TỪ TRANG 219-225

khoảng cách giữa các pixel trong ảnh với và biểu thị vectơ toạ độ của từng pixel. Các hệ số tỷ lệ và xác định tỷ lệ tương đối và tốc độ hội tụ về 0 của mỗi thành phần.

Thử nghiệm thuật toán trên chương trình. Thêm hàm sau vào tệp ncut.py.

|  |
| --- |
|  |

Hàm này lấy một mảng hình ảnh và tạo một vectơ đặc trưng bằng cách sử dụng giá trị RGB hoặc giá trị mức xám tùy thuộc vào hình ảnh đầu vào. Vì trọng số biên chứa một thành phần khoảng cách, sử dụng hàm grid() để lấy các giá trị x và y cho mỗi vectơ đặc trưng pixel. Sau đó, hàm này sẽ lặp trên tất cả N pixel và điền vào các giá trị W trong ma trận cắt chuẩn hóa kích thước NxN.

Chúng ta có thể tính toán phân đoạn bằng cách cắt tuần tự từng vectơ riêng hoặc bằng cách lấy một số vectơ riêng và áp dụng phân cụm. Chúng ta chọn cách thứ 2 và không sửa đổi số lượng phân đoạn. Lấy các hàm vectơ riêng ndim trên cùng của ma trận Laplacian tương ứng với W và phân cụm pixel. Hàm sau thực hiện phân cụm, như có thể thấy nó gần như giống như ví dụ phân cụm phổ trong Mục 6.3.

|  |
| --- |
|  |

Ở đây, đã sử dụng thuật toán phân cụm K-means (xem Phần 6.1 để biết chi tiết) để nhóm các pixel dựa trên các giá trị vectơ giêngtrong ảnh. Chúng ta có thể thử bất kỳ thuật toán phân cụm hoặc tiêu chí phân nhóm nếu muốn thử nghiệm kết quả.

Bây giờ thử điều này trên một số hình ảnh mẫu. Chương trình sau đây cho thấy một ví dụ hoàn chỉnh:

|  |
| --- |
|  |

Ví dụ này thay đổi kích thước hình ảnh thành một kích thước cố định (50x50 trong ví dụ này), để thực hiện tính toán vectơ riêng nhanh hơn. Hàm linalg.svd() trong thư viện NumPy không đủ nhanh để xử lý các ma trận lớn (và đôi khi cho kết quả không chính xác ma trận lớn). Sử dụng phép nội suy song tuyến (bilincar interpolation) khi thay đổi kích thước hình ảnh trong khi sử dụng phương pháp nội suy lân cận gần nhất (nearestneighbor interpolation) khi thay đổi kích thướckết quả hình ảnh phân đoạn bởi vì chúng ta không muốn nội suy các nhãn lớp. Lưu ý chúng ta trước tiên định dạng lại mảng một chiều thành kích thước (*wid, wid*) sau đó thay đổi về kích thước ảnh gốc.

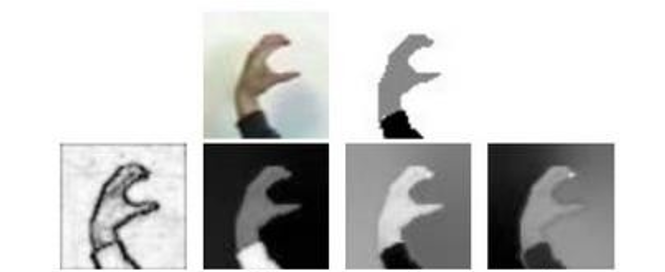
Trong ví dụ này, đã sử dụng một trong các hình ảnh cử chỉ tay từ *Static HandPosture Database* (xem Phần 8.1 để biết thêm chi tiết) với k = 3. Phân đoạn kết quả được hiển thị trong Hình 9.5 cùng với bốn vecto riêng đầu tiên.

Các vectơ riêng được trả về trong mảng V trong ví dụ nầy và có thể được hiển thị như hình ảnh:

|  |
| --- |
|  |

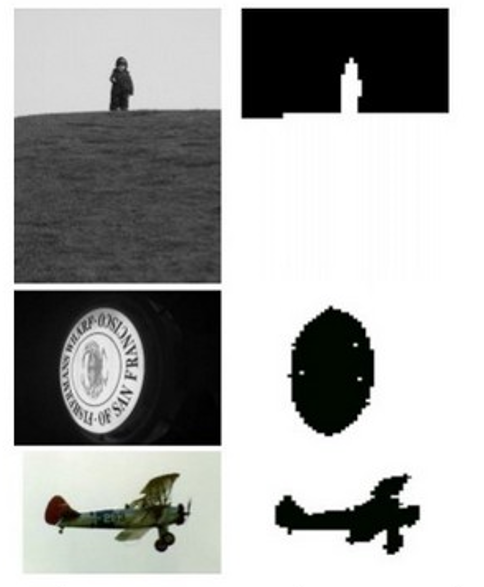
Đoạn chương trình này sẽ hiển thị vectơ riêng i dưới dạng hình ảnh ở kích thước hình ảnh gốc.

Hình 9.6 cho thấy một số ví dụ khác sử dụng cùng một chương trình ở trên. Hình ảnh máy bay được lấy từ loại "airplane” trong bộ dữ liệu Caltech 101. Đối với những ví dụ này giữ các tham số và , cho cùng các giá trị như trên. Thay đổi giá trị các tham số và có thể cho ra kết quả đúng hơn và hình ảnh vectơ riêng khá khác nhau.



*Hình 9.5: Phân đoạn ảnh bằng thuật toán cắt chuẩn hóa. Ảnh trên cùng là hình ảnh gốc và phân đoạn thành ba lớp. Ảnh dưới cùng là bốn vectơ riêng đầu tiên*

Điều đáng chú ý là ngay cả đối với những ví dụ khá đơn giản này, một ngưỡng sẽ không cho kết quả giống nhauvới bài toán phân cụm RGB hoặc giá trị mức xám. Điều này là do chúng ta không xem xét các vùng lân cận của pixel.



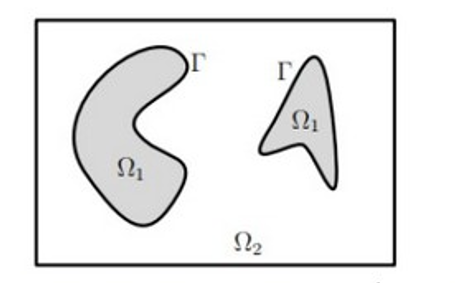
*Hình 9.6: Ví dụ về phân đoạn ảnh hai lớp bằng thuật toán cắt chuẩn hóa. Ảnh trái là ảnh gốc. Ảnh phải là kết quả phân đoạn*

**9.3. PHƯƠNG PHÁP BIẾN PHÂN**

Trong cuốn sách này, đã thấy một số ví dụ về cực tiểu hàm chi phí hoặc năng lượng để giải quyết các vấn đề về thị thị giác máy tính. Trong các phần trước, chúng ta đã cực tiểu lát cắt trong đồ thị nhưng cũng đã thấy các ví dụ về vấn đề khử nhiễu ROF, K-means và SVM. Đây là những ví dụ về bài toán tối ưu.

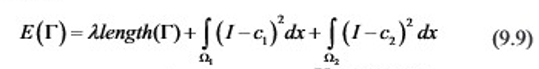
Khi tối ưu hóa được thực hiện trên các hàm, vấn đề khi đó được gọi là vấn đề biển phân và thuật toán để giải quyết các vấn đề như vậy được gọi là các phương pháp biến phân. Chúng ta sẽ phân tích một mô hình biển phân đơn giản và hiệu quả sau đây.

Mô hình phân đoạn Chan-Vese [6] giả định mô hình ảnh không đổi theo từng cặp cho các vùng ảnh được phân đoạn. Ở đây sẽ tập trung vào trường hợp của hai vùng, ví dụ tiền cảnh và hậu cảnh, nhưng mỗ hình mở rộng ra nhiều vùng, xem ví dụ [38]. Mô hình có thể được mô tả như sau.



*Hình 9.7: Mô hình phân đoạn Chan-Vese không đổi theo từng cặp*

Nếu lấy một tập các đường cong Ꞅ tách hình ảnh thành hai vùng và như trong Hình 9.7, kết quả phân đoạn đạt được bằng cách cực tiểu năng lượng trong mô hình Chan Vese. Thuật toán này thực hiện đo độ lệch từ hằng số mức sám trong mỗi vùng và .Ở đây các tích phân được lấy qua từng vùng và chiều dài của các đường cong phân cách tác động đến độ mượt phân đoạn.



Với hình ảnh không đổi theo từng cặp , hàm năng lượng có thể viết lại thành:



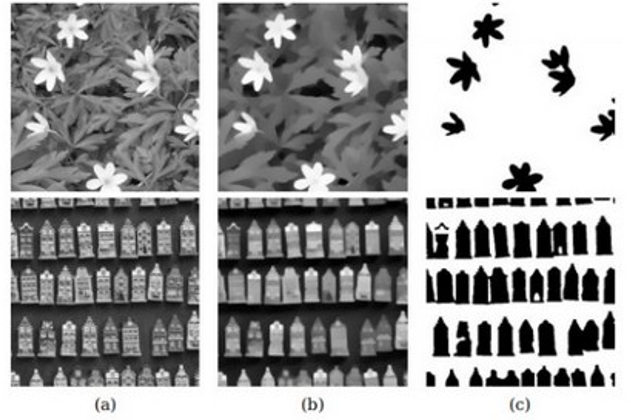
trong đó và là các hàm đặc trưng (chỉ số) cho hai vùng.

Phương trình này giống với phương trình ROF (1.1) với λ được thay thế bằng

λ |- |. Sự khác biệt duy nhất là trong trường hợp Chan-Vese là chúng ta đang tìm kiếm một hình ảnh U bất biến theo cặp. Có thể thấy rằng lấy ngưỡng trên kết quả ROF sẽ được kết quả cực tiểu khá tốt. Có thể tham khảo thêm ở tài liệu [8]. Vấn đề cực tiểu mô hình Chan-Vese bây giờ trở thành vấn đề giảm nhiễu ROF và sau đó lấy ngưỡng.

|  |
| --- |
|  |

Trong trường hợp này, chúng ta giảm khoảng ngưỡng để dùng số lần lặp ROF và à để chắc chắn rằng: số lần lặp là đủ. Hình 9.8 cho thấy kết quả phân đoạn trong trong hai hình ảnh khá khó.



*Hình 9.8: Ví dụ phân đoạn hình ảnh bằng cách cực tiểu mô hình Chan- Vese bằng cách sử dụng khử nhiễu ROF. (a) ảnh gốc, (b) ảnh sau khi khử nhiễu ROF, (c) Ảnh phân đoạn cuối cùng*

**9.4. BÀI TẬP**

1. Có thể tăng tốc tính toán cho phương pháp cắt đồ thị bằng cách giảm số cạnh. Cấu trúc đồ thị này được mô tả trong Phần 4.2 của [16]. Hãy thử điều này và đo kích thước đồ thị khác nhau và thời gian phân đoạn so với cấu trúc đơn giản đã sử dụng trong phần lý thuyết trên.

2. Tạo giao diện người dùng hoặc mô phỏng việc chọn vùng của người dùng để phân đoạn bằng phương pháp cắt đồ thị. Sau đó lập trình hậu cảnh và tiền cảnh bằng cách đặt trọng số có giá trị lớn.

3. Thay đổi vectơ đặc trưng trong phân đoạn bằng cắt đồ thị từ vectơ RGB sang một số mô tả khác. Bạn có thể cải thiện kết quả phân đoạn?

4. Thực hiện cách tiếp cận phân đoạn lặp bằng cách sử dụng phương pháp cắt đồ thị xem có cải thiện chất lượng phân đoạn hay không?

5. Bộ dữ liệu Microsoft Research Grab Cụt chứa các bản đồ phân đoạn thực (ground truth). Thực hiện hàm đo sai số phân đoạn và đánh giá các thông số cài đặt khác nhau?

6. Hãy thử thay đổi các trọng số cạnh trong phương pháp cắt chuẩn hóa và xem chúng ảnh hưởng như thế nào đến vectơ riêng và kết quả phân đoạn.

7. Tính gradient trên các vectơ riêng đầu tiên của phương pháp cắt chuẩn hóa. Kết hợp các ảnh gradient này để phát hiện các đường bao của các đối tượng trong ảnh.

8. Thực hiện tìm kiếm tuyến tính trên giá trị ngưỡng cho hình ảnh khử nhiễu trong phân đoạn Chan-Vese. Đối với mỗi ngưỡng, lưu trữ hàm năng lượng E (T) và chọn phân đoạn có giá trị thấp nhất.