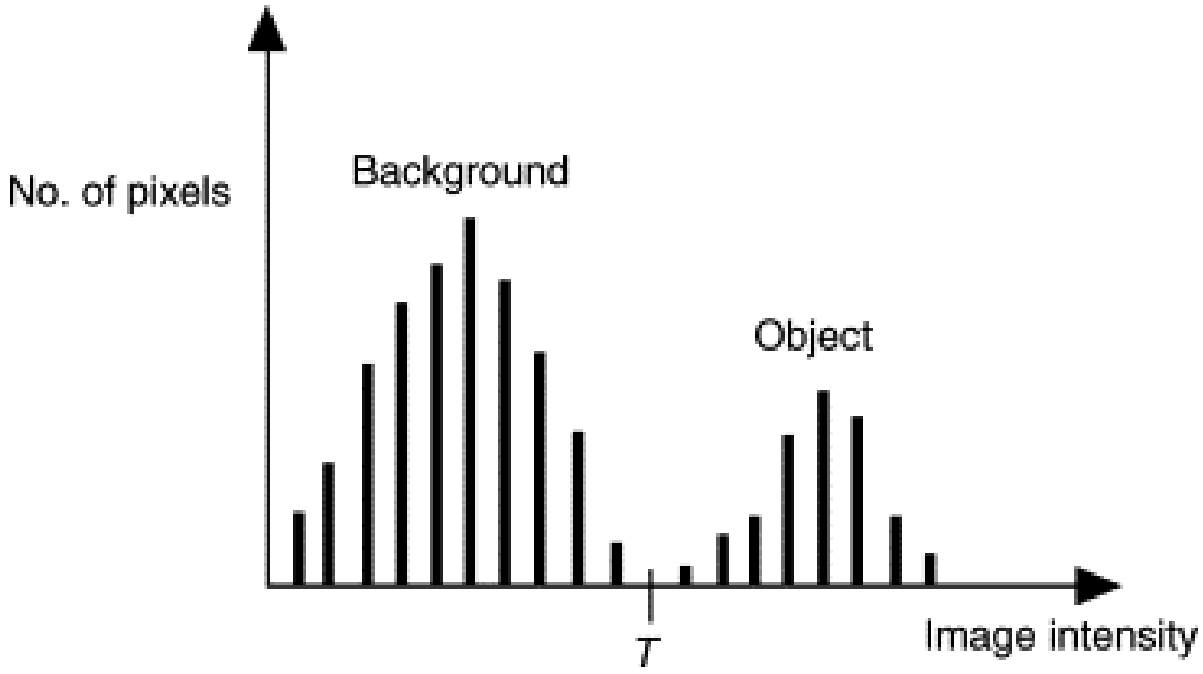


3.1. Phân vùng ảnh dựa trên ngưỡng (Threshold Based Method)

Phân vùng ảnh dựa trên ngưỡng

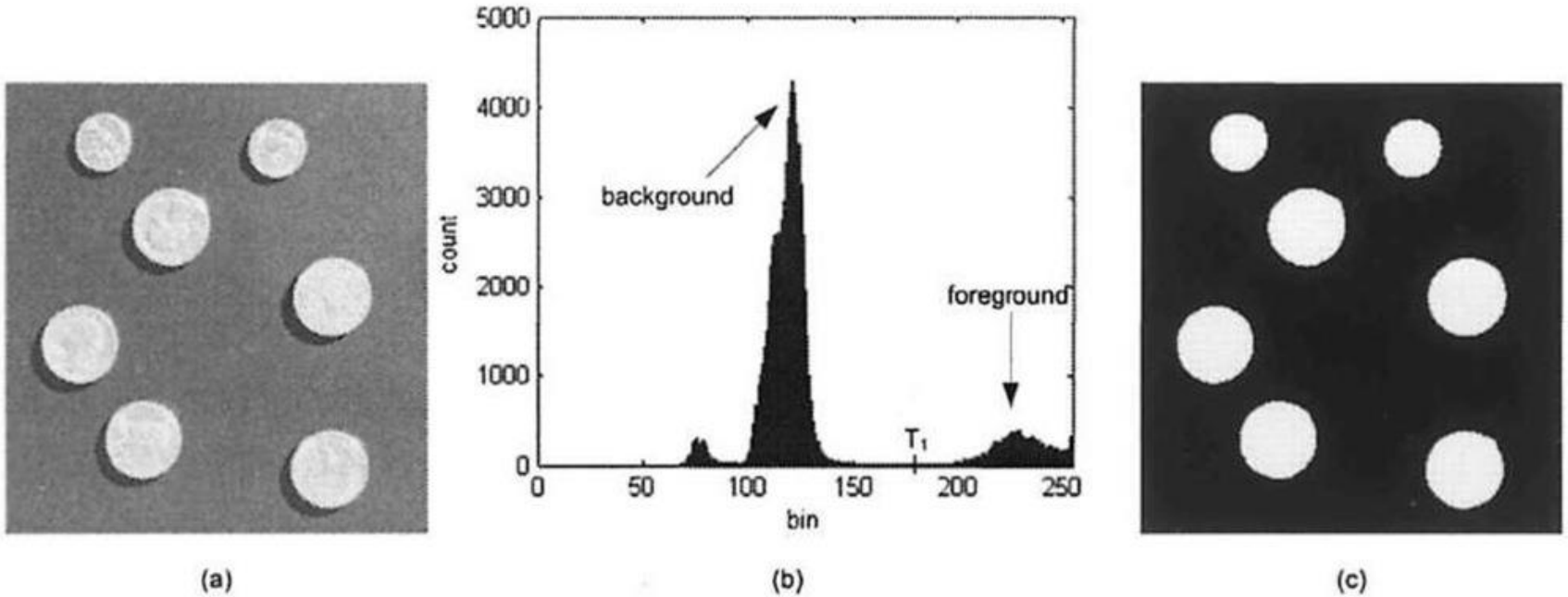
- Đây là phương pháp phân vùng đơn giản, giúp tạo ra một hình ảnh nhị phân hoặc nhiều màu dựa trên việc đặt giá trị ngưỡng theo cường độ pixel của ảnh gốc.

- Đối với một ảnh có nền và đối tượng, có thể chia ảnh thành các vùng dựa trên cường độ của đối tượng và nền. Nhưng ngưỡng này phải được thiết lập hoàn hảo để phân đoạn hình ảnh thành một đối tượng và một nền.



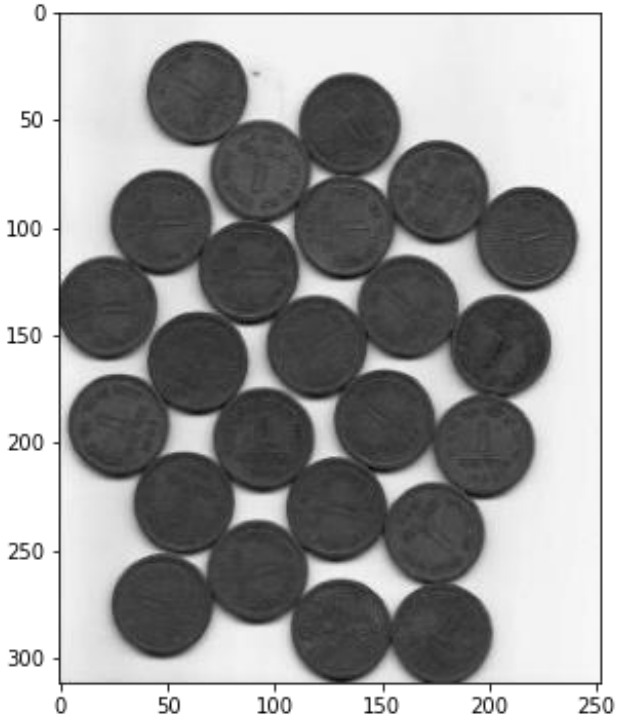
Phân vùng ảnh dựa trên ngưỡng

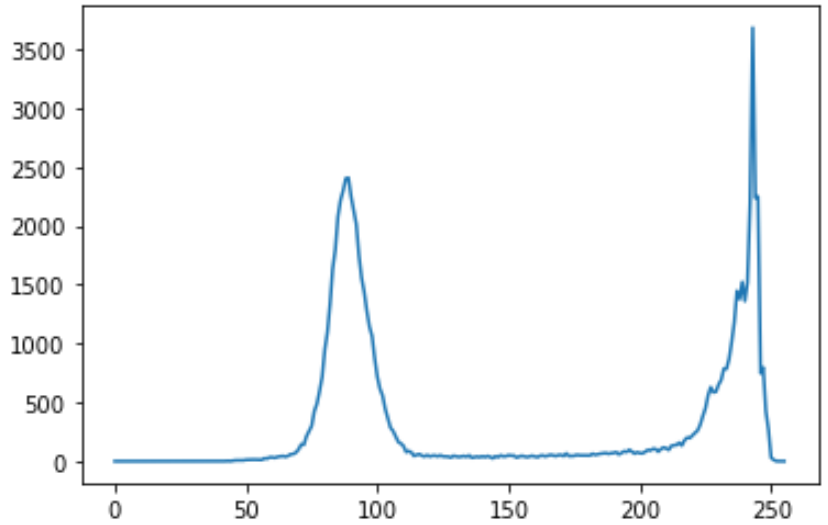
- Phân ngưỡng bao gồm các kỹ thuật như:
  - ngưỡng toàn cục (Global thresholding);
  - ngưỡng thủ công (Manual thresholding);
  - ngưỡng thích ứng (Adaptive Thresholding);
  - ngưỡng tối ưu (Optimal Thresholding);
  - ngưỡng thích ứng cục bộ (Local Adaptive Thresholding).

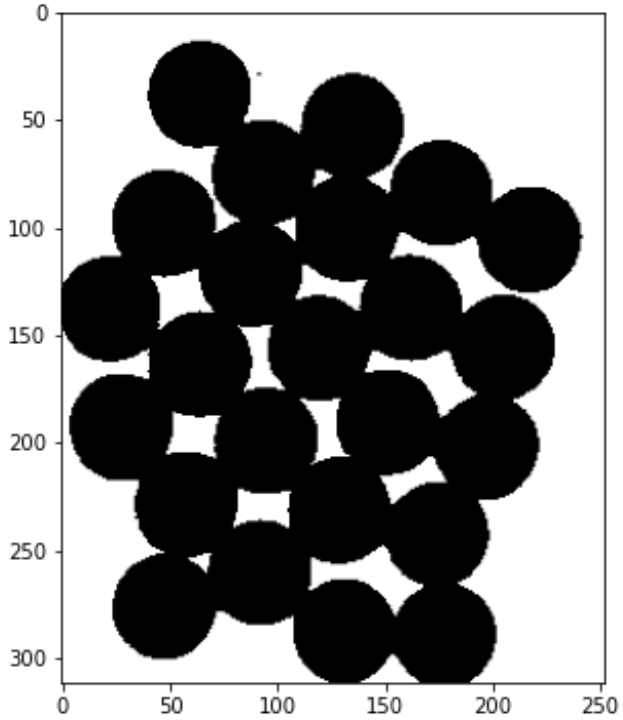


Ngưỡng toàn cục (Global thresholding)

- Sử dụng một ngưỡng T chung cho tất cả các pixel trong một ảnh.





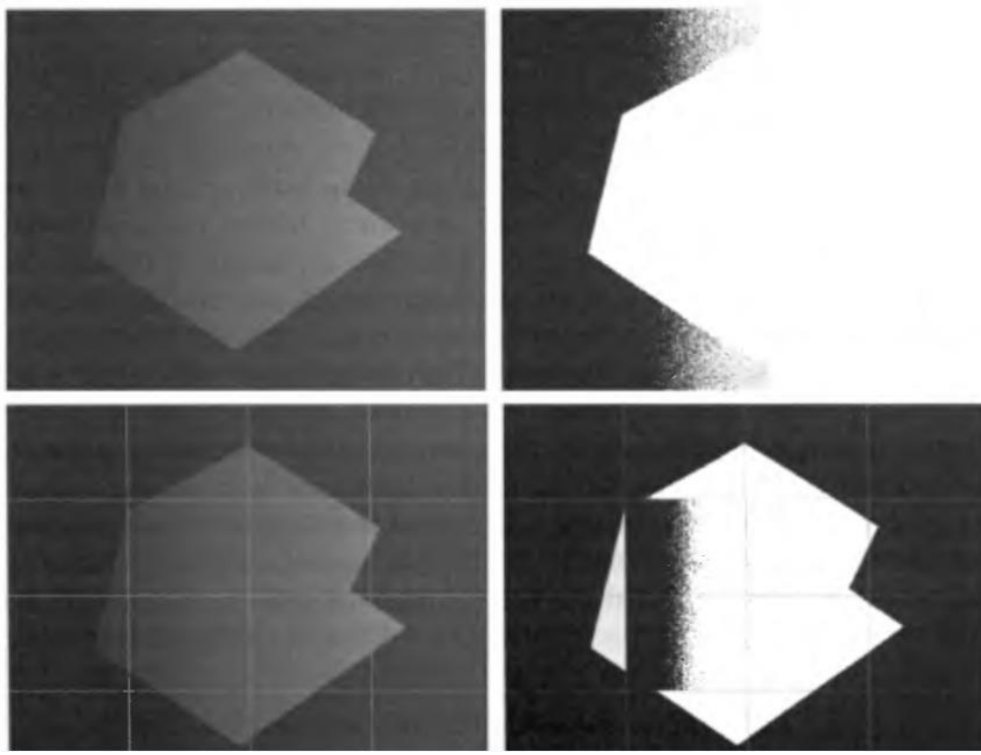



```
#Apply a threshold:
(T, thresh) = cv2.threshold(img_orignal, 170, 255, cv2.THRESH_BINARY)

plt.figure(figsize=(8,6))
plt.imshow(thresh,cmap='gray')
plt.show()
```

Ngưỡng thích ứng (Adaptive Thresholding)

- Phương pháp phân ngưỡng toàn cục không phù hợp cho nhiều trường hợp như là với trường hợp ánh sáng không đồng đều trên ảnh. Khi đó phân ngưỡng thích là một giải pháp.

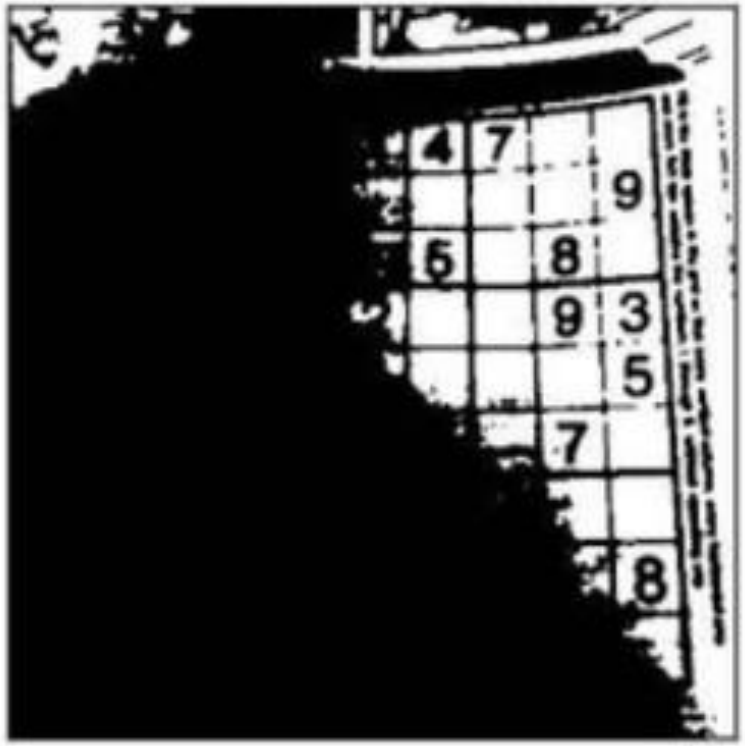
- OpenCV cung cấp phương thức adaptiveThreshold để thực hiện phân ngưỡng thích nghi. Phương thức này sẽ tính giá trị trung bình của n điểm ảnh xung quanh điểm ảnh đó rồi trừ đi C (n là thường là số lẻ, C là một số nguyên bất kỳ)



Ngưỡng thích ứng (Adaptive Thresholding)

Original Image

Global Thresholding



Adaptive Thresholding

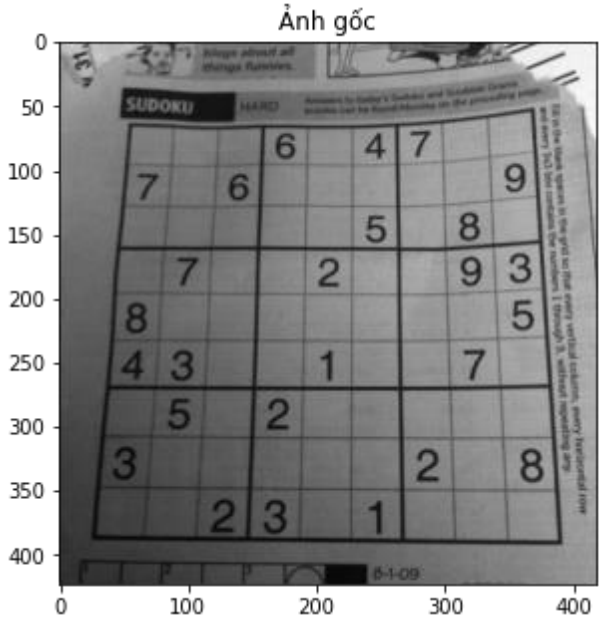
Ngưỡng thích ứng (Adaptive Thresholding)

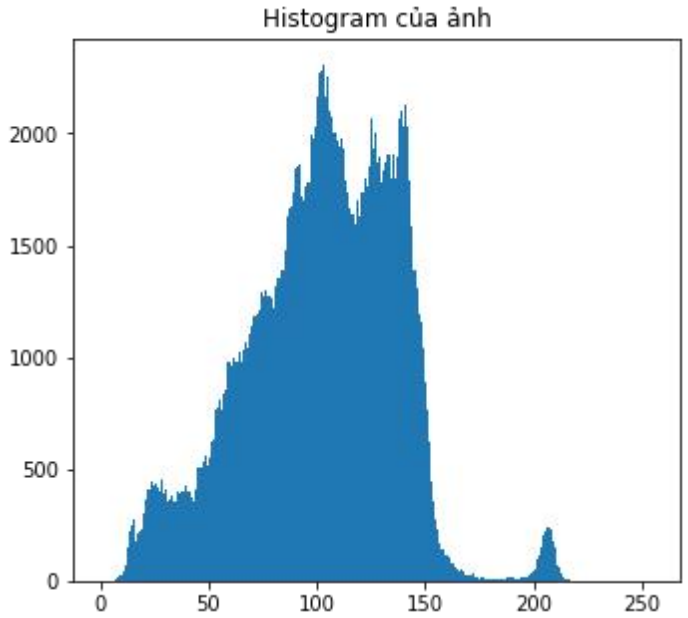
pic = cv2.adaptiveThreshold(img, maxValue, thresholdType, type, n, C)

Trong đó:

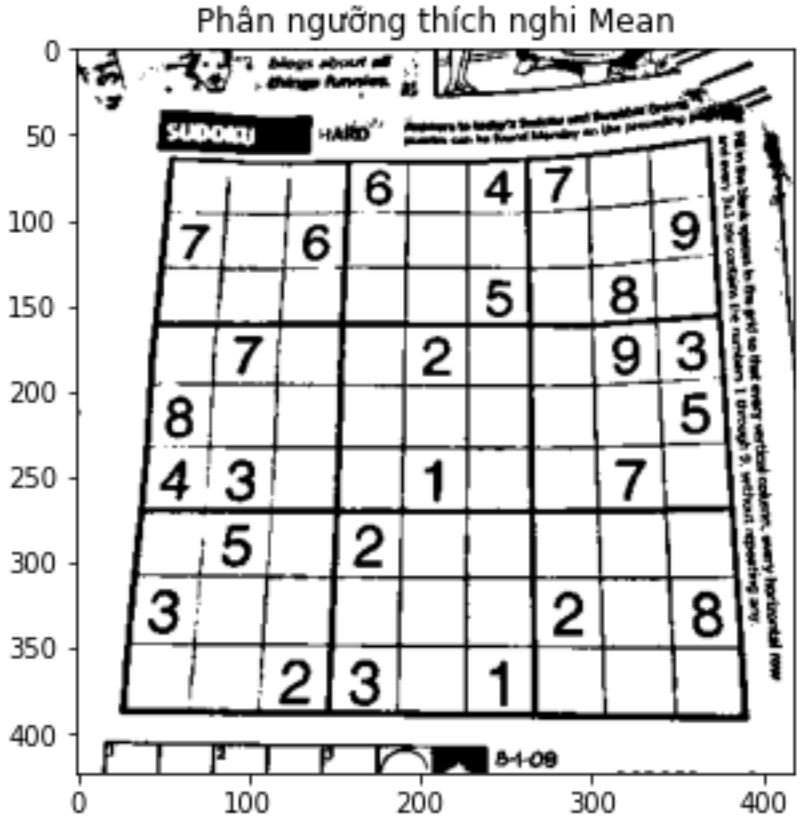
- img: ảnh gốc
- maxValue: giá thiết lập nếu > giá trị ngưỡng
- adaptiveMethod: phương pháp tính cho các điểm lân cận lấy trung bình hay theo Gaussian
  - ADAPTIVE_THRESH_MEAN_C − tính trung bình các điểm lân cận
  - ADAPTIVE_THRESH_GAUSSIAN_C − tính các điểm lân cận theo Gaussian
- type: Loại xử lý cắt ngưỡng
  - THRESH_BINARY: Cắt ngưỡng nhị phân.
  - THRESH_BINARY_INV: Cắt ggưỡng nhị phân đảo ngược.
- n: Số lượng điểm ảnh xung quanh sử dụng để tính giá trị cho điểm đang xét, là số lẻ.
- C: Giá trị số bất kỳ được sử dụng để trừ.

Ngưỡng thích ứng (Adaptive Thresholding)





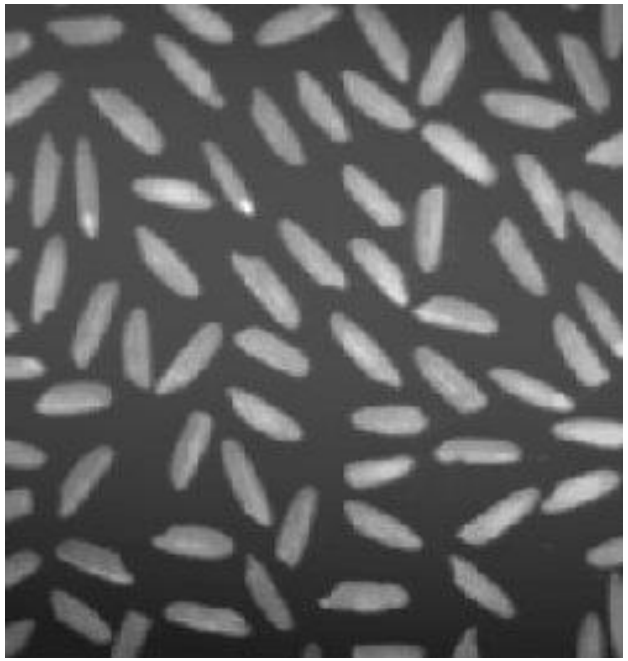

```python
#Thực hiện phân ngưỡng thích nghi
#Phương pháp tính trung bình:
th_mean = cv2.adaptiveThreshold(img_sudoku,255,
                                cv2.ADAPTIVE_THRESH_MEAN_C,
                                cv2.THRESH_BINARY,31,10)


#Phương pháp tính Gaussian:
th_gaussian = cv2.adaptiveThreshold(img_sudoku,255,
                                    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                    cv2.THRESH_BINARY,31,10)

```

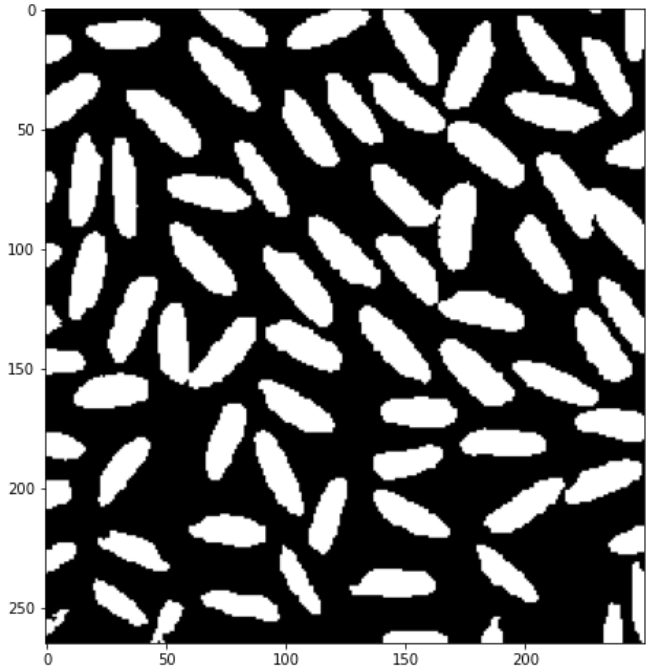


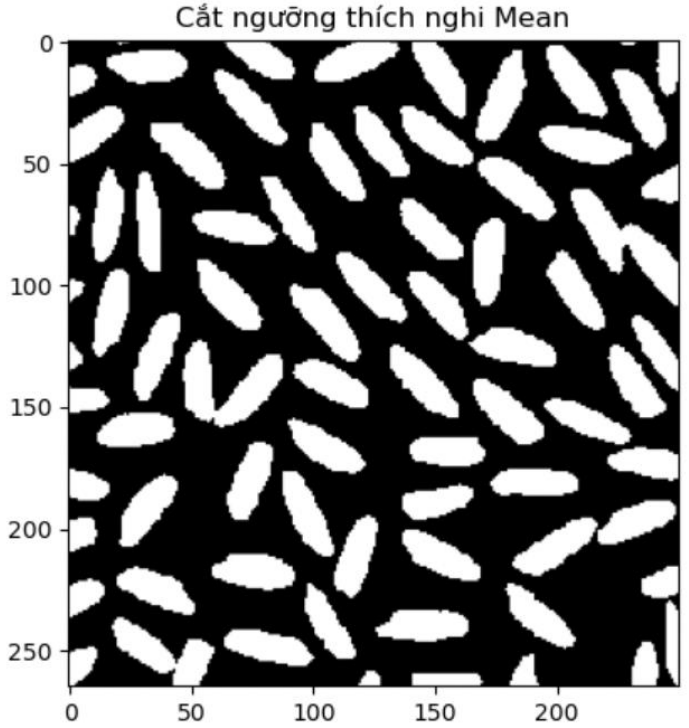
Thực hành 4.1

Thực hành

1. Đọc và hiển thị ảnh Thuchanh4_1.jpeg ở dạng ảnh xám
2. Phân vùng ảnh sử dụng phương pháp phân ngưỡng toàn cục (Hãy xác định ngưỡng phù hợp để đạt kết quả tốt nhất)
3. Phân vùng ảnh sử dụng phương pháp phân ngưỡng thích nghi (Hãy xác định các tham số phù hợp để đạt kết quả tốt nhất)







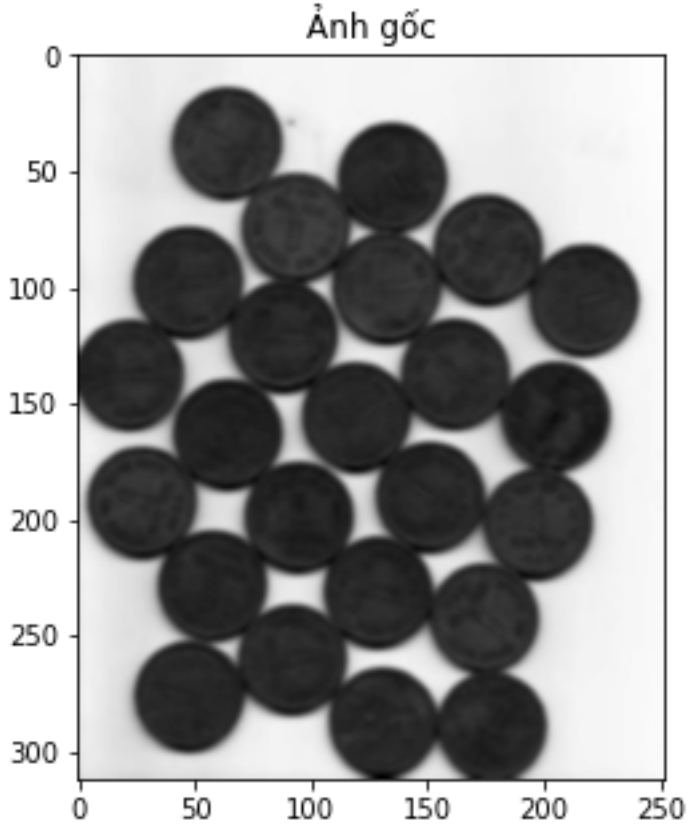
3.2. Phân vùng ảnh dựa trên cạnh, biên (Edge Based Method)

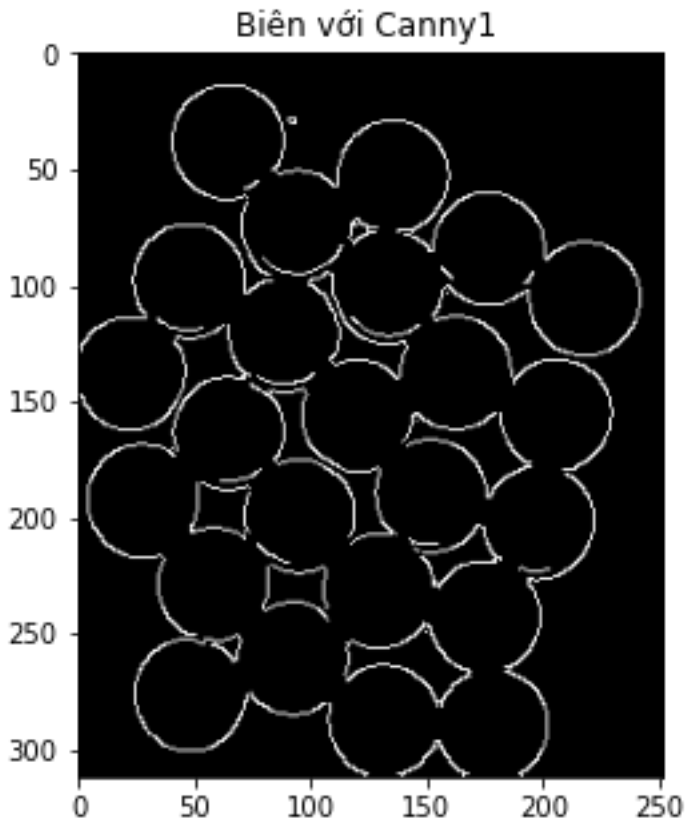
Phân vùng ảnh dựa trên cạnh, biên

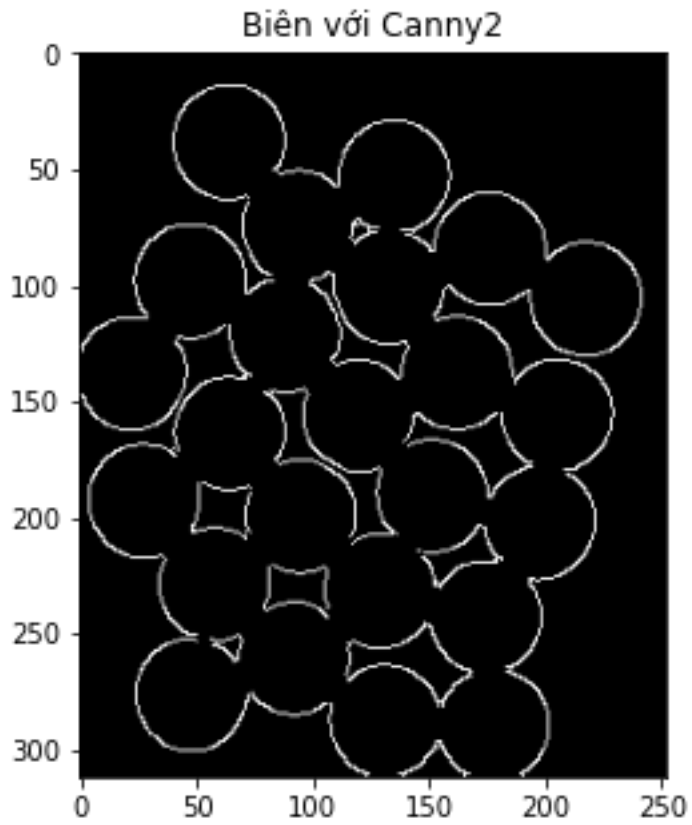
- Cạnh trong ảnh đánh dấu những vị trí hình ảnh không liên tục về mức xám, màu sắc, kết cấu, v.v. Khi di chuyển từ vùng này sang vùng khác, mức xám có thể thay đổi. Vì vậy, nếu tìm thấy sự gián đoạn đó, ta có thể tìm thấy cạnh. Thực tế, có nhiều toán tử phát hiện cạnh, nhưng hình ảnh thu được là kết quả phân vùng trung gian, và không nên nhầm lẫn với hình ảnh được phân vùng cuối cùng.
- Để ra được kết quả cuối, cần thực hiện một số bước bổ sung bao gồm: kết hợp các phân vùng cạnh thu được làm một, để giảm số lượng phân vùng và có được một đường viền liền mạch của đối tượng.
- Như vậy, có thể thấy, phân vùng cạnh đưa ra một kết quả phân vùng trung gian. Kết quả này sau đó có thể áp dụng theo vùng hoặc bất kỳ kiểu phân đoạn nào khác, nhằm có được hình ảnh được phân vùng cuối.

Phân vùng ảnh dựa trên cạnh, biên

- Sử dụng Canny để phân vùng:



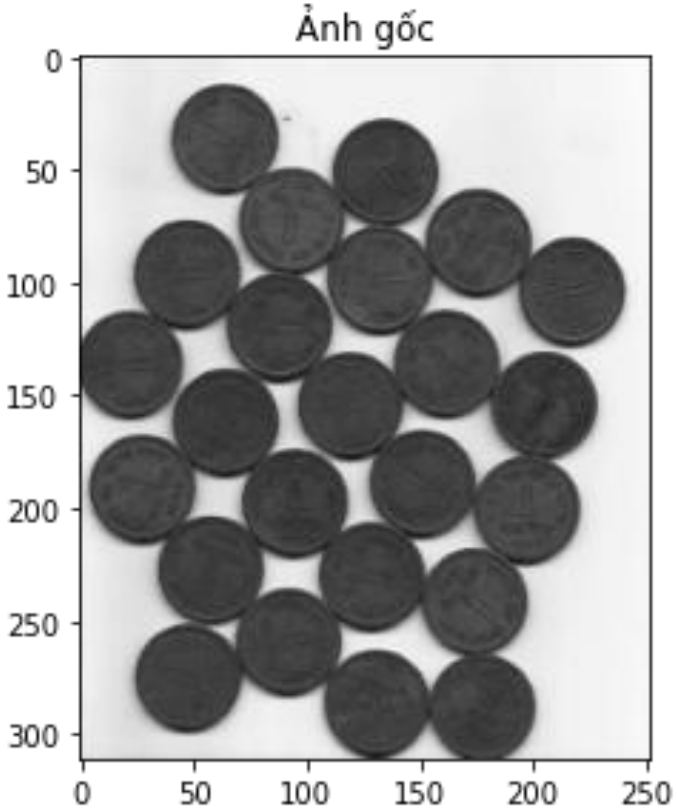




Phân vùng ảnh dựa trên cạnh, biên

- Sử dụng các phương thức OpenCV cung cấp thực hiện đếm số lượng đồng xu trong ảnh

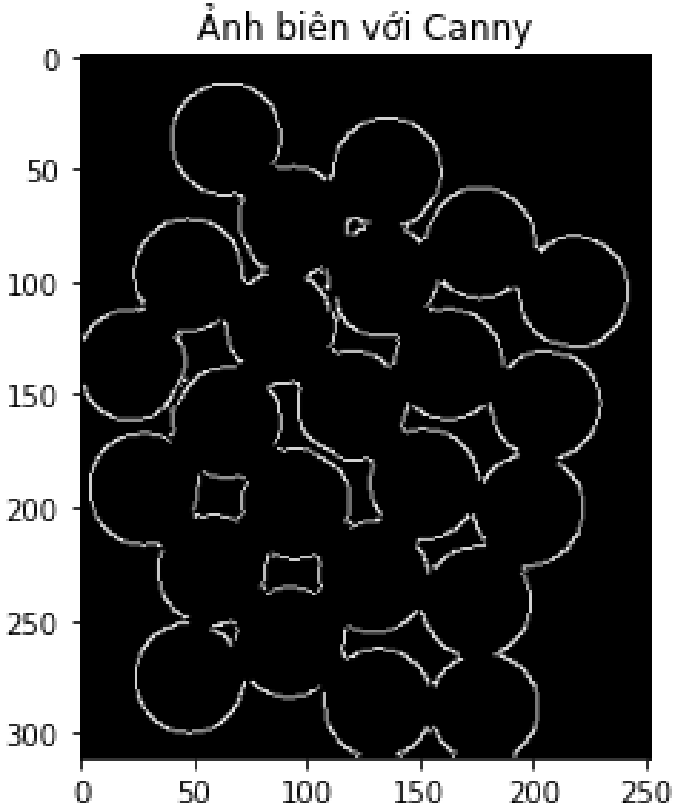
```
1  #Bước 1: Đọc ảnh ở chế độ ảnh màu:
2  image = cv2.imread('images/pic_money.jpeg')
3
4  #Hiển thị ảnh gốc ở dạng xám:
5  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
6
7  plt.figure(figsize=(10,10))
8  plt.subplot(2,2,1)
9  plt.imshow(gray, cmap='gray');
10 plt.title('Ảnh gốc')
```



Phân vùng ảnh dựa trên cạnh, biên

- Sử dụng các phương thức OpenCV cung cấp thực hiện đếm số lượng đồng xu trong ảnh

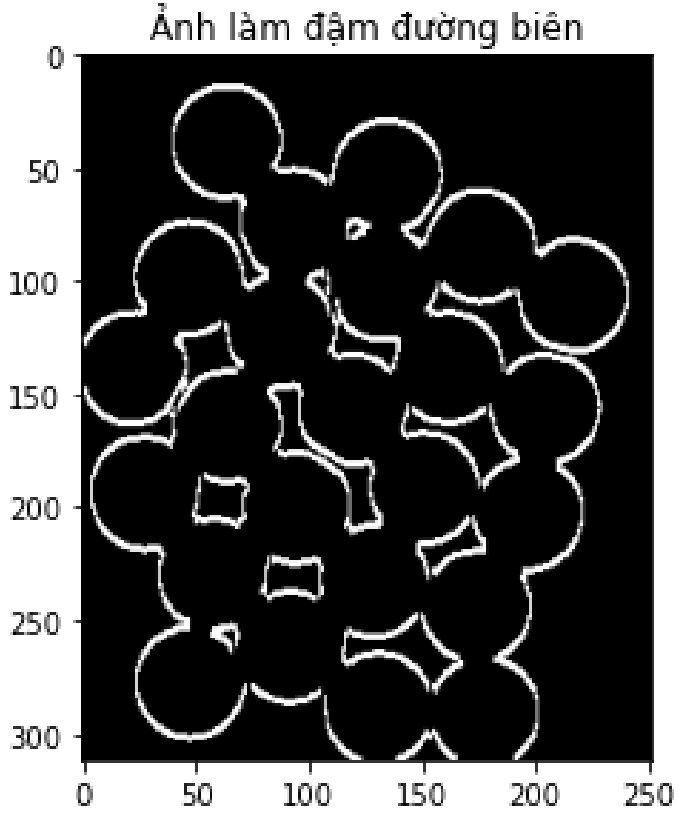
```
12  #Bước 2: Phát hiện biên với Canny:
13  #Thực hiện làm mịn ảnh với Gausian:
14  blur = cv2.GaussianBlur(gray, (11,11), 0)
15
16  #Phát hiện biên với Canny
17  canny = cv2.Canny(blur, 30, 150, 3)
18  plt.subplot(2,2,2)
19  plt.imshow(canny, cmap='gray')
20  plt.title('Ảnh biên với Canny')
```



Phân vùng ảnh dựa trên cạnh, biên

- Sử dụng các phương thức OpenCV cung cấp thực hiện đếm số lượng đồng xu trong ảnh

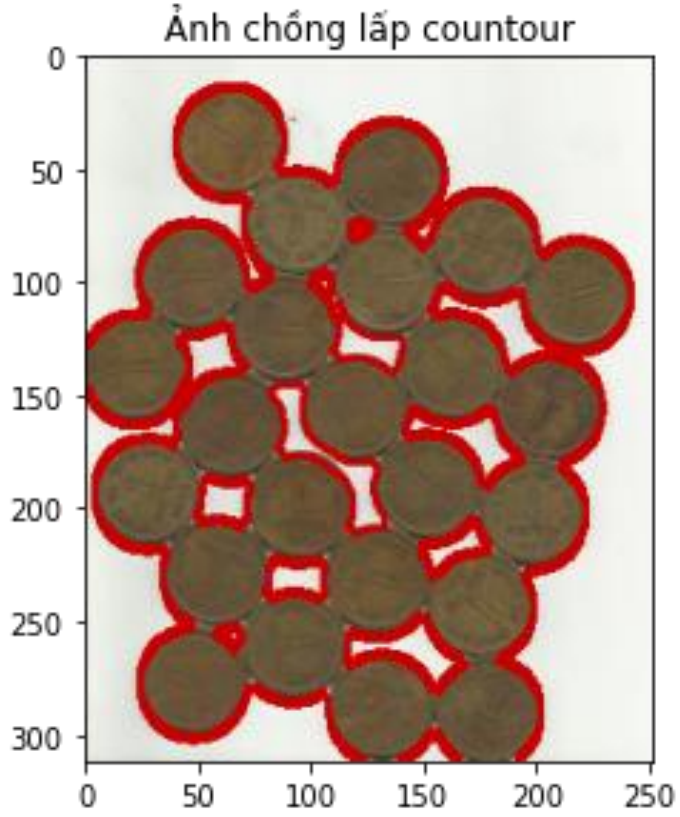
```
23  #Bước 3: Làm đậm đường biên:
24  dilated = cv2.dilate(canny, (1,1), iterations = 2)
25  plt.subplot(2,2,3)
26  plt.imshow(dilated, cmap='gray')
27  plt.title('Ảnh làm đậm đường biên')
```



Phân vùng ảnh dựa trên cạnh, biên

- Sử dụng các phương thức OpenCV cung cấp thực hiện đếm số lượng đồng xu trong ảnh

```
29 #Bước 4: Lấy countour của ảnh và đếm số lượng:
30 (cnt, heirarchy) = cv2.findContours(dilated.copy(),
31                                     cv2.RETR_EXTERNAL,
32                                     cv2.CHAIN_APPROX_NONE)
33
34 rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
35 cv2.drawContours(rgb, cnt, -1, (200,0,0), 2)
36
37 plt.subplot(2,2,4)
38 plt.imshow(rgb)
39 plt.title('Ảnh chồng lấp countour')
40 plt.show()
41
42 print('Số lượng đồng xu trong ảnh là: ', len(cnt))
```




```
Số lượng đồng xu trong ảnh là:  24
```

Thực hành 4.2

Thực hành

1. Đọc và hiển thị ảnh Thuchanh4_2.jpg ở dạng ảnh màu.
2. Sử dụng phương pháp Canny với các tham số phù hợp phát hiện biên của các đối tượng trong ảnh.
3. Đếm số lượng bóng bay trong bức ảnh với OpenCV

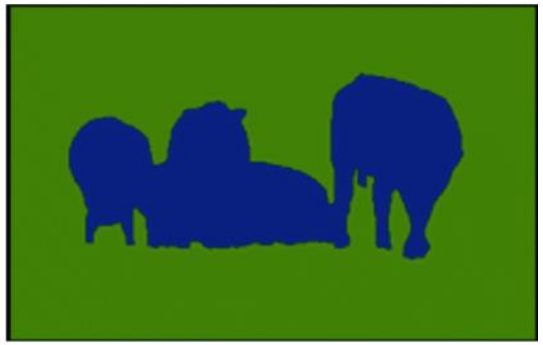
3.3. Phân vùng ảnh dựa trên kỹ thuật phân cụm

Phân vùng ảnh dựa trên kỹ thuật phân cụm

- Phân cụm (Clustering) là một loại thuật toán học máy không giám sát, được sử dụng phổ biến trong phân vùng ảnh.

- Một trong những thuật toán Clustering thường được ứng dụng cho tác vụ phân vùng ảnh là KMeans Clustering. Loại phân cụm này có thể được sử dụng để tạo các phân đoạn trong một hình ảnh có màu.

a



b

Phân vùng ảnh dựa trên kỹ thuật phân cụm

- OpenCV cung cấp phương thức kmeans để thực hiện phân cụm

ret,label,center=cv2.kmeans(samples, K, criteria, attempts, flags)

Tham số đầu vào:
1.samples : Mẫu thì nên là kiểu dữ liệu của np.float32, và mỗi đối tượng cần được đặt trong cột duy nhất.
2.nclusters (K): Số nhóm yêu cầu tại thời điểm cuối cùng
3.criteria : Đây là tiêu chí chấm dứt vòng lặp. Khi tiêu chí này được thỏa mãn, thuật toán sẽ dừng vòng lặp. Trên thực tế, nó phải là một nhóm 3 thông số (type, max_iter, epsilon):

1. 3.a - loại tiêu chí chấm dứt: Nó có 3 lá cờ như sau:
    1. cv2.TERM_CRITERIA_EPS - ngăn sự lặp lại thuật toán nếu đạt được độ chính xác nhất định - epsilon đạt được.
    2. cv2.TERM_CRITERIA_MAX_ITER - dừng thuật toán sau khi số lượng nhất định được lặp đi lặp lại.
    3. cv2.TERM_CRITERIA_EPS max_iter + cv2.TERM_CRITERIA_MAX_ITER - ngăn chặn sự lặp đi lặp lại khi một trong các điều kiện trên nó được đáp ứng
2. 3.b - max_iter - Một số nguyên xác định số lượng tối đa vòng lặp.
3. 3.c - Độ chính xác - epsilon

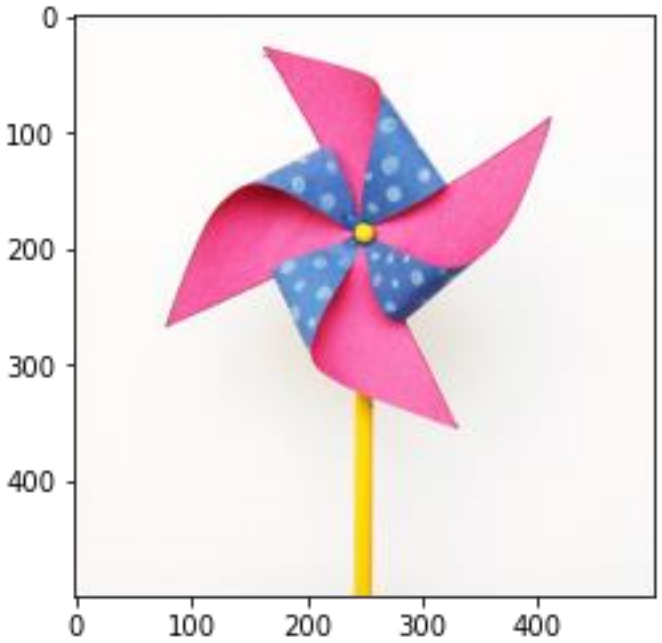
4.attempts: Cờ để xác định số lần các thuật toán được thực hiện bằng cách sử dụng việc đánh nhãn khởi tạo khác nhau.
5.flags : Cờ này được sử dụng để xác định trung tầm ban đầu được chọn như thế nào. Về cơ bản có 2 cờ được sử dụng là : cv2.KMEANS_PP_CENTERS và cv2.KMEANS_RANDOM_CENTERS.

Tham số đầu ra:
1.compactness : Đây là tổng của bình phương khoảng cách từ mỗi điểm đến trọng tâm tương tự của họ.
2.labels : Đây là mảng các label trong đó mỗi phần tử của mảng được đánh dấu '0', '1' .....
3.centers : Đây là mảng trọng tâm của các nhóm.

Phân vùng ảnh dựa trên kỹ thuật phân cụm

- Bước 1: Đọc ảnh màu cần thực hiện phân vùng

```
# Đọc ảnh màu
img = cv2.imread('images/pic2.jpeg')

#chuyển sang hệ màu RGB
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.show()
```



- Bước 2: Biến đổi dữ liệu

```
#Chuyển đổi mảng 3 chiều về ma trận, mỗi cột ứng với một kênh màu
twoDimage = img.reshape((-1,3))
twoDimage = np.float32(twoDimage)
```

```
twoDimage
```

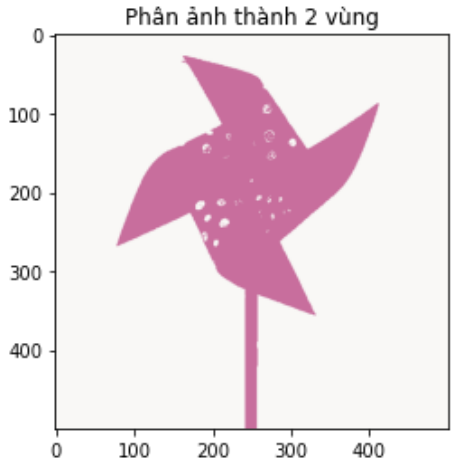
```
array([[253., 252., 250.],
       [253., 252., 250.],
       [253., 252., 250.],
       ...,
       [246., 246., 246.],
       [246., 246., 246.],
       [246., 246., 246.]], dtype=float32)
```

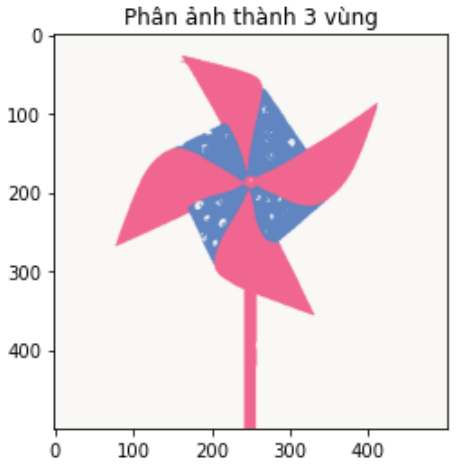
Phân vùng ảnh dựa trên kỹ thuật phân cụm

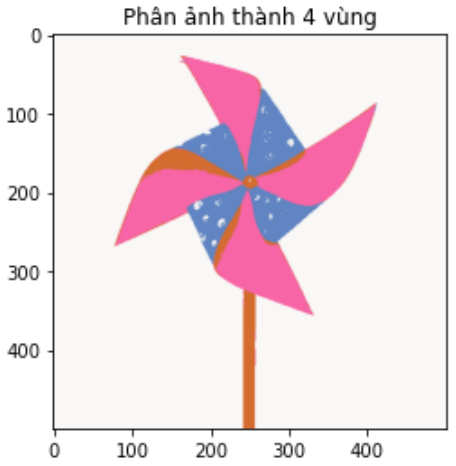
- Bước 3: Thiết lập các tham số và thực hiện phân vùng ảnh với Kmeans

```
#Thiết lập các tham số của cụm
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 1.0)
K = 2
attempts=50
```

```
#Sử dụng phương thức cv2.kmeans để phân cụm
ret,label,center=cv2.kmeans(twoDimage,
                            K,
                            None,
                            criteria,
                            attempts,
                            cv2.KMEANS_PP_CENTERS)
center = np.uint8(center)
res = center[label.flatten()]
result_image = res.reshape((img.shape))
```





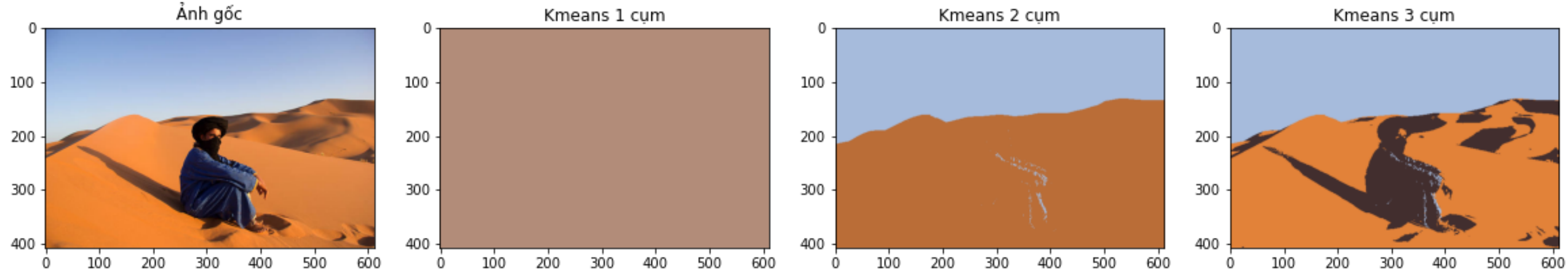


Thực hành 4.3

Thực hành

1. Đọc và hiển thị ảnh Thuchanh4_3.jpeg ở dạng ảnh màu.
2. Sử dụng phương pháp phân vùng ảnh dựa trên kỹ thuật phân cụm Kmeans để phân ảnh ra thành:
    * 1 cụm
    * 2 cụm
    * 3 cụm

* Hiển thị kết quả: ảnh 1 cụm - ảnh 2 cụm - ảnh 3 cụm

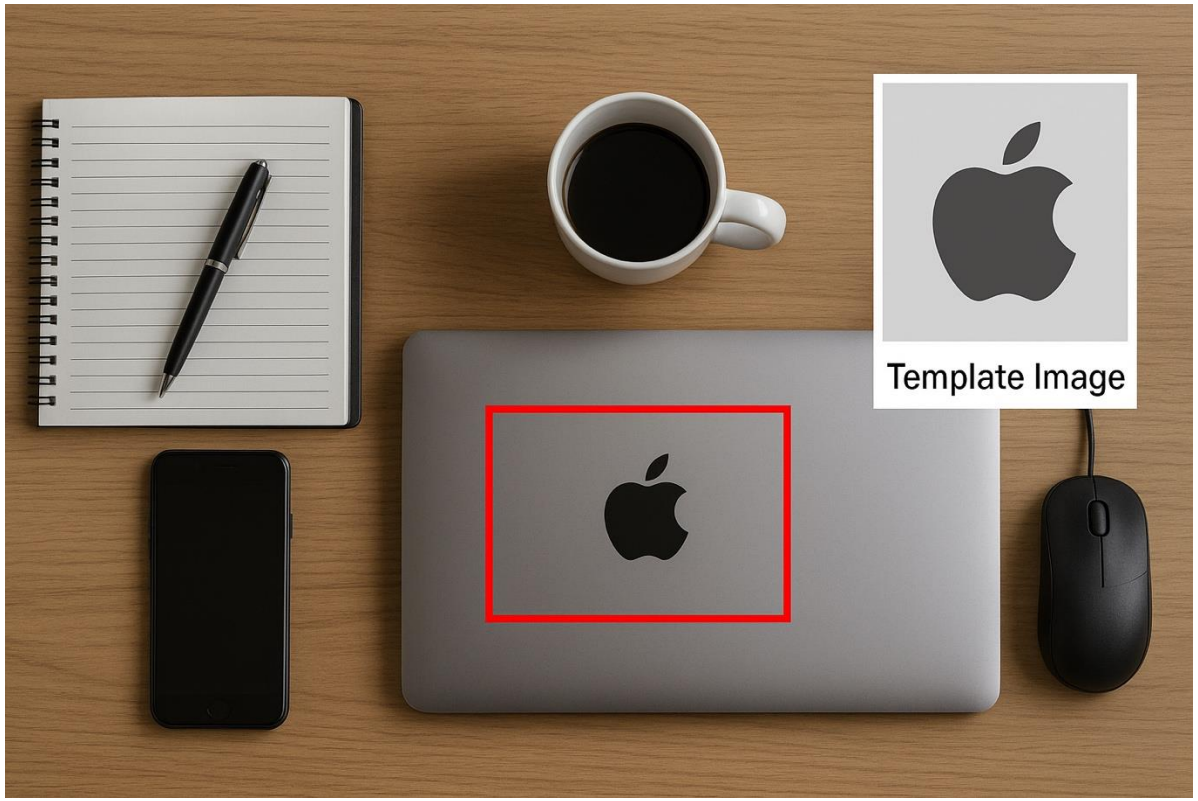


4. Tìm kiếm vùng trong ảnh

Tìm kiếm vùng trong ảnh

- Trong nhiều bài toán thị giác máy tính, chúng ta cần xác định xem **một mẫu đối tượng cụ thể** có xuất hiện trong một bức ảnh lớn hay không, và nếu có thì **vị trí của nó nằm ở đâu**.
- Bài toán này xuất hiện phổ biến trong:
  - Nhận dạng logo, nhãn hiệu, icon trong giao diện hoặc ảnh sản phẩm.
  - Xác định vị trí linh kiện trong dây chuyền sản xuất tự động.
  - Tìm khuôn mặt, con dấu, chữ ký trong tài liệu số hóa.
  - Định vị một mẫu đối tượng tĩnh trong ảnh vệ tinh hoặc ảnh camera giám sát.

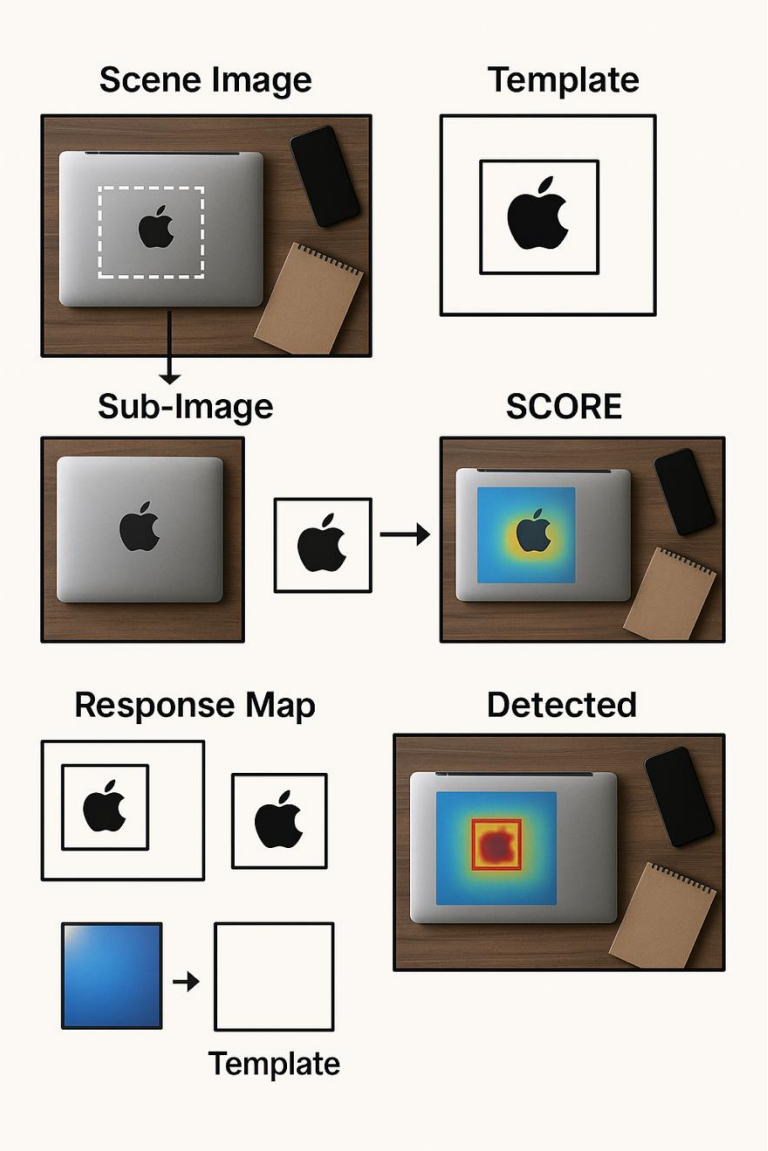
Đây là bài toán cơ bản nhưng có vai trò quan trọng trong nhiều hệ thống nhận dạng và kiểm tra chất lượng.



Tìm kiếm vùng trong ảnh

- **Nguyên lý cốt lõi của bài toán:**
  1. Ảnh mẫu được dịch chuyển lần lượt qua mọi vị trí khả dĩ của ảnh gốc.
  2. Tại mỗi vị trí, trích ra một vùng con của ảnh gốc với kích thước bằng template.
  3. So sánh vùng con này với ảnh mẫu bằng một thước đo tương đồng.
  4. Tạo ra một ma trận điểm số (response map).
  5. Vị trí có điểm số cao (hoặc thấp, tùy thước đo) → vị trí khớp tốt nhất.

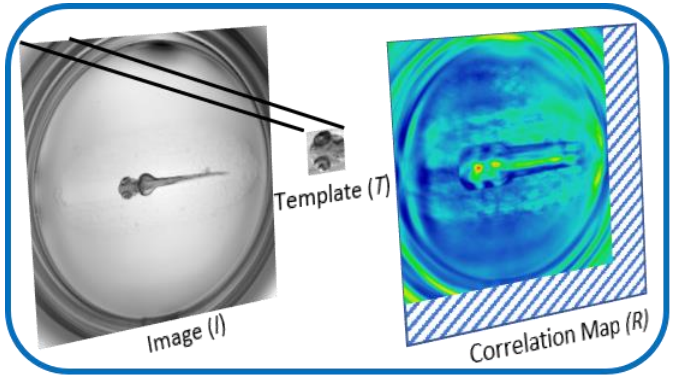
Cách tiếp cận này được gọi là **Template Matching**, và là nền tảng cho nhiều thuật toán phức tạp hơn sau này.

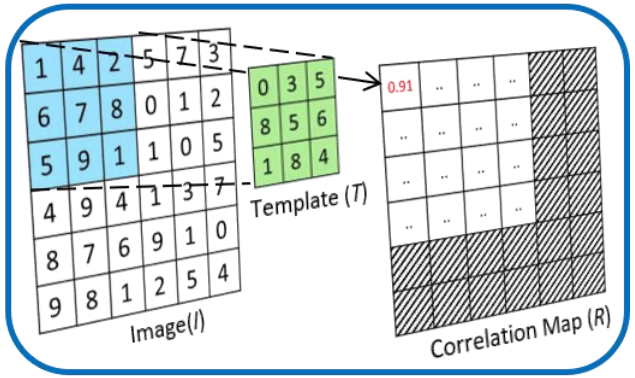


Tìm kiếm vùng trong ảnh

Đặc điểm của bài toán:

- Đây là phương pháp **so khớp theo cường độ pixel**, do đó phù hợp trong môi trường mà:
  - đối tượng không thay đổi kích thước,
  - không quay, không bị méo dạng,
  - ánh sáng tương đối ổn định,
  - nền ít nhiễu.
- Phương pháp đơn giản, dễ triển khai, và hiệu quả đối với các tác vụ kiểm tra hình ảnh trong môi trường kiểm soát tốt.





Tìm kiếm vùng trong ảnh

Sử dụng OpenCV để thực hiện Template matching:

result = cv2.matchTemplate(image, template, method)

Tham số:

- **image**: Ảnh gốc (ảnh lớn hơn).
- **template**: Ảnh mẫu cần tìm (kích thước nhỏ hơn ảnh gốc).
- **method**: Phương pháp so khớp.

Phương pháp	Ý nghĩa	Nhận xét
TM_SQDIFF	Tổng sai số bình phương	Giá trị nhỏ nhất là khớp tốt nhất
TM_SQDIFF_NORMED	Chuẩn hóa SQDIFF	0 → khớp hoàn hảo
TM_CCORR	Tương quan chéo	Giá trị lớn nhất là tốt
TM_CCORR_NORMED	Chuẩn hóa CCORR	Thường dùng
TM_CCOEFF	Tương quan hệ số	Bỏ ảnh hưởng độ sáng
TM_CCOEFF_NORMED	Chuẩn hóa CCOEFF	**Phổ biến nhất**

Tìm kiếm vùng trong ảnh

Kết quả result: là ma trận điểm số (score map) có kích thước:

$$(W-w+1, H-h+1)(W - w + 1, H - h + 1)(W-w+1, H-h+1)$$

trong đó:

- (W,H) là kích thước ảnh gốc,
- (w,h) là kích thước ảnh mẫu.

• Mỗi phần tử trong result là **mức độ khớp** giữa vùng con trong ảnh và mẫu tại vị trí đó.

Sau khi tính result, để xác định vị trí khớp tốt nhất ta thường dùng:

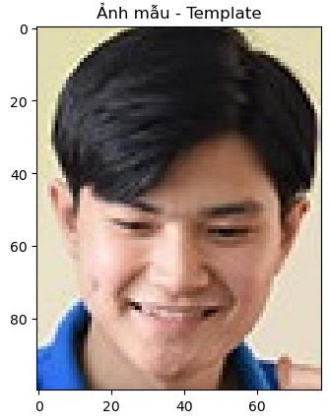
```
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
```

- Với **TM_SQDIFF** → vị trí khớp tốt nhất là min_loc.
- Với các phương pháp khác → vị trí khớp tốt nhất là max_loc.

Tìm kiếm vùng trong ảnh

- Bước 1: Đọc ảnh gốc và ảnh mẫu tìm kiếm

```
#Đọc ảnh gốc:
img = cv2.imread('images/humg.jpg',cv2.IMREAD_COLOR_RGB)
#Đọc ảnh mẫu cần tìm:
tem = cv2.imread('images/face1.jpg',cv2.IMREAD_COLOR_RGB)

#Hiển thị ảnh gốc - ảnh mẫu
print(tem.shape)
plt.imshow(img,cmap='gray')
plt.title('Ảnh gốc')
plt.show()

plt.imshow(tem,cmap='gray')
plt.title('Ảnh mẫu - Template')
plt.show()
```



Tìm kiếm vùng trong ảnh

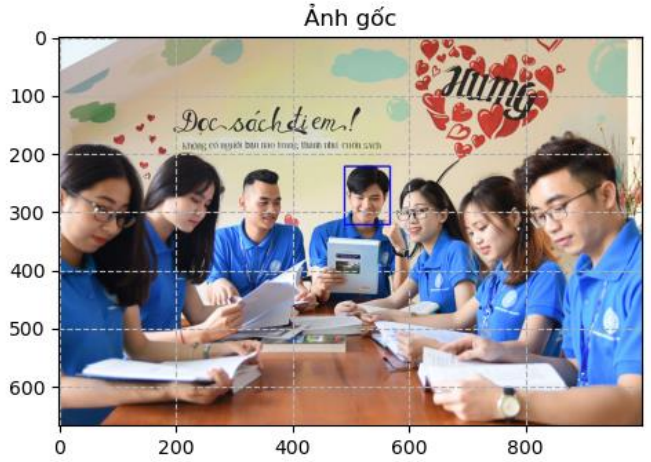
- Bước 2:
  - Chuyển sang ảnh xám để tăng tốc tìm kiếm;
  - Sử dụng hàm matchTemplate để tìm vùng phù hợp
  - Xác định vị trí của vùng khớp tốt nhất

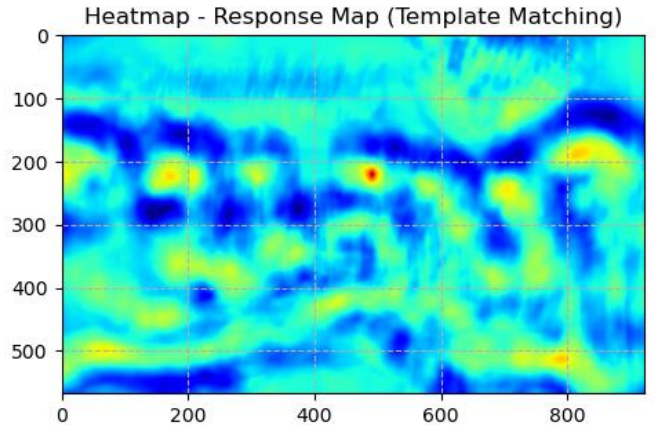
```
# Chuyển ảnh sang Xám để tăng tốc và ổn định
img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
temp_gray = cv2.cvtColor(temp, cv2.COLOR_RGB2GRAY)

#Sử dụng phương thức matchTemplate để tìm vùng phù hợp:
result = cv2.matchTemplate(img_gray,temp_gray, cv2.TM_CCOEFF)

#Xác định vị trí khớp tốt nhất:
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
```

Tìm kiếm vùng trong ảnh

- Bước 3: Hiển thị kết quả tìm kiếm

```
#Hiển thị kết quả tìm kiếm:
#1. lấy kích thước của ảnh mẫu
h, w = tem.shape[:2]

#2.Lấy tọa độ góc trên trái của vùng khớp
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)

#3. Vẽ khung chữ nhật quanh vùng khớp
cv2.rectangle(img, top_left, bottom_right, (0, 0, 255), 2)

#4. Hiển thị kết quả:
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.imshow(img)
plt.title("Ảnh gốc")
plt.grid(ls='--')

# Vẽ Response Map bằng Heatmap
plt.subplot(1,2,2)
plt.imshow(result, cmap='jet')
#plt.colorbar(label="Matching Score")
plt.title("Heatmap - Response Map (Template Matching)")
plt.grid(ls='--')
plt.show()
```





Thực hành 4.4

Thực hành

1. Đọc và hiển thị ảnh Thuchanh4_4.jpg.
2. Thực hiện tìm kiếm logo Apple trong ảnh


QUESTIONS
Q&A
ANSWERS