ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO FINAL PROJECT
LẬP TRÌNH DSP

CHỦ ĐỀ

IMAGE SEGMENTATION

GV hướng dẫn: Nguyễn Hồng Thịnh

Nhóm 2:

Họ và tên	*MSSV*
Ngô Minh Đức	20021517
Phùng Đình Thành	20021582
Đỗ Xuân Hiểu	20021527
Nguyễn Sỹ Sơn	20021578

Hà Nội, 2023

FINAL PROJECT - LẬP TRÌNH DSP
IMAGE SEGMENTATION

Ngô Minh Đức, Phùng Đình Thành, Đỗ Xuân Hiểu, Nguyễn Sỹ Sơn

Ngày 29 tháng 11 năm 2023

Tóm tắt nội dung

Phân đoạn ảnh (Image segmentation) đóng một vai trò quan trọng trong xử lý ảnh số. Đó là quá trình chia hình ảnh thành nhiều phân đoạn khác nhau được gọi là đối tượng hình ảnh (image objects), từ đó làm giảm độ phức tạp của hình ảnh để phân tích chúng một cách hiệu quả. Mục đích là để thể hiện một hình ảnh không bị nhiễu và biến dạng. Hơn nữa, nó được sử dụng để xác định vị trí của các đối tượng và ranh giới của chúng. Nhiều kỹ thuật phân đoạn ảnh đã được nghiên cứu và sử dụng trước đó. Trong dự án này, nhóm trình bày các kỹ thuật phân đoạn ảnh khác nhau như: Region-Based Segmentation (Phân bổ đặc trưng), Edge Detection Segmentation (Cắt góc), Segmentation based on Clustering (KMeans). Từ đó so sánh giữa các phương pháp nói trên

Từ khóa: Phân đoạn ảnh, Segmentation, Region-Based Segmentation, Edge Detection Segmentation, Clustering, KMeans, Threshold, Canny Edge, Sobel,..

1 Giới thiệu

Trong những năm gần đây, cùng với sự phát triển vượt bậc của khoa học máy tính, thì xử lý ảnh là một là một lĩnh vực đang được quan tâm. Để phân tích các đối tượng trong ảnh, chúng ta cần phải phân biệt được các đối tượng cần quan tâm với phần còn lại của ảnh. Những đối tượng này có thể tìm ra được nhờ các kỹ thuật phân đoạn ảnh, theo nghĩa tách phần tiền cảnh ra khỏi hậu cảnh trong ảnh. Có thể hiểu phân vùng là tiến trình chia ảnh thành nhiều vùng, mỗi vùng chứa một đối tượng hay nhóm đối tượng cùng kiểu. Phân vùng ảnh còn là một thao tác ở mức thấp trong toàn bộ quá trình xử lý ảnh. Quá trình này thực hiện việc phân vùng ảnh thành các vùng rời rạc và đồng nhất với nhau hay nói cách khác là xác định các biên của các vùng ảnh đó. Các vùng ảnh đồng nhất này thông thường sẽ tương ứng với tòan bộ hay từng phần của các đối tượng thật sự bên trong ảnh. Vì thế, trong hầu hết các ứng dụng của lĩnh vực xử lý ảnh (image processing), thị giác máy tính, phân vùng ảnh luôn đóng một vai trò cơ bản và thường là bước tiền xử lý đầu tiên trong toàn bộ quá trình trước khi thực hiện các thao tác khác ở mức cao hơn như nhận dạng đối tượng, biểu diễn đối tượng, nén ảnh dựa trên đối tượng, hay truy vấn ảnh dựa vào nội dung. Do đó, nhóm sẽ nghiên cứu và so sánh một số phương pháp phân đoạn hình ảnh

2 Tổng quan về Image Segmentation

2.1 Giới thiệu tổng quan

Image Segmentation là một lĩnh vực quan trọng của thị giác máy tính, tập trung vào việc phân chia hình ảnh thành các phần nhỏ hơn, những vùng quan trọng và ý nghĩa. Điều này cho phép chúng ta hiểu rõ hơn về cấu trúc và nội dung của hình ảnh, mang lại sự linh hoạt và chi tiết trong quá trình xử lý hình ảnh. Trong khi các phương pháp truyền thống hướng tới việc phân loại hình ảnh toàn bộ, Image Segmentation mở rộng khả năng này bằng cách chú trọng vào việc định rõ đường biên và vùng lân cận.

2.2 Vai Trò Quan Trọng của Image Segmentation

Image Segmentation chơi một vai trò không thể phủ nhận trong nhiều lĩnh vực ứng dụng, mang lại những đóng góp quan trọng về hiểu biết và tương tác với thế giới xung quanh. Dưới đây là những khía cạnh cụ thể về vai trò của Image Segmentation:

• Y Tế: Trong lĩnh vực y tế, Image Segmentation đóng một vai trò quan trọng trong việc phân đoạn và xác định vị trí của các cơ quan và cấu trúc trong hình ảnh y tế. Chẳng hạn, trong quá trình chẩn đoán hình ảnh y tế, Image Segmentation giúp xác định ranh giới của các tế bào, mô và cơ quan, làm tăng độ chính xác và nhanh chóng của quá trình chẩn đoán.

• Công Nghiệp Tự Động Hóa: Trong lĩnh vực công nghiệp tự động hóa, Image Segmentation là một công cụ quan trọng để nhận biết và phân loại đối tượng. Trong hệ thống lái xe tự động, nó giúp xe ô tô xác định và theo dõi các phương tiện và vật thể xung quanh, tăng cường khả năng an toàn và hiệu suất của hệ thống.

• Kinh tế, đời sống: Trong thị trường kinh tế, Image Segmentation là chìa khóa để tạo ra trải nghiệm người dùng tốt hơn. Từ nhận dạng khuôn mặt cho đến phân loại sản phẩm, Image Segmentation giúp cá nhân và doanh nghiệp tối ưu hóa quá trình tìm kiếm và tương tác với hình ảnh.

• Nghiên Cứu Dược: Trong lĩnh vực nghiên cứu dược, Image Segmentation đóng vai trò quan trọng trong việc phân loại và đánh giá hiệu suất của các phác đồ thử nghiệm. Việc tự động hóa quy trình phân tích hình ảnh giúp nâng cao hiệu suất và độ chính xác của các nghiên cứu, đồng thời giảm thời gian và nguy cơ sai sót.

2.3 Tiến bộ đột phá của Deep Learning trong Image Segmentation

Những tiến bộ đáng kể trong lĩnh vực Image Segmentation đến từ sự phát triển của Deep Learning, đặc biệt là sự ra đời và ứng dụng rộng rãi của Convolutional Neural Networks (CNNs). Các mô hình này không chỉ có khả năng học được các đặc trưng phức tạp mà còn giảm thiểu sự phụ thuộc vào các đặc trưng được định nghĩa thủ công. Sự tích hợp sâu rộng này đã cung cấp cơ hội mới và đưa Image Segmentation từ khái niệm nghiên cứu sang ứng dụng thực tế, mở ra tiềm năng lớn cho sự đổi mới và tiến bộ trong tương lai.

3 Phương pháp Image segmentation

3.1 Phân Loại Các Phương Pháp Truyền Thống

3.1.1 Phương pháp Phân Cụm (Clustering)

Phương pháp này tập trung vào việc phân chia các điểm dữ liệu thành các nhóm (cụm) dựa trên đặc trưng nào đó. Trong Image Segmentation, phân cụm có thể áp dụng các thuật toán như K-Means để gom nhóm các pixel có giá trị màu sắc tương đồng.

3.1.2 Phương Pháp Dựa trên cạnh (Edge-Based)

Các phương pháp này tìm kiếm đường biên và ranh giới trong hình ảnh để phân chia các vùng. Sự thay đổi cục bộ của độ sáng hoặc cường độ màu sắc giữa các khu vực được sử dụng để xác định các đường biên. Tuy nhiên, chúng có thể nhạy cảm với nhiễu và đòi hỏi một lượng lớn tính toán.

3.2 Deep Learning trong Image Segmentation

3.2.1 Mạng Neuron Convolutions (CNNs)

CNNs đã chứng minh sức mạnh của mình trong việc giải quyết bài toán Image Segmentation. Những mô hình này sử dụng các lớp convolution để học các đặc trưng ảnh và tạo ra các đặc trưng hiệu quả cho việc phân loại từng pixel. U-Net là một kiến trúc nổi bật, thường được sử dụng trong các ứng dụng y tế.

3.2.2 Mạng Neuron Convolutions dựa trên Transfer Learning

Transfer Learning giúp giải quyết vấn đề thiếu dữ liệu bằng cách sử dụng các mô hình đã được huấn luyện trước đó trên tập dữ liệu lớn. Với việc sử dụng trọng số đã học được từ các nhiệm vụ như nhận diện hình ảnh chung, mô hình có thể hiệu quả hóa quá trình học và tăng cường khả năng tổng quát hóa.

3.2.3 Mạng Neuron Convolutions 3D (3D CNNs)

Trong ngữ cảnh của Image Segmentation cho dữ liệu không gian 3D như hình ảnh y tế, 3D CNNs được thiết kế để xử lý thông tin không chỉ theo chiều rộng và chiều cao mà còn theo chiều sâu. Điều này đặc biệt quan trọng trong việc phân đoạn các cấu trúc phức tạp như cơ quan trong cơ thể.

3.3 Phương pháp hỗn hợp

Kết hợp các phương pháp truyền thống và Deep Learning

Một số nghiên cứu tiến xa bằng cách kết hợp các phương pháp truyền thống với Deep Learning để tận dụng những lợi ích của cả hai. Chẳng hạn, việc sử dụng kết quả của K-Means Clustering như là một đặc trưng đầu vào cho một mô hình CNN có thể cải thiện độ chính xác và ổn định của phương pháp.

3.4 Tầm quan trọng của lựa chọn phương pháp

Việc chọn lựa phương pháp Image Segmentation phù hợp phụ thuộc vào bối cảnh ứng dụng cụ thể và yêu cầu của dự án. Đối với dữ liệu ít và yêu cầu thời gian tính toán thấp, các phương pháp truyền thống có thể là lựa chọn phù hợp. Ngược lại, khi có sẵn một lượng lớn dữ liệu và đòi hỏi độ chính xác cao, Deep Learning trở thành lựa chọn hàng đầu, đặc biệt là khi sử dụng các mô hình đã được huấn luyện sẵn trên các tập dữ liệu lớn.

4 Các phương pháp sử dụng trong project

4.1 Phương pháp Threshold

Phương pháp Threshold

- Phương pháp Threshold Otsu là một phương pháp đơn giản nhưng mạnh mẽ trong việc phân đoạn hình ảnh. Nó dựa trên việc chọn một ngưỡng (threshold) tối ưu để tách biệt giữa các vùng trong hình ảnh. Thuật toán Otsu tự động tìm kiếm ngưỡng tối ưu bằng cách tối đa hóa phương sai giữa các lớp pixel của ảnh.

- Phương pháp Threshold Otsu được áp dụng để phân chia các vùng của tín hiệu số học, giúp xác định ranh giới giữa các đặc trưng và nền. Điều này quan trọng trong việc trích xuất thông tin quan trọng từ tín hiệu và làm sạch nhiễu.

4.2 Phương pháp Canny Edge

Phương pháp Canny Edge

- Canny Edge Detection là một phương pháp phổ biến để xác định đường biên trong hình ảnh. Nó sử dụng bộ lọc Gaussian để làm mờ ảnh và sau đó xác định đạo hàm độ lớn và hướng của biên sắc thái (gradient). Bằng cách sử dụng ngưỡng và hệ số đối chiếu, Canny Edge Detection tạo ra một ảnh nhị phân hiển thị các đường biên.

- Phương pháp Canny Edge được triển khai để phân biệt giữa các cạnh của tín hiệu. Điều này giúp xác định vị trí của các sự kiện đột ngột và đặc điểm nổi bật trong tín hiệu số.

4.3 Phương pháp K-Means

Phương pháp K-Means

- Phương pháp K-Means là một thuật toán phân loại dữ liệu không giám sát được sử dụng để phân chia dữ liệu thành các cụm dựa trên đặc trưng nào đó. Trong trường hợp của hình ảnh, K-Means có thể được áp dụng trực tiếp trên giá trị pixel để phân chia ảnh thành các cụm có màu sắc tương đồng.

- Phương pháp K-Means để phân loại các đặc trưng của tín hiệu thành các nhóm. Điều này giúp chúng tôi tạo ra các nhóm có ý nghĩa với các đặc điểm chung, tăng cường khả năng hiểu và xử lý của hệ thống.

4.4 Phương pháp Sobel

Phương pháp Sobel

- Phương pháp Sobel là một phương pháp sử dụng bộ lọc để xác định đạo hàm của hình ảnh. Nó chủ yếu được sử dụng để phát hiện cạnh trong hình ảnh bằng cách tính đạo hàm của hàm mức xám. Sobel sử dụng hai bộ lọc (một cho đạo hàm theo chiều ngang và một cho đạo hàm theo chiều dọc) để tìm gradient của hình ảnh.
- Trong phạm vi dự án DSP Segmentation, phương pháp Sobel được áp dụng để xác định độ chệch và sự thay đổi đột ngột trong tín hiệu số, giúp xác định các vùng có ý nghĩa và tạo ra biểu đồ chính xác của tín hiệu.

5 Mô tả chương trình

5.1 Hàm resize

Hình 1 là đoạn code Hàm image resize dimensions

```python
def image_resize_dimensions(image, ratio=1.0):
    scale_percent = ratio
    image_width = int(image.shape[1] * scale_percent)
    image_height = int(image.shape[0] * scale_percent)
    dim = (image_width, image_height)

    return cv2.resize(image, dim, interpolation=cv2.INTER_AREA)
```

Hình 1: Hàm image resize dimensions

- Hàm *image resize dimensions* được thiết kế để thay đổi kích thước của ảnh một cách linh hoạt bằng cách xác định tỷ lệ co giãn hoặc thu nhỏ.
- Hàm này quan trọng để tối ưu hóa quá trình xử lý ảnh. Việc co giãn hoặc thu nhỏ ảnh có thể giúp giảm độ phức tạp tính toán, tăng tốc quá trình xử lý, và giảm dung lượng lưu trữ. Đặc biệt, khi làm việc với ảnh đầu vào có kích thước lớn, việc resize trước khi xử lý có thể giúp tăng hiệu suất của ứng dụng.

5.2 Hàm Threshold

Hình 2 là đoạn code Hàm Threshold

```python
def image_threshold(image, threshold=127, adaptive=False, otsu=False, kernel_size=3):
    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    if adaptive:
        return cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, kernel_size, 0)
    elif otsu:
        ret_val, image = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
        return image
    else:
        ret_val, image = cv2.threshold(image, threshold, 255, cv2.THRESH_BINARY_INV)
        return image
```

Hình 2: Hàm Threshold

- Hàm *image threshold* thực hiện phương pháp Thresholding trên ảnh. Thresholding là một kỹ thuật xử lý ảnh phổ biến được sử dụng để chuyển đổi ảnh thành ảnh nhị phân. Ý tưởng cơ bản là đặt một ngưỡng, và mọi pixel có giá trị lớn hơn ngưỡng sẽ được gán giá trị cực đại, trong khi mọi pixel có giá trị nhỏ hơn hoặc bằng ngưỡng sẽ được gán giá trị tối thiểu.

- Trong trường hợp cụ thể của hàm này, có ba phương pháp Thresholding được hỗ trợ:

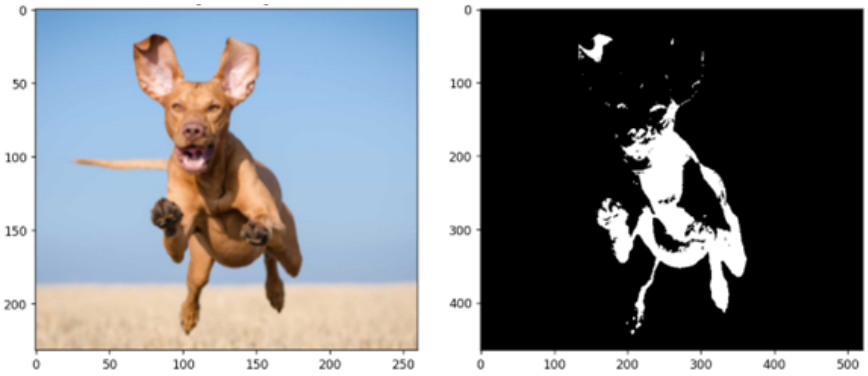
+ *Adaptive Thresholding:* Khi tham số *adaptive* được đặt là *True*. Đối với mỗi điểm ảnh, ngưỡng được xác định dựa trên giá trị trung bình của các pixel trong khu vực xung quanh (được xác định bởi kích thước kernel).

+ *Thresholding Otsu*: Khi tham số *otsu* được đặt là *True*. Thuật toán Otsu tự động xác định ngưỡng tối ưu để phân tách hai lớp pixel (nền và vật thể) dựa trên phương pháp tối ưu hóa phương sai giữa chúng.

+ *Thresholding Cơ Bản*: Khi cả hai tham số *adaptive* và *otsu* đều là *False*

Lưu ý: Tùy thuộc vào cách đặt các tham số, hàm này có thể thực hiện một trong ba phương pháp trên để chuyển đổi ảnh thành ảnh nhị phân theo ngưỡng xác định.

Hình 3 minh họa việc chuyển đổi ảnh thành ảnh nhị phân theo ngưỡng xác định



Hình 3: Chuyển đổi ảnh thành ảnh nhị phân theo ngưỡng xác định

5.3 Hàm sobel filter

Hình 4 là đoạn code Hàm Sobel Filter

```python
def image_sobel_gradient(image):
    if len(image.shape) == 3:
        full_grey = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    else:
        full_grey = np.copy(image)

    kernel_size = 3
    gradient_x = cv2.Sobel(full_grey, ddepth=cv2.CV_32F, dx=1, dy=0, ksize=kernel_size)
    gradient_y = cv2.Sobel(full_grey, ddepth=cv2.CV_32F, dx=0, dy=1, ksize=kernel_size)

    gradient_x = cv2.convertScaleAbs(gradient_x)
    gradient_y = cv2.convertScaleAbs(gradient_y)

    combined = cv2.addWeighted(gradient_x, 0.5, gradient_y, 0.5, 0)

    return combined
```

Hình 4: Hàm Sobel Filter

Hàm image sobel gradient thực hiện việc áp dụng bộ lọc Sobel để tính toán đạo hàm cục bộ theo chiều x và chiều y trên ảnh đầu vào, sau đó kết hợp chúng để tạo ra ảnh gradient.

Chuyển đổi ảnh thành ảnh xám (nếu cần):

- Nếu ảnh đầu vào có ba kênh màu (ảnh màu), hàm sẽ chuyển đổi nó thành ảnh xám bằng cách sử dụng cv2.cvtColor với mã màu cv2.COLOR RGB2GRAY.
- Nếu ảnh đầu vào đã là ảnh xám, hàm sẽ tạo một bản sao của nó để tránh thay đổi ảnh gốc.

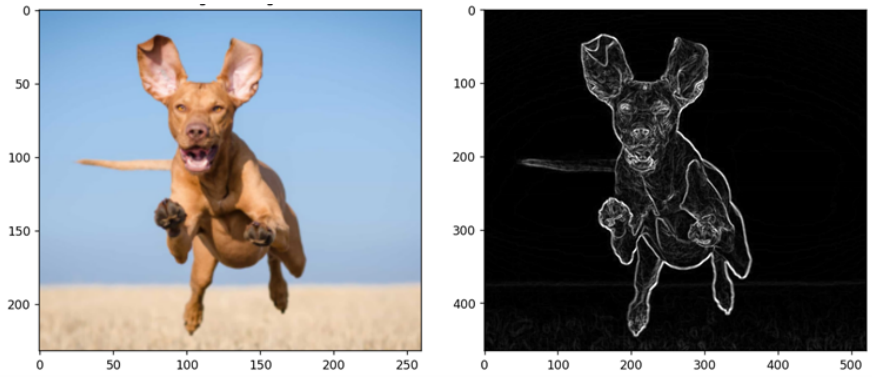
Áp dụng bộ lọc Sobel để tính đạo hàm cục bộ:

- Sử dụng hàm cv2.Sobel để tính toán đạo hàm cục bộ theo chiều x và chiều y.
- Kích thước kernel (ksize) được đặt là 3, nhưng có thể được điều chỉnh tùy thuộc vào độ phức tạp của biểu đồ đạo hàm bạn muốn. Kích thước kernel lớn hơn thì sẽ làm tăng cường độ phức tạp của phép toán, nhưng cũng có thể làm tăng thời gian tính toán. Kích thước kernel nhỏ hơn thì sẽ tạo ra hiệu ứng mượt mà hơn, nhưng có thể làm giảm độ chính xác của một số phép toán.

Chuyển đổi đạo hàm cục bộ thành giá trị tuyệt đối và kết hợp chúng:

- Sử dụng cv2.convertScaleAbs để chuyển đổi giá trị của đạo hàm cục bộ thành giá trị tuyệt đối và chuyển đổi sang kiểu dữ liệu 8-bit unsigned integer.
- Sử dụng cv2.addWeighted để kết hợp đạo hàm cục bộ theo chiều x và chiều y với trọng số 0.5 cho mỗi chiều, và không có trọng số cộng thêm vào.

Hình 5 minh họa sử dụng hàm Sobel



Hình 5: Sử dụng hàm Sobel

5.4 Hàm Canny Edge

Hình 6 là đoạn code Hàm Canny Edge

```
def image_canny_edges(image):
    med_val = np.median(image)
    lower = int(max(0, .7 * med_val))
    upper = int(min(255, 1.3 * med_val))
    edges = cv2.Canny(image=image, threshold1=lower, threshold2=upper)
    return edges
```

Hình 6: Hàm Canny Edge

Hàm *image canny edges* thực hiện phương pháp Canny Edge Detection để tìm ra các cạnh trong ảnh.

Xác định Ngưỡng Dưới và Ngưỡng Trên:

- Lấy giá trị trung bình của pixel trong ảnh (*med val*) sử dụng hàm *np.median. med val* đại diện cho giá trị trung bình (*median*) của pixel trong ảnh đầu vào.

+ Giá trị trung bình là một thước đo thống kê giúp đại diện cho mức độ sáng tối trung bình của ảnh. Trong trường hợp phương pháp Canny Edge Detection, nó thường được sử dụng để đặt ngưỡng cho việc phát hiện cạnh.

+ Cụ thể, ngưỡng dưới và ngưỡng trên sẽ được xác định dựa trên giá trị trung bình này.

- Xác định ngưỡng dưới (*lower*) và ngưỡng trên (*upper*) dựa trên giá trị trung bình.

+ lower được tính là 70 phần trăm của giá trị trung bình, giới hạn tối thiểu là 0.

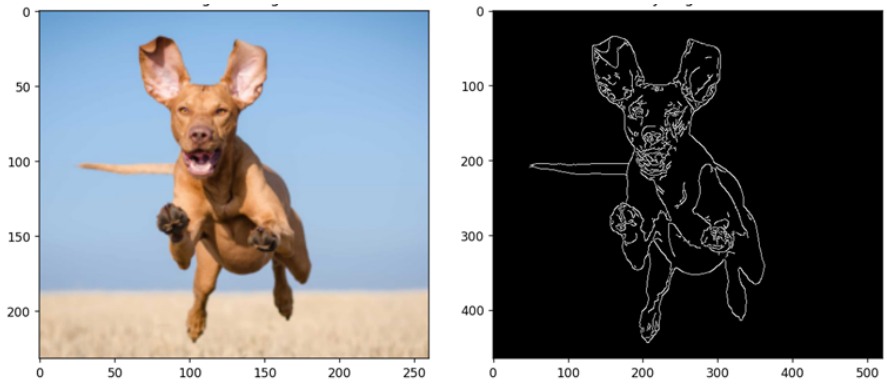
+ upper được tính là 130 phần trăm của giá trị trung bình, giới hạn tối đa là 255.

Sử dụng Hàm Canny để Phát hiện Cạnh:

- Áp dụng phương pháp Canny với ngưỡng dưới và ngưỡng trên đã xác định trước đó.

- Hàm cv2.Canny trả về một ảnh nhị phân, trong đó pixel có giá trị lớn đại diện cho các điểm cạnh.

Hình 7 minh họa ứng dụng của Hàm Canny Edge



Hình 7: Ứng dụng của Hàm Canny Edge

5.5 Hàm KMeans Clustering

Hình 8 là đoạn code Hàm KMeans Clustering

```
def image_kmean_segmentation(image, k):
    color = np.array(
        [[0, 255, 0], [255, 0, 0], [0, 0, 255], [255, 255, 0], [0, 255, 255], [255, 0, 255], [119, 159, 68],
         [97, 220, 151], [228, 164, 166], [191, 75, 184], [229, 40, 22], [243, 177, 234], [96, 43, 149], [113, 39, 186],
         [131, 227, 133], [203, 15, 240], [181, 110, 167], [187, 63, 206], [202, 70, 10], [43, 146, 61], [185, 10, 209],
         [28, 79, 72], [75, 183, 187], [135, 125, 93], [253, 76, 44], [212, 9, 132], [126, 215, 56], [84, 198, 179],
         [115, 104, 183], [243, 188, 33], [29, 150, 16], [6, 224, 62], [150, 92, 249], [249, 106, 81], [15, 91, 39],
         [51, 210, 91], [110, 81, 133], [102, 155, 71], [135, 35, 102], [165, 157, 110], [121, 221, 60], [152, 193, 20],
         [163, 222, 237], [177, 97, 149], [55, 23, 226], [114, 54, 212], [68, 73, 88], [128, 53, 147], [214, 19, 144],
         [98, 165, 163], [53, 170, 70], [108, 15, 97], [5, 250, 78], [65, 6, 215], [152, 55, 172], [101, 198, 200],
         [87, 109, 216], [233, 240, 202], [46, 44, 128], [184, 247, 112], [75, 33, 136], [189, 143, 210],
         [123, 90, 167], [83, 35, 232], [182, 187, 68], [92, 199, 225], [182, 56, 22], [122, 223, 138], [233, 166, 43],
         [113, 81, 231], [245, 189, 2], [11, 127, 78], [118, 82, 157], [41, 47, 48], [113, 224, 107], [156, 7, 203],
         [25, 228, 33], [104, 141, 56], [74, 7, 244], [28, 85, 27], [45, 109, 211], [228, 255, 8], [23, 194, 114],
         [32, 225, 32], [25, 30, 126], [83, 163, 112], [137, 143, 65], [20, 52, 218], [167, 13, 230], [38, 0, 117],
         [70, 102, 249], [93, 20, 233], [31, 248, 67]])

    color = np.uint8(color)
    image_2d = image.reshape((-1, 3))
    image_2d = np.float32(image_2d)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    ret, label, center = cv2.kmeans(image_2d, k, None, criteria=criteria, attempts=10, flags=cv2.KMEANS_PP_CENTERS)
    segmented_data = color[label.flatten()]

    return segmented_data.reshape(image.shape)
```

Hình 8: Hàm KMeans Clustering

Bước 1: Đầu tiên, tạo ra một mảng numpy có tên là color, trong đó mỗi hàng của mảng đại diện cho một màu sắc trong hệ thống màu RGB. Mỗi hàng của mảng là một mảng 1 chiều với 3 phần tử, tương ứng với giá trị Red, Green và Blue của màu sắc.

- Ví dụ, [0, 255, 0] đại diện cho màu xanh lá cây, với giá trị Red là 0, Green là 255 và Blue là 0.
- Các giá trị của mỗi phần tử nằm trong khoảng từ 0 đến 255, là giá trị tối đa mà một kênh màu có thể có trong hệ thống màu RGB.
- Mảng này thường được sử dụng để gán màu cho các cụm trong phân cụm màu sắc, như trong thuật toán K-means phân cụm màu.
- color được chuyển về kiểu dữ liệu uint8 để phù hợp với định dạng ảnh.
- Ảnh đầu vào image được chuyển sang kích thước (-1, 3), trong đó mỗi hàng là một vector 3 chiều đại diện cho một điểm ảnh. Sau đó được chuyển tiếp thành kiểu dữ liệu float32 để phù hợp với đầu vào của hàm K-means.

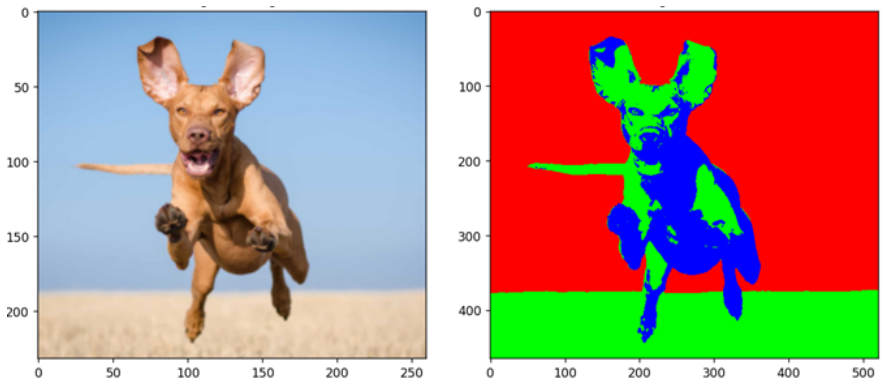
Bước 2: Áp Dụng Thuật Toán K-means:

- Các tham số cho thuật toán K-means được xác định:
- k: Số lượng cụm mong muốn.
- criteria: Điều kiện dừng của thuật toán.
- attempts: Số lần thử để chọn ra kết quả tốt nhất.
- flags: Phương pháp chọn các điểm trung tâm ban đầu, trong trường hợp này là cv2.KMEANS PP CENTERS.

Bước 3: Sau khi áp dụng KMeans để phân cụm các điểm ảnh trong không gian màu RGB, thì hàm thực hiện gán Màu Cụm cho Từng Pixel:

- Nhãn của từng pixel (label) được sử dụng để gán màu tương ứng từ mảng color.
- Mảng màu được làm phẳng (label.flatten()) để tương ứng với số lượng pixel trong ảnh.

Hình 9 minh họa ứng dụng của Hàm KMeans Clustering



Hình 9: Ứng dụng của Hàm KMeans Clustering

6 GUI chương trình

6.1 Giới thiệu

Chương trình DSP Segmentation của nhóm em sử dụng thư viện Tkinter và Matplotlib để tạo ra một giao diện người dùng đơn giản (GUI) cho xử lý ảnh. Dưới đây là các chức năng mà GUI nhóm em có:

Nút "Select Image":

- Nút này kích hoạt hàm *select image* khi được nhấp. Hàm này sử dụng filedialog để mở hộp thoại chọn file và lưu đường dẫn của ảnh được chọn.

Nút "Execute":

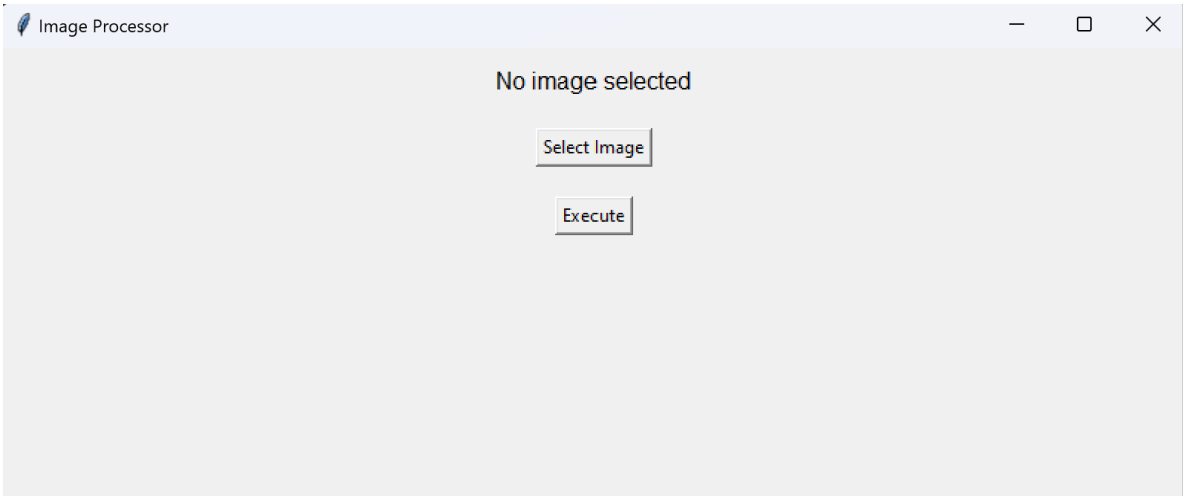
- Nút này kích hoạt hàm *execute processing* khi được nhấp. Hàm này đọc ảnh được chọn, gọi các hàm xử lý từ module functions (các hàm xử lý phân đoạn ảnh) và sau đó hiển thị kết quả lên GUI.

Hàm *display results*

- Hàm này tạo ra và hiển thị một Matplotlib figure chứa các hình ảnh kết quả sau khi xử lý. Mỗi hình ảnh được hiển thị trong một subplot khác nhau.

6.2 Giao diện

Hình 10 là giao diện của chương trình tạo Phân đoạn ảnh



Hình 10: Giao diện chương trình

6.3 Chạy chương trình

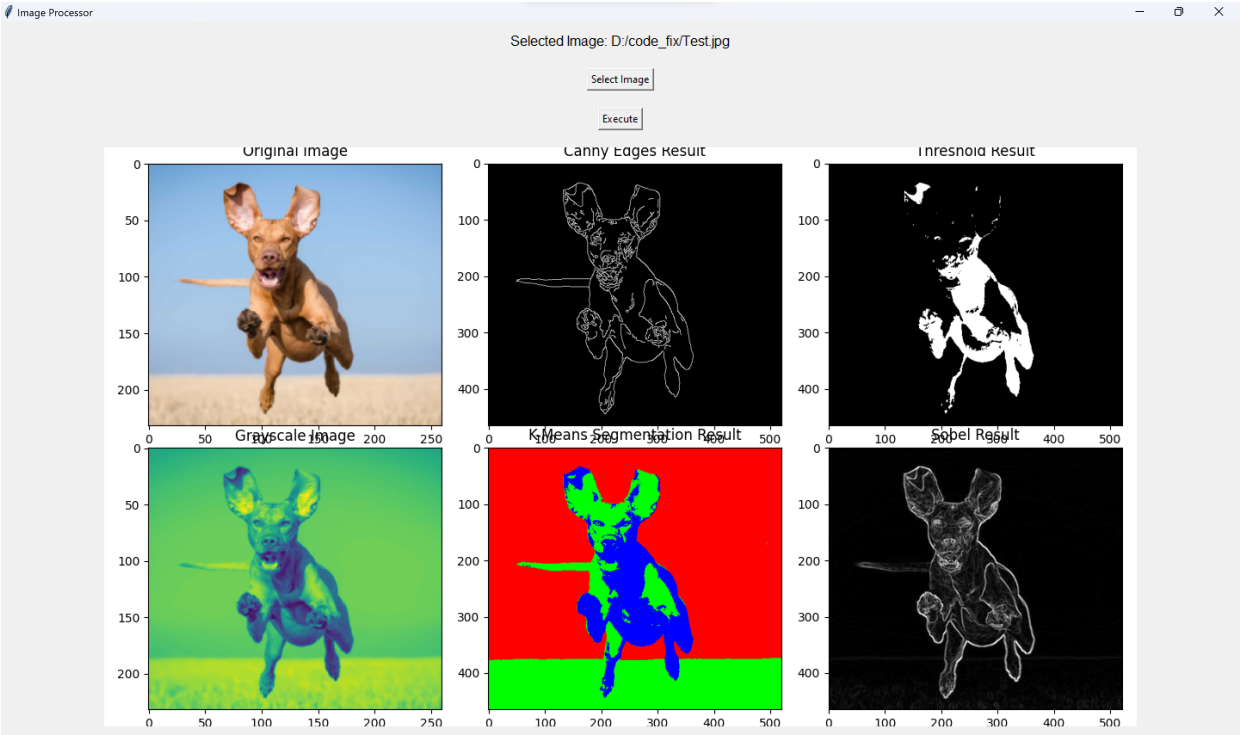
Nhóm sẽ thử với hình ảnh dưới
Hình 11 là hình ảnh nhóm chọn để test

Hình 11: Ảnh test

6.3.1 Các bước chạy chương trình và kết quả

Thực hiện các bước sau để chạy chương trình

Bước 1: Chọn nút "Select Image"
Bước 2: Chọn 1 ảnh bất kì để thực hiện Phân đoạn ảnh
Bước 3: Chọn nút "Execute"
Bước 4: Màn hình hiện ra kết qảu chạy của các phương pháp Phân đoạn ảnh
Hình 12 là kết quả chạy hình test minh họa



Hình 12: Kết quả chạy chương trình

6.3.2 Đánh giá kết quả

Sau khi chạy chương trình, dựa vào kết quả thu được, nhóm có một số đánh giá như sau

- *Về độ chính xác của phân đoạn*: Độ chính xác là một yếu tố quan trọng để đánh giá chất lượng phân đoạn ảnh. Trong trường hợp này, Canny Edge và Sobel thường cho kết quả phân đoạn chính xác hơn so với Threshold và K-Means. Canny Edge và Sobel có khả năng phát hiện các đường biên rõ ràng và có thể tạo ra các vùng phân loại chính xác hơn dựa trên tính chất gradient và độ dốc của ảnh.

- *Về độ mịn và chi tiết của phân đoạn:* Khi xét đến độ mịn và chi tiết trong kết quả phân đoạn, Canny Edge thường tạo ra các đường biên mịn và chi tiết hơn so với các phương pháp khác. Điều này là do Canny Edge sử dụng các bước như làm mờ và ngưỡng hysteresis để tìm ra các đường biên chính xác và mịn.

- *Về tính ổn định và độ tin cậy:* Khi xử lý các ảnh có nhiễu hoặc các vùng có độ tương phản thấp, Canny Edge và Sobel thường cho kết quả tốt hơn so với Threshold và K-Means. Canny Edge và Sobel có khả năng xử lý nhiễu tốt hơn và đáp ứng tốt hơn với các biến đổi độ tương phản trong ảnh.

Như vậy, Canny Edge và Sobel thường cho kết quả phân đoạn tốt hơn so với Threshold và K-Means trong nhiều trường hợp.

Tuy nhiên, việc lựa chọn phương pháp thích hợp phụ thuộc yêu cầu cụ thể của bài toán. Đôi khi, việc kết hợp các phương pháp hoặc tinh chỉnh các thông số của từng phương pháp có thể cải thiện chất lượng phân đoạn ảnh.

6.3.3 Đánh giá ưu nhược điểm của các phương pháp

Từ đánh giá nêu trên, nhóm so sánh những ưu điểm và hạn chế riêng khi sử dụng các phương pháp này:

- *Phương pháp Threshold:*

+ Ưu điểm: Đơn giản, dễ hiểu và tính toán nhanh. Phương pháp này dựa trên việc so sánh giá trị pixel với một ngưỡng (threshold) để phân loại các pixel thành các vùng nhị phân (ví dụ: trắng và đen).

+ Hạn chế: Phương pháp Threshold không đảm bảo kết quả phân đoạn chính xác khi ảnh có nhiễu hoặc các vùng cần phân đoạn có độ tương phản thấp.

- *Phương pháp Canny Edge:*

+ Ưu điểm: Phương pháp Canny Edge tạo ra các đường biên (edge) rõ nét và mịn. Nó dựa trên thuật toán phát hiện biên Canny, kết hợp các bước như làm mờ, tính độ dốc gradient và ngưỡng hysteresis để tìm ra các đường biên chính xác.

+ Hạn chế: Phương pháp này yêu cầu các thông số đầu vào như ngưỡng để thành công. Nếu không chọn các thông số phù hợp, kết quả phân đoạn có thể không tốt. Canny Edge cũng đòi hỏi tính toán phức tạp hơn so với Threshold.

- *Phương pháp K-Means:*

+ Ưu điểm: K-Means là một phương pháp phân đoạn dựa trên việc gom cụm (clustering). Nó có thể được sử dụng để phân đoạn ảnh thành các vùng dựa trên các đặc trưng màu sắc của pixel. K-Means có thể tạo ra các vùng phân loại rõ ràng và có khả năng xử lý ảnh có độ phức tạp cao.

+ Hạn chế: K-Means có thể không hoạt động tốt khi có sự chồng chéo giữa các vùng hoặc khi có nhiễu trong ảnh. Nếu số lượng cụm không được chọn đúng, kết quả phân đoạn cũng có thể không chính xác.

- *Phương pháp Sobel:*

+ Ưu điểm: Sobel là một phương pháp phân đoạn dựa trên việc tính độ dốc gradient của ảnh. Nó tạo ra các biên rõ ràng và có khả năng phát hiện các đặc trưng hình học trong ảnh.

+ Hạn chế: Sobel có thể bị ảnh hưởng bởi nhiễu, vì nó dựa trên tính toán đạo hàm. Nó không cho kết quả phân đoạn nhị phân trực tiếp, mà chỉ tạo ra các giá trị gradient, do đó cần thêm bước xử lý để phân loại các vùng.

Tóm lại, không có một phương pháp phân đoạn ảnh duy nhất nào hoàn hảo cho tất cả các tình huống. Lựa chọn phương pháp phụ thuộc vào loại ảnh cần phân đoạn, yêu cầu cụ thể và ưu tiên của bạn. Thường thì việc kết hợp các phương pháp hoặc tinh chỉnh thông số cho từng phương pháp sẽ đem lại kết quả tốt nhất.

7 Kết luận

Như vậy, qua dự án trên, nhóm đã nghiên cứu và so sánh các phương pháp trong bài toán Phân đoạn ảnh - Image Segmentation. Dựa vào kết quả chạy chương trình, nhóm đã rút ra kết luận: "Phương pháp phân đoạn ảnh Canny Edge và Sobel thường cho kết quả phân đoạn tốt hơn so với Threshold và K-Means trong nhiều trường hợp". Bên cạnh đó, nhóm cũng đã xây dựng được GUI ứng dụng của chương trình. Trong thời gian tới, nhóm sẽ nghiên cứu thêm các phương pháp Phân đoạn ảnh khác; đồng thời sẽ cải tiến thêm giao diện của ứng dụng để dự án thêm hoàn thiện

8 Tài liệu tham khảo

Một số bài nghiên cứu nhóm đã đọc

[1] Haralick R M, Shapiro L G. Image segmentation techniques[J]. Computer vision, graphics, and image processing, 1985, 29(1): 100- 132.

[2] Marr D, Hildreth E. Theory of edge detection[J]. Proceedings of the Royal Society of London B: Biological Sciences, 1980, 207(1167): 187-217.

[3] Wu Z, Leahy R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 1993, 15(11): 1101-1113.

[4] Lin D, Dai J, Jia J, et al. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 3159-3167.

[5] Davis L S, Rosenfeld A, Weszka J S. Region extraction by averaging and thresholding[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1975 (3): 383-388.

[6] Kohler R. A segmentation system based on thresholding[J]. Computer Graphics and Image Processing, 1981, 15(4): 319-338.

[7] Pal N R, Pal S K. A review on image segmentation techniques[J]. Pattern recognition, 1993, 26(9): 1277-1294.

[8] Wani M A, Batchelor B G. Edge-region-based segmentation of range images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994, 16(3): 314-319.

[9] Fan W. Color image segmentation algorithm based on region growth[J]. Jisuanji Gongcheng/ Computer Engineering, 2010, 36(13).

[10] Angelina S, Suresh L P, Veni S H K. Image segmentation based on genetic algorithm for region growth and region merging[C]//Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on. IEEE, 2012: 970-974.

[11] Ugarriza L G, Saber E, Vantaram S R, et al. Automatic image segmentation by dynamic region growth and multiresolution merging[J]. IEEE transactions on image processing, 2009, 18(10): 2275-2288.