ĐẠI HỌC THÁI NGUYÊN
KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH MÔN HỌC

XỬ LÝ ẢNH

Người soạn : PGS. TS. ĐỖ NĂNG TOÀN,
TS. PHẠM VIỆT BÌNH

Thái Nguyên, Tháng 11 năm 2007

LỜI NÓI ĐẦU

Khoảng hơn mười năm trở lại đây, phần cứng máy tính và các thiết bị liên quan đã có sự tiến bộ vượt bậc về tốc độ tính toán, dung lượng chứa, khả năng xử lý v.v.. và giá cả đã giảm đến mức máy tính và các thiết bị liên quan đến xử lý ảnh đã không còn là thiết bị chuyên dụng nữa. Khái niệm ảnh số đã trở nên thông dụng với hầu hết mọi người trong xã hội và việc thu nhận ảnh số bằng các thiết bị cá nhân hay chuyên dụng cùng với việc đưa vào máy tính xử lý đã trở nên đơn giản.

Trong hoàn cảnh đó, xử lý ảnh là một lĩnh vực đang được quan tâm và đã trở thành môn học chuyên ngành của sinh viên ngành công nghệ thông tin trong nhiều trường đại học trên cả nước. Tuy nhiên, tài liệu giáo trình còn là một điều khó khăn. Hiện tại chỉ có một số ít tài liệu bằng tiếng Anh hoặc tiếng Pháp, tài liệu bằng tiếng Việt thì rất hiếm. Với mong muốn đóng góp vào sự nghiệp đào tạo và nghiên cứu trong lĩnh vực này, chúng tôi biên soạn cuốn giáo trình ***Xử lý ảnh*** dựa trên đề cương môn học đã được duyệt. Cuốn sách tập trung vào các vấn đề cơ bản của xử lý ảnh nhằm cung cấp một nền tảng kiến thức đầy đủ và chọn lọc nhằm giúp người đọc có thể tự tìm hiểu và xây dựng các chương trình ứng dụng liên quan đến xử lý ảnh.

Giáo trình được chia làm 5 chương và phần phụ lục: Chương 1, trình bày Tổng quan về xử lý ảnh, các khai niệm cơ bản, sơ đồ tổng quát của một hệ thống xử lý ảnh và các vấn đề cơ bản trong xử lý ảnh. Chương 2, trình bày các kỹ thuật nâng cao chất lượng ảnh dựa vào các thao tác với điểm ảnh, nâng cao chất lượng ảnh thông qua việc xử lý các điểm ảnh trong lân cận điểm ảnh đang xét. Chương này cũng trình bày các kỹ thuật nâng cao chất lượng ảnh nhờ vào các phép toán hình thái. Chương 3, trình bày các kỹ thuật cơ bản trong việc phát hiện biên của các đối tượng ảnh theo cả hai khuynh hướng: Phát hiện biên trực tiếp và phát hiện biên gián tiếp. Chương 4 thể hiện cách kỹ thuật tìm xương theo khuynh hướng tính toán trục trung vị và hướng tiếp cận xấp xỉ nhờ các thuật toán làm mảnh song song và gián tiếp. Và cuối cùng là Chương 5 với các kỹ thuật hậu xử lý.

Giáo trình được biên soạn dựa trên kinh nghiệm giảng dạy của tác giả trong nhiều năm tại các khóa đại học và cao học của ĐH Công nghệ - ĐHQG Hà Nội, ĐH Khoa học tự nhiên – ĐHQG Hà Nội, Khoa Công nghệ thông tin – ĐH Thái Nguyên v.v.. Cuốn sách có thể làm tài liệu tham khảo cho sinh viên các hệ kỹ sư, cử nhân và các bạn quan tâm đến vấn đề nhận dạng và xử lý ảnh.

Các tác giả bày tỏ lòng biết ơn chân thành tới các bạn đồng nghiệp trong Phòng Nhận dạng và công nghệ tri thức, Viện Công nghệ thông tin, Bộ môn Hệ thống thông tin, Khoa Công nghệ thông tin, ĐH Thái Nguyên, Khoa Công nghệ thông tin, ĐH Công nghệ, ĐHQG Hà Nội, Khoa Toán – Cơ – Tin, ĐH Khoa học tự nhiên, ĐHQG Hà Nội đã động viên, góp ý và giúp đỡ để hoàn chỉnh nội dung cuốn sách này. Xin cám ơn Lãnh đạo Khoa Công nghệ thông tin, ĐH Thái Nguyên, Ban Giám đốc ĐH Thái Nguyên đã hỗ trợ và tạo điều kiện để cho ra đời giáo trình này.

Mặc dù rất cố gắng nhưng tài liệu này chắc chắn không tránh khỏi những sai sót. Chúng tôi xin trân trọng tiếp thu tất cả những ý kiến đóng góp của bạn đọc cũng như các bạn đồng nghiệp để có chỉnh lý kịp thời.

Thư góp ý xin gửi về: Phạm Việt Bình,
Khoa Công nghệ thông tin – ĐH Thái nguyên.
Xã Quyết Thắng, Tp. Thái Nguyên

Điện thoại: 0280.846506 Email: pvbinh@ictu.edu.vn

Thái Nguyên, ngày 22 tháng 11 năm 2007

CÁC TÁC GIẢ

MỤC LỤC

LỜI NÓI ĐẦU 2

MỤC LỤC 4

Chương 1: TỔNG QUAN VỀ XỬ LÝ ẢNH 9

1.1. XỬ LÝ ẢNH, CÁC VẤN ĐỀ CƠ BẢN TRONG XỬ LÝ ẢNH 9

1.1.1. Xử lý ảnh là gì? 9

1.1.2. Các vấn đề cơ bản trong xử lý ảnh 10

1.1.2.1. Một số khái niệm cơ bản 10

1.1.2.2. Nắn chỉnh biến dạng 10

1.1.2.3. Khử nhiễu 11

1.1.2.4. Chỉnh mức xám 11

1.1.2.5. Phân tích ảnh 11

1.1.2.6. Nhận dạng 12

1.1.2.7. Nén ảnh 13

1.2. THU NHẬN VÀ BIỂU DIỄN ẢNH 14

1.2.1. Màu sắc 14

1.2.1.1. Mô hình màu RGB (Red, Green, Bule) 14

1.2.1.2. Mô hình màu CMY (Cyan, Magenta, Yellow) 15

1.2.1.3. Mô hình màu HSV (Hue, Saturation, Value) 16

1.2.1.4. Mô hình màu HLS 19

1.2.2. Thu nhận, các thiết bị thu nhận ảnh 22

1.2.2.1. Giai đoạn lấy mẫu 23

1.2.2.2. Lượng tử hóa 24

1.2.3. Biểu diễn ảnh 24

1.2.3.1. Mô hình Raster 24

1.2.3.2. Mô hình Vector 25

Chương 2: CÁC KỸ THUẬT NÂNG CAO CHẤT LƯỢNG ẢNH 26

2.1. CÁC KỸ THUẬT KHÔNG PHỤ THUỘC KHÔNG GIAN 26

2.1.1. Giới thiệu 26

2.1.2. Tăng giảm độ sáng 26

2.1.3. Tách ngưỡng 27
2.1.4. Bó cụm 27
2.1.5. Cân bằng histogram 28
2.1.6. Kỹ thuật tìm tách ngưỡng tự động 29
2.1.7. Biến đổi cấp xám tổng thể 30
2.2. CÁC KỸ THUẬT PHỤ THUỘC KHÔNG GIAN 31
2.2.1. Phép nhân chập và mẫu 31
2.2.2. Một số mẫu thông dụng 33
2.2.3. Lọc trung vị 34
2.2.4. Lọc trung bình 36
2.2.5. Lọc trung bình theo k giá trị gần nhất 37
2.3. CÁC PHÉP TOÁN HÌNH THÁI HỌC 38
2.3.1. Các phép toán hình thái cơ bản 38
2.3.2. Một số tính chất của phép toán hình thái 39
Chương 3: BIÊN VÀ CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN 44
3.1. GIỚI THIỆU 44
3.2. CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN TRỰC TIẾP 44
3.2.1. Kỹ thuật phát hiện biên Gradient 44
3.2.1.1. Kỹ thuật Prewitt 46
3.2.1.2. Kỹ thuật Sobel 47
3.2.1.3. Kỹ thuật la bàn 47
3.2.2. Kỹ thuật phát hiện biên Laplace 48
3.2.3. Kỹ thuật Canny 49
3.3. PHÁT HIỆN BIÊN GIÁN TIẾP 50
3.3.1 Một số khái niệm cơ bản 50
3.3.2. Chu tuyến của một đối tượng ảnh 51
3.3.3. Thuật toán dò biên tổng quát 53
3.4. PHÁT HIỆN BIÊN DỰA VÀO TRUNG BÌNH CỤC BỘ 56
3.4.1. Biên và độ biến đổi về mức xám 56
3.4.2. Phát hiện biên dựa vào trung bình cục bộ 57
3.5. PHÁT HIỆN BIÊN DỰA VÀO CÁC PHÉP TOÁN HÌNH THÁI 60

3.5.1. Xấp xỉ trên và xấp xỉ dưới đối tượng ảnh 60
3.5.1. Thuật toán phát hiện biên dựa vào phép toán hình thái 61
Chương 4: XƯƠNG VÀ CÁC KỸ THUẬT TÌM XƯƠNG 63
4.1. GIỚI THIỆU 63
4.2. TÌM XƯƠNG DỰA TRÊN LÀM MẢNH 63
4.2.1. Sơ lược về thuật toán làm mảnh 63
4.2.2. Một số thuật toán làm mảnh 65
4.3. TÌM XƯƠNG KHÔNG DỰA TRÊN LÀM MẢNH 65
4.3.1. Khái quát về lược đồ Voronoi 66
4.3.2. Trục trung vị Voronoi rời rạc 66
4.3.3. Xương Voronoi rời rạc 67
4.3.4. Thuật toán tìm xương 68
Chương 5: CÁC KỸ THUẬT HẬU XỬ LÝ 71
5.1. RÚT GỌN SỐ LƯỢNG ĐIỂM BIỂU DIỄN 71
5.1.1. Giới thiệu 71
5.1.2. Thuật toán Douglas Peucker 71
5.1.2.1. Ý tưởng 71
5.1.2.2. Chương trình 72
5.1.3. Thuật toán Band width 73
5.1.3.1. Ý tưởng 73
5.1.3.2. Chương trình 75
5.1.4. Thuật toán Angles 76
5.1.4.1. Ý tưởng 76
5.1.4.2. Chương trình 76
5.2. XẤP XỈ ĐA GIÁC BỞI CÁC HÌNH CƠ SỞ 77
5.2.1 Xấp xỉ đa giác theo bất biến đồng dạng 78
5.2.1.1. Xấp xỉ đa giác bằng đường tròn 80
5.2.1.2. Xấp xỉ đa giác bằng ellipse 80
5.2.1.3. Xấp xỉ đa giác bởi hình chữ nhật 80
5.2.1.4. Xấp xỉ đa giác bởi đa giác đều n cạnh 81
5.2.2 Xấp xỉ đa giác theo bất biến aphin 81
5.3. BIẾN ĐỔI HOUGH 82

5.3.1. Biến đổi Hongh cho đường thẳng 82
5.3.2. Biến đổi Hough cho đường thẳng trong tọa độ cực 84
Chương 6: ỨNG DỤNG XỬ LÝ ẢNH 85
6.1. PHÁT HIỆN GÓC NGHIÊNG VĂN BẢN DỰA VÀO CHU TUYẾN 85
6.1.1. Tính toán kích thước chủ đạo của các đối tượng ảnh 85
6.1.2. Biến đổi Hough và phát hiện góc nghiêng văn bản 87
6.1.2.1. Áp dụng biến đổi Hough trong phát hiện góc nghiêng văn bản 87
6.1.2.2. Thuật toán phát hiện và hiệu chỉnh góc nghiêng văn bản 88
6.1.2.3. Thực nghiệm và kết quả 91
6.2. PHÂN TÍCH TRANG TÀI LIỆU 93
6.2.1. Quan hệ Q_θ 93
6.2.2. Phân tích trang văn bản nhờ khoảng cách Hausdorff bởi quan hệ Q_θ 94
6.2.3. Phân tích trang văn bản dựa vào mẫu 96
6.2.3.1. Đánh giá độ lệch cấu trúc văn bản theo mẫu 96
6.2.3.2. Thuật toán phân tích trang văn bản dựa vào mẫu 99
6.3. CẮT CHỮ IN DÍNH DỰA VÀO CHU TUYẾN 101
6.3.1. Đặt vấn đề 101
6.3.2. Một số khái niệm cơ bản 103
6.3.3. Thuật toán cắt chữ in dính dựa vào chu tuyến 104
6.3.3.1. Phân tích bài toán 104
6.3.3.2. Thuật toán CutCHARACTER cắt chữ in dính dựa vào chu tuyến 106
6.4. NHẬN DẠNG CHỮ VIẾT 107
6.5. TÁCH CÁC ĐỐI TƯỢNG HÌNH HỌC TRONG PHIẾU ĐIỀU TRA DẠNG DẤU 108
6.5.1. Giới thiệu 108
6.5.2. Tách các đối tượng nhờ sử dụng chu tuyến 109
6.6. TÁCH BẢNG DỰA TRÊN TẬP CÁC HÌNH CHỮ NHẬT RỜI RẠC 110
6.6.1. Phân tích bài toán 111

6.7. PHÁT HIỆN ĐỐI TƯỢNG CHUYỂN ĐỘNG 113
6.7.1. Phát hiện đối tượng chuyển động dựa theo hướng tiếp cận trừ khung hình liền kề 113
6.7.2. Phát hiện đối tượng chuyển động theo hướng tiếp cận kết hợp 117
6.7.2.1. Trừ ảnh và đánh dấu Iwb 117
6.7.2.2. Lọc nhiễu và phát hiện độ dịch chuyển 118
6.7.2.3. Phát hiện biên ảnh đa cấp xám Igc 118
6.7.2.4. Kết hợp ảnh Igc với Iwb 119
Phụ lục 1: MỘT SỐ ĐỊNH DẠNG TRONG XỬ LÝ ẢNH 121
1. Định dạng ảnh IMG 121
2. Định dạng ảnh PCX 122
3. Định dạng ảnh TIFF 123
4. Định dạng file ảnh BITMAP 125
Phụ lục 2: CÁC BƯỚC THAO TÁC VỚI FILE AVI 127
1. Bước 1: Mở và đóng thư viện 127
2. Bước 2: Mở và đóng file AVI để thao tác: 127
3. Bước 3: Mở dòng dữ liệu để thao tác 128
4. Bước 4: Trường hợp thao tác với dữ liệu hình của phim 128
5. Bước 5: Thao tác với frame 128
Phụ lục 3: MỘT SỐ MODUL CHƯƠNG TRÌNH 129
1. Nhóm đọc, ghi và hiển thị ảnh 129
1.1. Nhóm đọc ảnh 129
1.2. Nhóm ghi ảnh 137
1.3. Nhóm hiển thị ảnh 139
2. Nhóm phát hiện góc nghiêng văn bản 144
TÀI LIỆU THAM KHẢO 157

Chương 1:

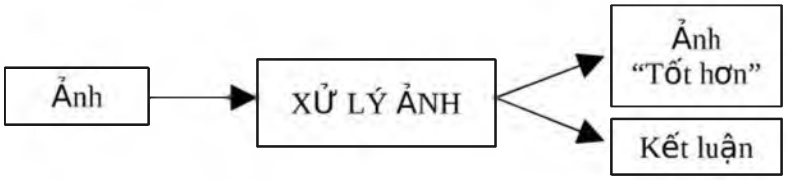
TỔNG QUAN VỀ XỬ LÝ ẢNH

1.1. XỬ LÝ ẢNH, CÁC VẤN ĐỀ CƠ BẢN TRONG XỬ LÝ ẢNH

1.1.1. Xử lý ảnh là gì?

Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tinh, xử lý ảnh và đồ hoạ đó phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ hoạ đóng một vai trò quan trọng trong tương tác người máy.

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh "tốt hơn" hoặc một kết luận.



Hình 1.1. Quá trình xử lý ảnh

Ảnh có thể xem là tập hợp các điểm ảnh và mỗi điểm ảnh được xem như là đặc trưng cường độ sáng hay một dấu hiệu nào đó tại một vị trí nào đó của đối tượng trong không gian và nó có thể xem như một hàm n biến $P(c_1, c_2,..., c_n)$. Do đó, ảnh trong xử lý ảnh có thể xem như ảnh n chiều.

Sơ đồ tổng quát của một hệ thống xử lý ảnh:

Error: Reference source not foundHình 1.2. Các bước cơ bản trong một hệ thống xử lý ảnh

1.1.2. Các vấn đề cơ bản trong xử lý ảnh

1.1.2.1. Một số khái niệm cơ bản

**** Ảnh và điểm ảnh:***

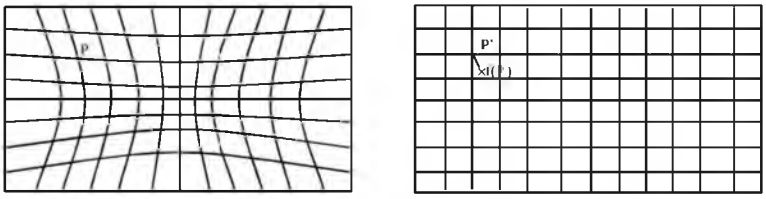
Điểm ảnh được xem như là dấu hiệu hay cường độ sáng tại 1 toạ độ trong không gian của đối tượng và ảnh được xem như là 1 tập hợp các điểm ảnh.

**** Mức xám, màu***

Là số các giá trị có thể có của các điểm ảnh của ảnh

1.1.2.2. Nắn chỉnh biến dạng

Ảnh thu nhận thường bị biến dạng do các thiết bị quang học và điện tử.



Ảnh thu nhận **Ảnh mong muốn**

Hình 1.3. Ảnh thu nhận và ảnh mong muốn

Để khắc phục người ta sử dụng các phép chiếu, các phép chiếu thường được xây dựng trên tập các điểm điều khiển.

Giả sử (P_i, P_i') $i = \overline{1,n}$ có n các tập điều khiển

Tìm hàm f: P_i f (P_i) sao cho

$$\sum_{i=1}^{n} \left| \left| f(P_i) - P_i' \right| \right|^2 \rightarrow \min$$

Giả sử ảnh bị biến đổi chỉ bao gồm: Tịnh tiến, quay, tỷ lệ, biến dạng bậc nhất tuyến tính. Khi đó hàm f có dạng:

$f(x, y) = (a_1x + b_1y + c_1, a_2x + b_2y + c_2)$

Ta có:

$$\phi = \sum_{i=1}^{n} (f(Pi) - Pi')^2 = \sum_{i=1}^{n} \left[\left(a_1x_i + b_1y_i + c_1 - x_i'\right)^2 + \left(a_2x_i + b_2y_i + c_2 - y_i'\right)^2 \right]$$

Để cho $\phi \rightarrow \min$

$$\begin{cases} \dfrac{\partial \phi}{\partial a_1} = 0 \\ \dfrac{\partial \phi}{\partial b_1} = 0 \\ \dfrac{\partial \phi}{\partial c_1} = 0 \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^{n} a_1 x_i^2 + \sum_{i=1}^{n} b_1 x_i y_i + \sum_{i=1}^{n} c_1 x_i = \sum_{i=1}^{n} x_i x_i' \\ \sum_{i=1}^{n} a_1 x_i y_i + \sum_{i=1}^{n} b_1 y_i^2 + \sum_{i=1}^{n} c_1 y_i = \sum_{i=1}^{n} y_i x_i' \\ \sum_{i=1}^{n} a_1 x_i + \sum_{i=1}^{n} b_1 y_i + n c_1 = \sum_{i=1}^{n} x_i' \end{cases}$$

Giải hệ phương trình tuyến tính tìm được a_1, b_1, c_1

Tương tự tìm được a_2, b_2, c_2

$\Rightarrow$ Xác định được hàm f

1.1.2.3. Khử nhiễu

Có 2 loại nhiễu cơ bản trong quá trình thu nhận ảnh

- Nhiều hệ thống: là nhiễu có quy luật có thể khử bằng các phép biến đổi
- Nhiễu ngẫu nhiên: vết bẩn không rõ nguyên nhân → khắc phục bằng các phép lọc

1.1.2.4. Chỉnh mức xám

Nhằm khắc phục tính không đồng đều của hệ thống gây ra. Thông thường có 2 hướng tiếp cận:

- Giảm số mức xám: Thực hiện bằng cách nhóm các mức xám gần nhau thành một bó. Trường hợp chỉ có 2 mức xám thì chính là chuyển về ảnh đen trắng. Ứng dụng: In ảnh màu ra máy in đen trắng.
- Tăng số mức xám: Thực hiện nội suy ra các mức xám trung gian bằng kỹ thuật nội suy. Kỹ thuật này nhằm tăng cường độ mịn cho ảnh

1.1.2.5. Phân tích ảnh

Là khâu quan trọng trong quá trình xử lý ảnh để tiến tới hiểu ảnh. Trong phân tích ảnh việc trích chọn đặc điểm là một bước quan trọng. Các đặc điểm của đối tượng được trích chọn tuỳ theo mục đích nhận dạng trong quá trình xử lý ảnh. Có thể nêu ra một số đặc điểm của ảnh sau đây:

Đặc điểm không gian: Phân bố mức xám, phân bố xác suất, biên độ, điểm uốn v.v..

Đặc điểm biến đổi: Các đặc điểm loại này được trích chọn bằng việc thực hiện lọc vùng (zonal filtering). Các bộ vùng được gọi là "mặt nạ

đặc điểm" (feature mask) thường là các khe hẹp với hình dạng khác nhau (chữ nhật, tam giác, cung tròn v.v..)

Đặc điểm biên và đường biên: Đặc trưng cho đường biên của đối tượng và do vậy rất hữu ích trong việc trích trọn các thuộc tính bất biến được dùng khi nhận dạng đối tượng. Các đặc điểm này có thể được trích chọn nhờ toán tử gradient, toán tử la bàn, toán tử Laplace, toán tử "chéo không" (zero crossing) v.v..

Việc trích chọn hiệu quả các đặc điểm giúp cho việc nhận dạng các đối tượng ảnh chính xác, với tốc độ tính toán cao và dung lượng nhớ lưu trữ giảm xuống.

1.1.2.6. Nhận dạng

Nhận dạng tự động (automatic recognition), mô tả đối tượng, phân loại và phân nhóm các mẫu là những vấn đề quan trọng trong thị giác máy, được ứng dụng trong nhiều ngành khoa học khác nhau. Tuy nhiên, một câu hỏi đặt ra là: mẫu (pattern) là gì? Watanabe, một trong những người đi đầu trong lĩnh vực này đã định nghĩa: "Ngược lại với hỗn loạn (chaos), mẫu là một thực thể (entity), được xác định một cách ang áng (vaguely defined) và có thể gán cho nó một tên gọi nào đó". Ví dụ mẫu có thể là ảnh của vân tay, ảnh của một vật nào đó được chụp, một chữ viết, khuôn mặt người hoặc một ký đồ tín hiệu tiếng nói. Khi biết một mẫu nào đó, để nhận dạng hoặc phân loại mẫu đó có thể:

Hoặc **phân loại có mẫu** (supervised classification), chẳng hạn phân tích phân biệt (discriminant analyis), trong đó mẫu đầu vào được định danh như một thành phần của một lớp đã xác định.

Hoặc **phân loại không có mẫu** (unsupervised classification hay clustering) trong đó các mẫu được gán vào các lớp khác nhau dựa trên một tiêu chuẩn đồng dạng nào đó. Các lớp này cho đến thời điểm phân loại vẫn chưa biết hay chưa được định danh.

Hệ thống nhận dạng tự động bao gồm ba khâu tương ứng với ba giai đoạn chủ yếu sau đây:

1º. Thu nhận dữ liệu và tiền xử lý.

2º. Biểu diễn dữ liệu.

3º. Nhận dạng, ra quyết định.

Bốn cách tiếp cận khác nhau trong lý thuyết nhận dạng là:

1º. Đối sánh mẫu dựa trên các đặc trưng được trích chọn.

2º. Phân loại thống kê.

3º. Đối sánh cấu trúc.

4º. Phân loại dựa trên mạng nơ-ron nhân tạo.

Trong các ứng dụng rõ ràng là không thể chỉ dùng có một cách tiếp cận đơn lẻ để phân loại "tối ưu" do vậy cần sử dụng cùng một lúc nhiều phương pháp và cách tiếp cận khác nhau. Do vậy, các phương thức phân loại tổ hợp hay được sử dụng khi nhận dạng và nay đã có những kết quả có triển vọng dựa trên thiết kế các hệ thống lai (hybrid system) bao gồm nhiều mô hình kết hợp.

Việc giải quyết bài toán nhận dạng trong những ứng dụng mới, nảy sinh trong cuộc sống không chỉ tạo ra những thách thức về thuật giải, mà còn đặt ra những yêu cầu về tốc độ tính toán. Đặc điểm chung của tất cả những ứng dụng đó là những đặc điểm đặc trưng cần thiết thường là nhiều, không thể do chuyên gia đề xuất, mà phải được trích chọn dựa trên các thủ tục phân tích dữ liệu.

1.1.2.7. Nén ảnh

Nhằm giảm thiểu không gian lưu trữ. Thường được tiến hành theo cả hai cách khuynh hướng là nén có bảo toàn và không bảo toàn thông tin. Nén không bảo toàn thì thường có khả năng nén cao hơn nhưng khả năng phục hồi thì kém hơn. Trên cơ sở hai khuynh hướng, có 4 cách tiếp cận cơ bản trong nén ảnh:

- Nén ảnh thống kê: Kỹ thuật nén này dựa vào việc thống kê tần xuất xuất hiện của giá trị các điểm ảnh, trên cơ sở đó mà có chiến lược mã hóa thích hợp. Một ví dụ điển hình cho kỹ thuật mã hóa này là *.TIF
- Nén ảnh không gian: Kỹ thuật này dựa vào vị trí không gian của các điểm ảnh để tiến hành mã hóa. Kỹ thuật lợi dụng sự giống nhau của các điểm ảnh trong các vùng gần nhau. Ví dụ cho kỹ thuật này là mã nén *.PCX
- Nén ảnh sử dụng phép biến đổi: Đây là kỹ thuật tiếp cận theo hướng nén không bảo toàn và do vậy, kỹ thuật thướng nến hiệu quả hơn. *.JPG chính là tiếp cận theo kỹ thuật nén này.
- Nén ảnh Fractal: Sử dụng tính chất Fractal của các đối tượng ảnh, thể hiện sự lặp lại của các chi tiết. Kỹ thuật nén sẽ tính toán để chỉ cần lưu trữ phần gốc ảnh và quy luật sinh ra ảnh theo nguyên lý Fractal

1.2. THU NHẬN VÀ BIỂU DIỄN ẢNH

1.2.1. Màu sắc

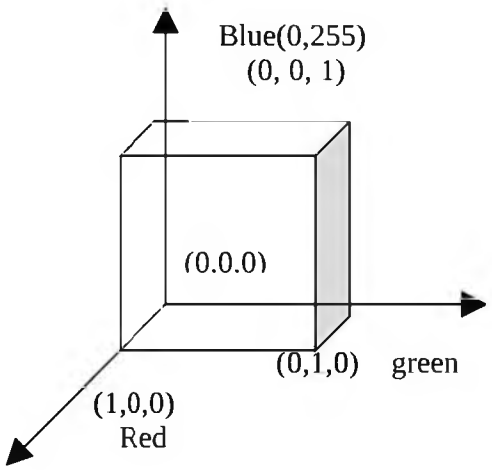
Mắt người có thể phân biệt được vài chục màu nhưng chỉ có thể cảm nhận được hàng ngàn màu. Ba thuộc tính của một màu đó là: Sắc (Hue), Độ thuần khiết (Saturation), và độ sáng hay độ chói (Itensity).

Trong xử lý ảnh và đồ họa, mô hình màu là một chỉ số kỹ thuật của một hệ tọa độ màu 3 chiều với tập các màu nhỏ thành phần có thể trông thấy được trong hệ thống tọa độ màu thuộc một gam màu đặc trưng. Ví dụ như mô hình màu RGB (Red, Green, Blue): là một đơn vị tập các màu thành phần sắp xếp theo hình lập phương của hệ trục tọa độ Đề các.

Mục đích của mô hình màu là cho phép các chỉ số kỹ thuật quy ước của một số loại màu sắc thích hợp với các màu sắc của một số gam màu khác. Chúng ta có thể nhìn thấy trong mô hình màu này, không gian màu là một tập hợp nhỏ hơn của không gian các màu có thể nhìn thấy được, vì vậy một mô hình màu không thể được sử dụng để định rõ tất cả có thể nhìn thấy. Sau đây, ta xem xét một số mô hình hay được sử dụng nhất.

1.2.1.1. Mô hình màu RGB (Red, Green, Bule)

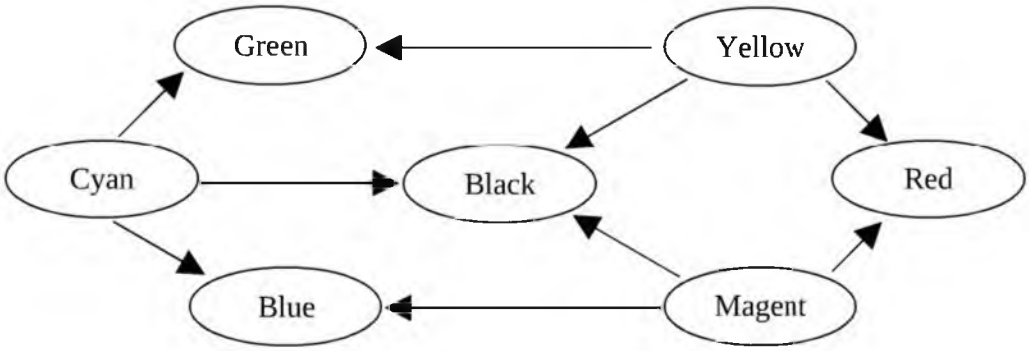
Màu đỏ, lục – xanh lá cây, lam – xanh da trời (RGB) được sử dụng phổ biến nhất. Những màu gốc RGB được thêm vào những màu gốc khác điều đó tạo nên sự đóng góp riêng của từng màu gốc được thêm cùng nhau để mang lại kết qaủ. Tập hợp màu nhỏ thành phần sắp xếp theo khối lập phương đơn vị. Đường chéo chính của khối lập phương với sự cân bằng về số lượng từng màu gốc tương ứng với các mức độ xám với đen là (0,0,0) và trắng (1,1,1).



Hình 1.4. **Mô hình màu RGB**

1.2.1.2. Mô hình màu CMY (Cyan, Magenta, Yellow)

Là phần bù tương ứng cho các màu đỏ, lục, lam và cúng được sử dụng như những bộ lọc loại trừ các màu này từ ánh sáng trắng. Vì vậy CMY còn được gọi là các phần bù loại trừ của màu gốc. Tập hợp màu thành phần biểu diễn trong hệ tọa độ Đề-các cho mô hình mầu CMY cũng giống như cho mô hình màu RGB ngoại trừ màu trắng (ánh sáng trắng), được thay thế màu đen (không có ánh sáng) ở tại nguồn sáng. Các màu thường được tạo thành bằng cách loại bỏ hoặc được bù từ ánh sáng trắng hơn là được thêm vào những màu tối.



***Hình 1.5*. Các màu gốc bù và sự pha trộn giữa chúng**

Khi bề mặt được bao phủ bởi lớp mực màu xanh tím, sẽ không có tia màu đỏ phản chiếu từ bề mặt đó. Màu xanh tím đã loại bỏ phần màu đỏ phản xạ khi có tia sáng trắng, mà bản chất là tổng của 3 màu đỏ, lục, lam. Vì thế ta có thể coi màu Cyan là màu trắng trừ đi màu đỏ và đó cũng là màu lam cộng màu lục. Tương tự như vậy ta có màu đỏ thẫm (magenta) hấp thụ màu lục, vì thế nó tương đương với màu đỏ cộng màu lam. Và cuối cùng màu vàng (yellow) hấp thụ màu lam, nó sẽ bằng màu đỏ cộng với lục.

Khi bề mặt của thực thể được bao phủ bởi xanh tím và vàng, chúng sẽ hấp thụ hết các phần màu đỏ và xanh lam của bề mặt. Khi đó chỉ tồn tại duy nhất màu lục bị phản xạ từ sự chiếu sáng của ánh sáng trắng. Trong trường hợp khi bề mặt được bao phủ bởi cả 3 màu xanh tím, vàng, đỏ thẫm, hiện tượng hấp thụ xảy ra trên cả 3 màu đỏ, lục và lam. Do đó, màu đen sẽ màu của bề mặt. Những mối liên hệ này có thể được miêu tả bởi:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

***Hình 1.6*. Sự biến đổi từ RGB thành CMY**

1.2.1.3. Mô hình màu HSV (Hue, Saturation, Value)

Các mô hình màu RGB, CMY được định hướng cho phần cứng trái ngược với mô hình màu HSV của Smith hay còn được gọi là mẫu HSB với B là Brightness (độ sáng), được định hướng người sử dụng dựa trên cơ sở nền tảng về trực giác về tông màu, sắc độ và sắc thái mỹ thuật.

Hệ thống tọa độ có dạng hình trụ và tập màu thành phần của không gian bên trong mô hình màu được xác định là hình nón hoặc hình chóp sáu cạnh như trong hình 1.7. Đỉnh hình chóp là sáu cạnh khi V= 1 chứa đựng mối quan hệ giữa các màu sáng và những màu trên mặt phẳng với V= 1 đều có màu sáng.

Hình 1.7. Mô hình màu HSV

Sắc màu (hue) hoặc H được đo bởi góc quanh trục đứng với màu đỏ là 0^o, màu lục là 120^o, màu lam là 240^o (xem hình 1.7). Các màu bổ sung trong hình chóp HSV ở 180^o đối diện với màu khác. Giá trị của S là một tập các giá trị đi từ 0 trên đường trục tâm (trục V) đến 1 trên các mặt bên tại đỉnh của hình chóp sáu cạnh. Sự bão hòa được đo tương đối cho gam màu tương ứng với mô hình màu này.

Mô hình màu dạng hình chóp sáu cạnh này đường cao V với đỉnh là điểm gốc tọa độ (0,0). Điểm ở đỉnh là màu đen có giá trị tọa độ màu V= 0, tại các điểm này giá trị của H và S là không liên quan với nhau. Khi điểm có S= 0 và V= 1 là điểm màu trắng, những giá trị trung gian của V đối với S= 0 (trên đường thẳng qua tâm) là các màu xám. Khi S= 0 giá trị của H phụ thuộc được gọi bởi các quy ước không xác định, ngược lại khi S khác 0 giá trị của H sẽ là phụ thuộc.

Như vậy một màu nào đó V= 1, S= 1 là giống như màu thuần khiết trong mỹ thuật được sử dụng như điểm khởi đầu trong các màu pha trên. Thêm màu trắng phù hợp để giảm S (không có sự thay đổi V) tạo nên sự thay đổi sắc thái của gam màu. Sự chuyển màu được tạo ra bởi việc giữ S= 1 và giảm V tạo nên sự thay đổi ề sắc độ và tông màu tạo thành bởi việc thay đổi cả hai S và V.

Chuyển đổi từ RGB sang HSV

```
Hàm RGB_HSV_Conversion
H: Sắc độ màu [0-360] với màu đỏ tại điểm 0
S: Độ bão hòa [0-1]
V: Giá trị cường độ sáng [0-1]
Max: Hàm lấy giá trị cực đại
Min: Hàm lấy giá trị nhỏ nhất
{
  //Xác định giá trị cường độ sáng
  V= Max(R,G,B)
  //Xác định độ bão hòa
  Temp= Min(R,G,B)
  If V=0 than
    S= 0
  Else
    S= (V-Temp)/V
  End
  //Xác định sắc màu
  IF s=0 THEN
    H= Undefined
  Else
    Cr= (V-R)/(V-Temp);
    Cg= (V-G)/(V-Temp);
    Cb= (V-B)/(V-Temp);
    // Màu nằm trong khoảng giữa vàng (Yellow) và đỏ tía (Magenta)
    If R=V then
      H= Cb-Cg
    // Màu nằm trong khoảng giữa xanh tím (cyan) và vàng (yellow)
    If G= V then
      H= 2+Cr-Cb
    // Màu nằm trong khoảng giữa đỏ tươi (magenta) và xanh (cyan)
    If B=V then
      H= 4+ Cg – Cr
```

```
        H= 60*H // Chuyển sang độ
        //Loại các giá trị âm
        If H < 0 then
            H= H+360
    }
```

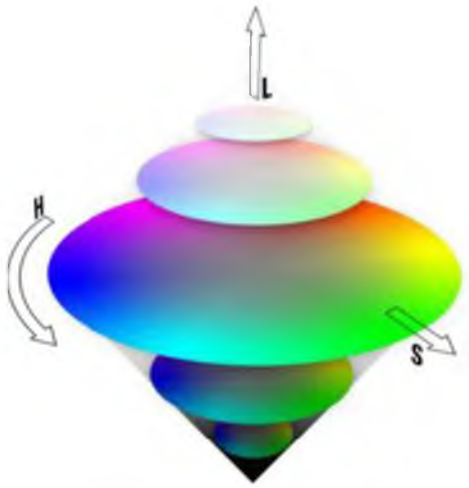
Chuyển đổi từ HSV sang RGB

```
    Hàm HSV_RGB_Conversion()
    H: Sắc độ màu [0-360] với màu đỏ tại điểm 0
    S: Độ bão hòa [0-1]
    V: Giá trị cường độ sáng [0-1]
    {
      //Kiểm tra trường hợp ánh sáng không màu
      If S=0 then
        If H=Undifined then
          R= V
          G= V
          B= V
        Endif
      Else
        If H=360 then
          H= 0
        Else
          H= H/60
      endif
      I= Floor(H)
      F= H-I
      M= V*(1-S)
      N= V*(1-S*F)
      K= V*(1-S*(1-F))
      //(R,G,B)=(V,K,M) ⇔ R= V; C= K; B= M
      If I=0 then
        (R,G,B)=(V,K,M);
```

```
If I=1 then
   (R,G,B)=(N,V,M);
If I=2 then
   (R,G,B)=(M,V,K);
If I=3 then
   (R,G,B)=(M,N,V);
If I=4 then
   (R,G,B)=(K,M,V);
If I=5 then
   (R,G,B)=(V,M,N);
}
```

1.2.1.4. Mô hình màu HLS

Mô hình màu HLS được xác định bởi tập hợp hình chóp sáu cạnh đôi của không gian hình trụ. Sắc màu là góc quanh trục đứng cảu hình chóp sáu cạnh đôi với màu đỏ tại góc 0o. Các màu sẽ xác định theo thứ tự giống như trong biểu đồ CIE khi ranh giới của nó bị xoay ngược chiều kim đồng hồ: Màu đỏ, màu vàng, màu lục, màu xanh tím, màu lam và đỏ thẫm. Điều này cũng giống như thứ tự sắc xếp trong mẫu hình chóp sáu cạnh đơn HSV.



Hình 1.8. **Mô hình màu HLS**

Chúng ta có thể xem mẫu HLS như một sự biến dạng cảu mẫu HSV mà trong đó mãu này màu trắng được kéo hướng lên hình chóp sáu cạnh phía trên từ mặt V= 1. Như với mẫu hình chóp sáu cạnh đơn, phần bổ sung của một màu sắc được đặt ở vị trí 180º hơn là xunh quanh hình chóp sáu cạnh đôi, sự bão hòa được đo xung quanh trục đứng, từ 0 trên trục tới 1 trên bề mặt. Độ sáng bằng không cho màu đen và bằng một cho màu trắng.

Chuyển đổi từ RGB sang HLS

Hàm RGB_HLS_Conversion()

H: Sắc độ màu [0-360] với màu đỏ tại điểm 0

S: Độ bão hòa [0-1]

V: Giá trị cường độ sáng [0-1]

Max: Hàm lấy giá trị cực đại

Min: Hàm lấy giá trị nhỏ nhất

```
{
  //Xác định độ sáng
  M1= Max(R,G,B)
  M2= Min(R,G,B)
  L= (M1+M2)
  //Xác định độ bão hòa
  If M1=M2 //Trường hợp không màu
    S= 0
    H= Undefined
  Else
  If L <= 0.5 then //Trường hợp màu
    S= (M1-M2)/(M1+M2)
  Else
    S= (M1-M2)/(2-M1-M2)
  Endif
  //Xác định sắc độ
  Cr= (M1-R)/(M1-M2)
  Cg= (M1-G)/(M1-M2)
  Cb= (M1-B)/(M1-M2)
  if R=M1 then
    H= Cb-Cg
  If G=M1 then
    H= 2+Cr-Cb
  If B=M1 then
    H= 4+Cg-Cr
```

```
    H= H*60
    if H<0 then
      H= H+360
    endif
  }
```

Chuyển đổi từ HLS sang RGB

```
  Hàm HLS_RGB_Conversion()
  H: Sắc độ màu [0-360] với màu đỏ tại điểm 0
  S: Độ bão hòa [0-1]
  V: Giá trị cường độ sáng [0-1]
  {
    If L <= 0.5 then
      M2= L*(1+S)
    Else
      M2= L+S-L*S
    Endif
    M1= 2*L-M2
    //Kiểm tra độ bão hòa = 0
    If S=0 then
      If H=Undefined
        R=L
        G=L
        B=L
      Else //Error: Dữ liệu nhập sai
      Endif
    Else //Xác định giá trị của RGB
      RGB(H+120, M1,M2,Value)
      R= Value
      RGB(H, M1,M2,Value)
      G= Value
      RGB(H-120, M1,M2,Value)
      B= Value
```

```
    Endif
}

//Hàm điều chỉnh giá trị của H cho phù hợp khoảng xác định
Hàm RGB(H, M1, M2, Value)
{
    If H < 0 then
        H= H+360
    If H < 60 then
        Value= M1+(M2-M1)*H/60
    If H >=60 and H < 180 then
        Value= M2
    If H>= 180 and H < 240 then
        Value = M1+(M2-M1)*(240-H)/60
    If H > 240 and H <= 360 then
        Value= M1
    Return
}
```

1.2.2. Thu nhận, các thiết bị thu nhận ảnh

Các thiết bị thu nhận ảnh bao gồm camera, scanner các thiết bị thu nhận này có thể cho ảnh đen trắng

Các thiết bị thu nhận ảnh có 2 loại chính ứng với 2 loại ảnh thông dụng Raster, Vector.

Các thiết bị thu nhận ảnh thông thường Raster là camera các thiết bị thu nhận ảnh thông thường Vector là sensor hoặc bàn số hoá Digitalizer hoặc được chuyển đổi từ ảnh Raster.

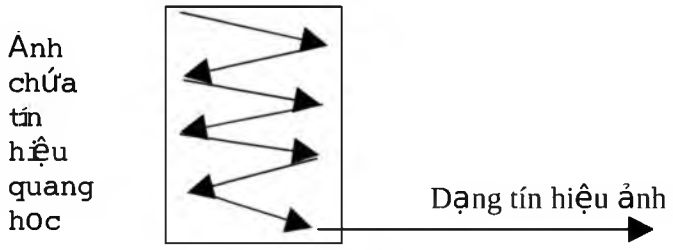
Nhìn chung các hệ thống thu nhận ảnh thực hiện 1 quá trình

- Cảm biến: biến đổi năng lượng quang học thành năng lượng điện (giai đoạn lấy mẫu)
- Tổng hợp năng lượng điện thành ảnh (giai đoạn lượng tử hóa)

1.2.2.1. Giai đoạn lấy mẫu

Người ta sử dụng bộ cảm biến hoặc máy quét để biến tín hiệu quang của ảnh thành tín hiệu điện liên tục. Phương pháp sử dụng máy quét phổ

biến hơn. Máy quét sẽ quét theo chiều ngang để tạo ra tín hiệu điện của ảnh, kết quả cho ra một tín hiệu điện hai chiều f(x,y) liên tục.



Xét ảnh liên tục được biểu diễn bởi hàm f(x, y), gọi Δx là khoảng cách giữa hai điểm được giữ lại theo trục x, gọi Δy là khoảng cách giữa hai điểm được giữ lại theo trục y. Δy, Δx được gọi là chu kỳ lấy mẫu theo trục x và y.

Giai đoạn lấy mẫu sẽ biến hàm liên tục $f(x,y) \rightarrow f(n\Delta x, m\Delta y)$. Với m,n là nguyên.

Theo SHANON để đảm bảo không xảy ra hiện tượng chồng phổ, cho phép tái tạo lại ảnh gốc từ ảnh đã số hóa:

- Gọi $fx = \frac{1}{\Delta x}$ là tần số lấy mẫu theo trục x.

- Gọi $fy = \frac{1}{\Delta y}$ là tần số lấy mẫu theo trục y.

Để không xảy ra hiện tượng chồng phổ thì tần số lấy mẫu phải ít nhất phải lớn hơn hoặc bằng 2 tần số cao nhất của tín hiệu ảnh. Tức là:

$fx >= 2fx_{max}$

$fy >= 2fy_{max}$

Trong đó fx_{max}, fy_{max} là tần số cao nhất của tín hiệu theo trục x, y.

1.2.2.2. Lượng tử hóa

Ảnh sau khi lấy mẫu sẽ có dạng f(m,n) với m, n là nguyên nhưng giá trị f(m, n) vẫn là giá trị vật lý liên tục. Quá trình biến đổi giá trị f(m,n) thành một số nguyên thích hợp để lưu trữ gọi là lượng tử hoá. Đây là quá trình ánh xạ một biến liên tục u vào biến rời rạc u* thuộc tập hữu hạn [u_1, u_2,..u_L] xác định trước, L là mức lượng tử hoá được tạo ra.

Ví dụ:

+ Tạo ảnh đa cấp xám thì L=256, f(m,n) = g ∈ [0, 255]

+ Tạo ảnh 2^{24} thì L=2^{24}, f(m, n) = g ∈ $|0, \quad 2^{24} - 1|$

1.2.3. Biểu diễn ảnh

Ảnh trên máy tính là kết quả thu nhận theo các phương pháp số hoá được nhúng trong các thiết bị kỹ thuật khác nhau. Quá trình lưu trữ ảnh nhằm 2 mục đích:

- Tiết kiệm bộ nhớ
- Giảm thời gian xử lý

Việc lưu trữ thông tin trong bộ nhớ có ảnh hưởng rất lớn đến việc hiển thị, in ấn và xử lý ảnh được xem như là 1 tập hợp các điểm với cùng kích thước nếu sử dụng càng nhiều điểm ảnh thì bức ảnh càng đẹp, càng mịn và càng thể hiện rõ hơn chi tiết của ảnh người ta gọi đặc điểm này là độ
phân giải.

Việc lựa chọn độ phân giải thích hợp tuỳ thuộc vào nhu cầu sử dụng và đặc trưng của mỗi ảnh cụ thể, trên cơ sở đó các ảnh thường được biểu diễn theo 2 mô hình cơ bản

1.2.3.1. Mô hình Raster

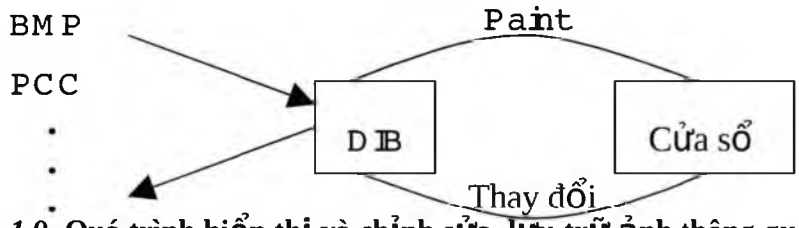
Đây là cách biểu diễn ảnh thông dụng nhất hiện nay, ảnh được biểu diễn dưới dạng ma trận các điểm (điểm ảnh). Thường thu nhận qua các thiết bị như camera, scanner. Tuỳ theo yêu cầu thực thể mà mỗi điểm ảnh được biểu diễn qua 1 hay nhiều bít

Mô hình Raster thuận lợi cho hiển thị và in ấn. Ngày nay công nghệ phần cứng cung cấp những thiết bị thu nhận ảnh Raster phù hợp với tốc độ nhanh và chất lượng cao cho cả đầu vào và đầu ra. Một thuận lợi cho việc hiển thị trong môi trường Windows là Microsoft đưa ra khuôn dạng ảnh DIB (Device Independent Bitmap) làm trung gian. Hình 1.4 thể hình quy trình chung để hiển thị ảnh Raster thông qua DIB.

Một trong những hướng nghiên cứu cơ bản trên mô hình biểu diễn này là kỹ thuật nén ảnh các kỹ thuật nén ảnh lại chia ra theo 2 khuynh hướng là nén bảo toàn và không bảo toàn thông tin nén bảo toàn có khả

năng phục hồi hoàn toàn dữ liệu ban đầu còn nếu không bảo toàn chỉ có khả năng phục hồi độ sai số cho phép nào đó. Theo cách tiếp cận này người ta đã đề ra nhiều quy cách khác nhau như BMP, TIF, GIF, PCX…

Hiện nay trên thế giới có trên 50 khuôn dạng ảnh thông dụng bao gồm cả trong đó các kỹ thuật nén có khả năng phục hồi dữ liệu 100% và nén có khả năng phục hồi với độ sai số nhận được.



Hình 1.9. **Quá trình hiển thị và chỉnh sửa, lưu trữ ảnh thông qua DIB**

1.2.3.2. Mô hình Vector

Biểu diễn ảnh ngoài mục đích tiết kiệm không gian lưu trữ dễ dàng cho hiển thị và in ấn còn đảm bảo dễ dàng trong lựa chọn sao chép di chuyển tìm kiếm… Theo những yêu cầu này kỹ thuật biểu diễn vector tỏ ra ưu việt hơn.

Trong mô hình vector người ta sử dụng hướng giữa các vector của điểm ảnh lân cận để mã hoá và tái tạo hình ảnh ban đầu ảnh vector được thu nhận trực tiếp từ các thiết bị số hoá như Digital hoặc được chuyển đổi từ ảnh Raster thông qua các chương trình số hoá

Công nghệ phần cứng cung cấp những thiết bị xử lý với tốc độ nhanh và chất lượng cho cả đầu vào và ra nhưng lại chỉ hỗ trợ cho ảnh Raster.

Do vậy, những nghiên cứu về biểu diễn vectơ đều tập trung từ chuyển đổi từ ảnh Raster.

Hình 1.10. **Sự chuyển đổi giữa các mô hình biểu diễn ảnh**

Chương 2:

CÁC KỸ THUẬT NÂNG CAO CHẤT LƯỢNG ẢNH

2.1. CÁC KỸ THUẬT KHÔNG PHỤ THUỘC KHÔNG GIAN

2.1.1. Giới thiệu

Các phép toán không phụ thuộc không gian là các phép toán không phục thuộc vị trí của điểm ảnh.

Ví dụ: Phép tăng giảm độ sáng , phép thống kê tần suất, biến đổi tần suất v.v..

Một trong những khái niệm quan trọng trong xử lý ảnh là biểu đồ tần suất (Histogram)

Biểu đồ tần suất của mức xám g của ảnh I là số điểm ảnh có giá trị g của ảnh I. Ký hiệu là h(g)

Ví dụ:

$$I = \begin{pmatrix} 1 & 2 & 0 & 4 \\ 1 & 0 & 0 & 7 \\ 2 & 2 & 1 & 0 \\ 4 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{pmatrix}$$

g	0	1	2	4	7
h(g)	5	7	5	2	1

2.1.2. Tăng giảm độ sáng

Giả sử ta có I ~ kích thước m × n và số nguyên c

Khi đó, kỹ thuật tăng, giảm độc sáng được thể hiện

```
for (i = 0; i < m; i + +)
for (j = 0; j < n; j + +)
        I [i, j] = I [i, j] + c;
```

- Nếu c > 0: ảnh sáng lên
- Nếu c < 0: ảnh tối đi

2.1.3. Tách ngưỡng

Giả sử ta có ảnh I ~ kích thước m × n, hai số Min, Max và ngưỡng θ khi đó: Kỹ thuật tách ngưỡng được thể hiện

```
for (i = 0; i < m; i + +)
for (j = 0; j < n; j + +)
    I [i, j] = I [i, j] > = θ? Max : Min;
```

**** Ứng dụng:***

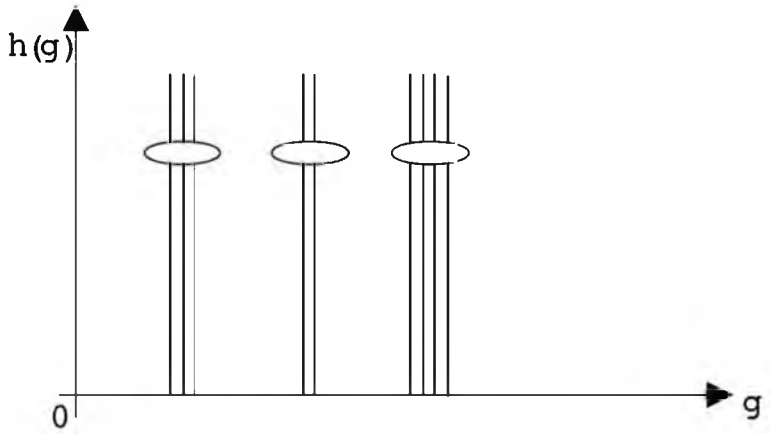
Nếu Min = 0, Max = 1 kỹ thuật chuyển ảnh thành ảnh đen trắng được ứng dụng khi quét và nhận dạng văn bản có thể xảy ra sai sót nền thành ảnh hoặc ảnh thành nền dẫn đến ảnh bị đứt nét hoặc dính.

2.1.4. Bó cụm

Kỹ thuật nhằm giảm bớt số mức xám của ảnh bằng cách nhóm lại số mức xám gần nhau thành 1 nhóm

Nếu chỉ có 2 nhóm thì chính là kỹ thuật tách ngưỡng. Thông thường có nhiều nhóm với kích thước khác nhau.

Để tổng quát khi biến đổi người ta sẽ lấy cùng 1 kích thước bunch_size



I [i,j] = I [i,j]/ bunch - size * bunch_size ∀(i,j)

Ví dụ: Bó cụm ảnh sau với bunch_size= 3

$$I = \begin{pmatrix} 1 & 2 & 4 & 6 & 7 \\ 2 & 1 & 3 & 4 & 5 \\ 7 & 2 & 6 & 9 & 1 \\ 4 & 1 & 2 & 1 & 2 \end{pmatrix}$$

$$I_{kq} = \begin{pmatrix} 0 & 0 & 3 & 6 & 6 \\ 0 & 0 & 3 & 3 & 3 \\ 6 & 0 & 6 & 9 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2.1.5. Cân bằng histogram

Ảnh I được gọi là cân bằng "lý tưởng" nếu với mọi mức xám g, g' ta có h(g) = h(g')

Giả sử, ta có ảnh I ~ kích thước $m \times n$

new_level ~ số mức xám của ảnh cân bằng

$TB = \frac{m \times n}{new\_level}$ ~ số điểm ảnh trung bình của mỗi mức xám của ảnh cân bằng

$t(g) = \sum_{i=0}^{g} h(i)$ ~ số điểm ảnh có mức xám ≤ g

Xác định hàm f: g ¡ f(g)

Sao cho: $f(g) = \max\left\{0, round\left(\frac{t(g)}{TB}\right) - 1\right\}$

Ví dụ: Cân bằng ảnh sau với new_level= 4

$$I = \begin{pmatrix} 1 & 2 & 4 & 6 & 7 \\ 2 & 1 & 3 & 4 & 5 \\ 7 & 2 & 6 & 9 & 1 \\ 4 & 1 & 2 & 1 & 2 \end{pmatrix}$$

g	h(g)	t(g)	f(g)
1	5	5	0
2	5	10	1
3	1	11	1
4	3	14	2
5	1	15	2
6	2	17	2
7	2	19	3
9	1	20	3

$$\begin{pmatrix} 0 & 1 & 2 & 2 & 3 \end{pmatrix}$$

$$I_{kq} = \begin{matrix} 1 & 0 & 1 & 2 & 2 \\ 3 & 1 & 2 & 3 & 0 \\ 2 & 0 & 1 & 0 & 1 \end{matrix}$$

Chú ý: Ảnh sau khi thực hiện cân bằng chưa chắc đã là cân bằng "lý tưởng "

2.1.6. Kỹ thuật tìm tách ngưỡng tự động

Ngưỡng θ trong kỹ thuật tách ngưỡng thường được cho bởi người sử dụng. Kỹ thuật tìm tách ngưỡng tự động nhằm tìm ra ngưỡng θ một cách tự động dựa vào histogram theo nguyên lý trong vật lý là vật thể tách làm 2 phần nếu tổng độ lệnh trong từng phần là tối thiểu.

Giả sử, ta có ảnh I ~ kích thước m × n

G ~ là số mức xám của ảnh kể cả khuyết thiếu

t(g) ~ số điểm ảnh có mức xám ≤ g

$m(g) = \frac{1}{t(g)} \sum_{i=0}^{g} i.h(i)$ ~ môмen quán tính TB có mức xám ≤ g

Hàm f: $g\ g \quad f(g)$

$$f(g) = \frac{t(g)}{mxn - t(g)} [m(g) - m(G-1)]^2$$

Tìm θ sao cho:

$$f(\theta) = \max_{0 \le g < G-1} \{ f(g) \}$$

Ví dụ: Tìm ngưỡng tự động của ảnh sau

$$I = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Lập bảng

g	h(g)	t(g)	g.h(g)	$\sum_{i=0}^{g} ih(i)$	m(g)	f(g)
0	15	15	0	0	0	1.35
1	5	20	5	5	0,25	1.66
2	4	24	8	13	0,54	1.54

3	3	27	9	22	0,81	1.10
4	2	29	8	30	1,03	0.49
5	1	30	5	35	1,16	∞

Ngưỡng cần tách θ= 1 ứng với f(θ)= 1.66

2.1.7. Biến đổi cấp xám tổng thể

Nếu biết ảnh và hàm biến đổi thì ta có thể tính được ảnh kết quả và do đó ta sẽ có được histogram của ảnh biến đổi. Nhưng thực tế nhiều khi ta chỉ biết histogram của ảnh gốc và hàm biến đổi, câu hỏi đặt ra là liệu ta có thể có được histogram của ảnh biến đổi. Nếu có như vậy ta có thể hiệu chỉnh hàm biến đổi để thu được ảnh kết quả có phân bố histogram như mong muốn.

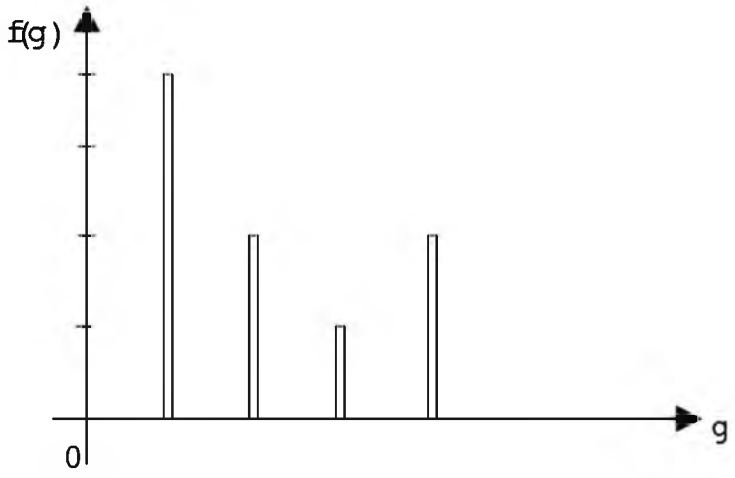
Bài toán đặt ra là biết histogram của ảnh, biết hàm biến đổi hãy vẽ histogram của ảnh mới.

Ví dụ:

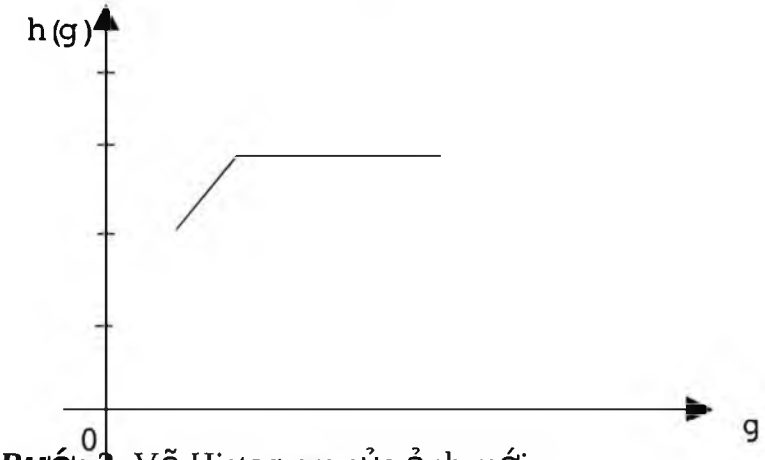
g	1	2	3	4
h(g)	4	2	1	2

$$f(g)=\begin{cases} g+1 & \text{nếu } g \le 2 \\ g & \text{nếu } g = 3 \\ g-1 & \text{nếu } g > 3 \end{cases}$$

Bước 1: Vẽ Histogram của ảnh cũ



Bước 2: Vẽ đồ thị hàm f(g)



Bước 3: Vẽ Histogram của ảnh mới

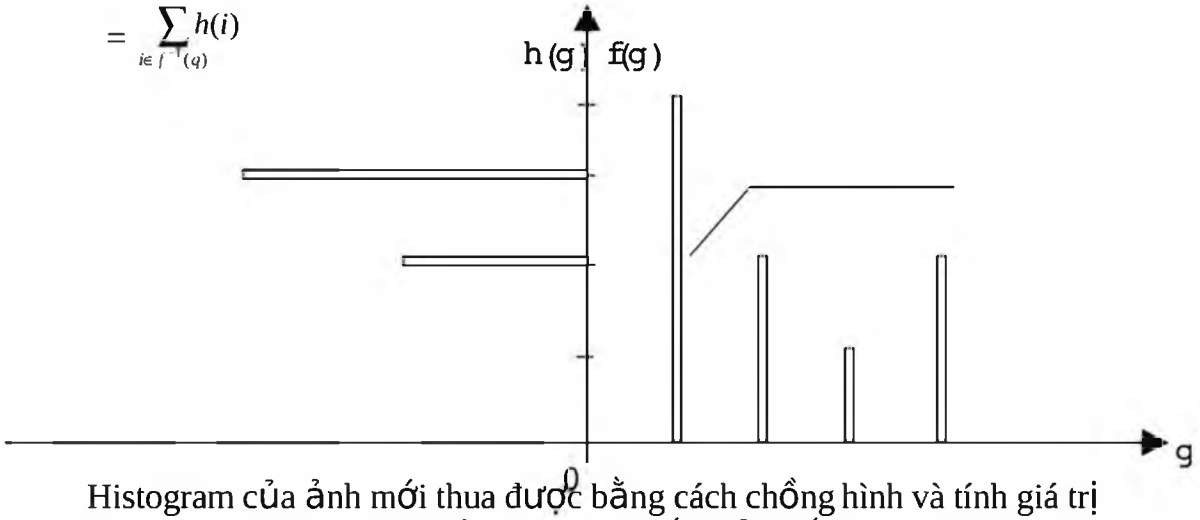
Đặt q = f(g)

h(q) = card ({P| I(P) = q})

= card ({P| I(P) = f(g)})

= card ({P| g = f^{-1} (I(P))})

= $\sum_{i \in f^{-1}(q)} h(i)$



Histogram của ảnh mới thua được bằng cách chồng hình và tính giá trị theo các q (= f(g)) theo công thức tính trên. Kết quả cuối thu được sau phép quay góc 90 thuận chiều kim đồng hồ.

2.2. CÁC KỸ THUẬT PHỤ THUỘC KHÔNG GIAN

2.2.1. Phép nhân chập và mẫu

Giả sử ta có ảnh I kích thước M × N, mẫu T có kích thước m × n khi đó, ảnh I nhân chập theo mẫu T được xác định bởi công thức.

$$I \otimes T(x,y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I(x+i, y+j) * T(i,j) \qquad (2.1)$$

Hoặc $I \otimes T(x,y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I(x-i, y-j) * T(i,j)$ (2.2)

VD:

$$I = \begin{pmatrix} 1 & 2 & 4 & 5 & 8 & 7 \\ 2 & 1 & 1 & 4 & 2 & 2 \\ 4 & 5 & 5 & 8 & 8 & 2 \\ 1 & 2 & 1 & 1 & 4 & 4 \\ 7 & 2 & 2 & 1 & 5 & 2 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$I \otimes T(x,y) = \sum_{i=0}^{1} \sum_{j=0}^{1} I(x+i, y+j) * T(i,j) = I(x,y) * T(0,0) + I(x+1, y+1) * T(1,1)$$

$$= I(x,y) + I(x+1, y+1)$$

$$I \otimes T = \begin{pmatrix} 2 & 3 & 8 & 7 & 10 & * \\ 7 & 6 & 9 & 12 & 4 & * \\ 6 & 6 & 6 & 12 & 12 & * \\ 3 & 4 & 2 & 6 & 6 & * \\ * & * & * & * & * & * \end{pmatrix}$$

Tính theo (2.1)

Tính theo công thức 2.2

$$I \otimes T = \begin{pmatrix} * & * & * & * & * & * \\ * & 2 & 3 & 8 & 7 & 10 \\ * & 7 & 6 & 9 & 12 & 4 \\ * & 6 & 6 & 6 & 12 & 12 \\ * & 3 & 4 & 2 & 6 & 6 \end{pmatrix}$$

**** Nhận xét:***

- Trong quá trình thực hiện phép nhân chập có một số thao tác ra ngoài ảnh, ảnh không được xác định tại những vị trí đó dẫn đến ảnh thu được có kích thước nhỏ hơn.

- Ảnh thực hiện theo công thức 2.1 và 2.2 chỉ sai khác nhau 1 phép dịch chuyển để đơn giản ta sẽ hiểu phép nhân chập là theo công thức 2.1

2.2.2. Một số mẫu thông dụng

- Mẫu:

$$T_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ & & \end{pmatrix}$$

$$\begin{matrix}1 & 1 & 1\end{matrix}$$

~ Dùng để khử nhiễu ⇒ Các điểm có tần số cao

VD1:

$$I = \begin{pmatrix} 1 & 2 & 4 & 5 & 8 & 7 \\ 2 & 31 & 1 & 4 & 2 & 2 \\ 4 & 5 & 5 & 8 & 8 & 2 \\ 1 & 2 & 1 & 1 & 4 & 4 \\ 7 & 2 & 2 & 1 & 5 & 2 \end{pmatrix}$$

$$I \otimes T_1 = \begin{pmatrix} 55 & 65 & 45 & 46 & * & * \\ 52 & 58 & 34 & 35 & * & * \\ 29 & 27 & 35 & 35 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

Áp dụng kỹ thuật cộng hằng số với c = -27, ta có:

$$I_{kq} = \begin{pmatrix} 28 & 38 & 18 & 19 & * & * \\ 25 & 31 & 7 & 8 & * & * \\ 2 & 0 & 8 & 8 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

- Mẫu:

$$T_2 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

~ Dùng để phát hiện các điểm có tần số cao

VD2:

$$I \otimes T2 = -1 \begin{pmatrix} 114 & -40 & 0 & -14 & * & * \\ -22 & 5 & 14 & 16 & * & * \\ -6 & -10 & -2 & * & * & \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

2.2.3. Lọc trung vị

*** Định nghĩa 2.1 (Trung vị)**

Cho dãy x_1; x_2...; x_n đơn điệu tăng (giảm). Khi đó trung vị của dãy ký hiệu là Med($\{x_n\}$), được định nghĩa:

+ Nếu n lẻ $x\left[\frac{n}{2}+1\right]$

+ Nếu n chẵn: $x\left[\frac{n}{2}\right]$ hoặc $x\left[\frac{n}{2}+1\right]$

*** Mệnh đề 2.1**

$$\sum_{i=1}^{n}|x-x_i| \rightarrow \min \text{ tại } Med(\{x_n\})$$

Chứng minh

+ Xét trường hợp n chẵn

Đặt $M=\frac{n}{2}$

Ta có:

$$\sum_{i=1}^{n}|x-x_i| = \sum_{i=1}^{M}|x-x_i|+\sum_{i=1}^{M}|x-x_{M+i}|$$

$$= \sum_{i=1}^{M}\left(|x-x_i|+|x_{M+i}-x|\right) \geq \sum_{i=1}^{M}|x_{M+i}-x_i|$$

$$= \sum_{i=1}^{M}\left[(x_{M+1}-x_M)+(x_M-x_i)\right]$$

$$= \sum_{i=1}^{M}|x_{M+i}-Med(\{x_i\})|+\sum_{i=1}^{M}|x_i-Med(\{x_i\})|$$

$$= \sum_{i=1}^{n}|x_i-Med(\{x_i\})|$$

+ Nếu n lẻ:

Bổ sung thêm phần tử $Med(\{x_i\})$ vào dãy. Theo trường hợp n chẵn ta có:

$$\sum_{i=1}^{n}|x-x_i|+|Med(\{x_i\})-Med(\{x_i\})| \rightarrow \min \text{ tại } \mathrm{Med}(\{x_n\})$$

$$\sum_{i=1}^{n}|x-x_i| \rightarrow \min \text{ tại } \mathrm{Med}(\{x_n\})$$

*** Kỹ thuật lọc trung vị**

Giả sử ta có ảnh I ngưỡng θ cửa sổ W(P) và điểm ảnh P

Khi đó kỹ thuật lọc trung vị phụ thuộc không gian bao gồm các bước cơ bản sau:

+ ***Bước 1:*** Tìm trung vị

$$\{I(q) | \, q \in W(P)\} \rightarrow \text{Med}\,(P)$$

+ ***Bước 2:*** Gán giá trị

$$I(P) = \begin{cases} I(P) & |I(P) - Med(P)| \le \theta \\ Med(P) & Nguoclai \end{cases}$$

Ví dụ:

$$I = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & 16 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

$W(3 \times 3)$; $\theta = 2$

$$I_{kq} = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & \textcircled{2} & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

Giá trị 16, sau phép lọc có giá trị 2, các giá trị còn lại không thay đổi giá trị.

2.2.4. Lọc trung bình

*** Định nghĩa 2.2 (Trung bình)**

Cho dãy $x_1, x_2..., x_n$ khi đó trung bình của dãy ký hiệu $AV(\{x_n\})$ ddược định nghĩa:

$$AV(\{x_n\}) = round\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)$$

*** Mệnh đề 2.2**

$$\sum_{i=1}^{n}(x - x_i)^2 \rightarrow \min \text{ tại } AV(\{x_n\})$$

Chứng minh:

Đặt: $\phi(x) = \sum_{i=1}^{n}(x - x_i)^2$

Ta có:

$$\phi(x) = 2\sum_{i=1}^{n}(x - x_i)$$

$$\phi'(x) = 0$$

$$\Leftrightarrow \sum_{i=1}^{n}(x - x_i) = 0$$

$$\Leftrightarrow x = \frac{1}{n}\sum_{i=1}^{n} x_i = AV(\{x_i\})$$

Mặt khác, $\phi''(x) = 2n > 0$

$$\Rightarrow \phi \rightarrow \min \text{ tại } x = AV(\{x_i\})$$

Kỹ thuật lọc trung bình

Giả sử ta có ảnh I, điểm ảnh P, cửa sổ W(P) và ngưỡng θ. Khi đó kỹ thuật lọc trung bình phụ thuộc không gian bao gồm các bước cơ bản sau:

+ ***Bước 1:*** Tìm trung bình

$\{I(q) | q \in W(P)\} \rightarrow AV(P)$

+ ***Bước 2:*** Gán giá trị

$$I(P) = \begin{cases} I(P) & |I(P) - AV(P)| \le \theta \\ AV(P) & Nguoclai \end{cases}$$

Ví dụ:

$$I = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & 16 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

W(3 × 3); θ = 2

$$I_{kq} = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & \textcircled{3} & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

Giá trị 16 sau phép lọc trung bình có giá trị 3, các giá trị còn lại giữ nguyên sau phép lọc.

2.2.5. Lọc trung bình theo k giá trị gần nhất

Giả sử ta có ảnh I, điểm ảnh P, cửa sổ W(P), ngưỡng θ và số k. Khi đó, lọc trung bình theo k giá trị gần nhất bao gồm các bước sau:

+ ***Bước 1***: Tìm K giá trị gần nhất

$$\{I(q) \mid q \in W(p)\} \rightarrow \{k \sim \text{giá trị gần I(P) nhất}\}$$

+ ***Bước 2:*** Tính trung bình

$$\{k \sim \text{giá trị gần I(P) nhất}\} \rightarrow AV_k(P)$$

+ ***Bước 3:*** Gán giá trị

$$I(P) = \begin{cases} I(P) & |I(P) - AV_k(P)| \le \theta \\ AV_k(P) & Nguoclai \end{cases}$$

Ví dụ:

$$I = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & 16 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

W(3 × 3); θ = 2; k = 3

$$I_{kq} = \begin{pmatrix} 1 & 2 & 3 & 2 \\ 4 & 8 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 2 & 1 & 2 & 1 \end{pmatrix}$$

**** Nhận xét:***

- Nếu k lớn hơn kích thước cửa sổ thì kỹ thuật chính là kỹ thuật lọc trung bình

- Nếu k= 1 thì ảnh kết quả không thay đổi

⇒ Chất lượng của kỹ thuật phụ thuộc vào số phân tử lựa chọn k.

2.3. CÁC PHÉP TOÁN HÌNH THÁI HỌC

2.3.1. Các phép toán hình thái cơ bản

Hình thái là thuật ngữ chỉ sự nghiên cứu về cấu trúc hay hình học topo của đối tượng trong ảnh. Phần lớn các phép toán của "Hình thái" được định nghĩa từ hai phép toán cơ bản là phép "giãn nở" (Dilation) và phép "co" (Erosion).

Các phép toán này được định nghĩa như sau: Giả thiết ta có đối tượng X và phần tử cấu trúc (mẫu) B trong không gian Euclide hai chiều. Kí hiệu B_x là dịch chuyển của B tới vị trí x.

Định nghĩa 2.3 (DILATION)

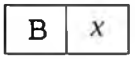
Phép "giãn nở" của X theo mẫu B là hợp của tất cả các B_x với x thuộc X. Ta có:

$$X \oplus B = \bigcup_{x \in X} B_x$$

Định nghĩa 2.4 (EROSION)

Phép "co" của X theo B là tập hợp tất cả các điểm x sao cho B_x nằm trong X. Ta có:

$X \ominus B = \{x : B_x \subseteq X\}$

Ví dụ: Ta có tập X như sau: $X = \begin{pmatrix} 0 & x & 0 & x & x \\ x & 0 & x & x & 0 \\ 0 & x & x & 0 & 0 \\ 0 & x & 0 & x & 0 \\ 0 & x & x & x & 0 \end{pmatrix}$ B = 

$$X \oplus B = \begin{pmatrix} 0 & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & x & 0 \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix} \text{ và } X \ B = \begin{pmatrix} 0 & 0 & 0 & x & 0 \\ 0 & 0 & x & 0 & 0 \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & x & x & 0 & 0 \end{pmatrix}$$

Đình nghĩa 2.5 (OPEN)

Phép toán mở (OPEN) của X theo cấu trúc B là tập hợp các điểm của ảnh X sau khi đã co và giãn nở liên liếp theo B. Ta có:

OPEN(X,B) = (X OB) ⊕ B

Ví dụ: Với tập X và B trong ví dụ trên ta có

$$\text{OPEN}(X,B) = (X(B) \oplus B = \begin{pmatrix} 0 & 0 & 0 & x & x \\ 0 & 0 & x & x & 0 \\ 0 & x & x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & x & x & x & 0 \end{pmatrix}$$

Định nghĩa 2.6 (CLOSE)

Phép toán đóng (CLOSE) của X theo cấu trúc B là tập hợp các điểm của ảnh X sau khi đã giãn nở và co liên tiếp theo B. Ta có:

CLOSE(X,B) = (X ⊕ B) B

Theo ví dụ trên ta có:

$$\text{CLOSE}(X,B) = (X \oplus B) \quad B = \begin{pmatrix} 0 & x & x & x & x \\ x & x & x & x & x \\ 0 & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & x & x & x & 0 \end{pmatrix}$$

2.3.2. Một số tính chất của phép toán hình thái

**** Mệnh đề 2.3*** **[Tính gia tăng]:**

$$\text{(i)} \ X \subseteq X' \Rightarrow \begin{cases} X \ XB \subseteq X' \ B & \forall B \\ X \oplus B \subseteq X' \oplus B & \forall B \end{cases}$$

$$\text{(ii)} \ B \subseteq B' \Rightarrow \begin{cases} X \ XB \supseteq X \ B' & \forall X \\ X \oplus B \subseteq X \oplus B' & \forall X \end{cases}$$

Chứng minh:

(i) $X \oplus B = \bigcup'_{x \in X} B_x \subseteq \bigcup'_{x \in X'} B_x = X' \oplus B$

$X \; XB = \{x / B_x \subseteq X\} \subseteq \{x / B_x \subseteq X'\} = X' \quad B$

(ii) $X \oplus B = \underset{x \in X}{\in} B_x \subseteq \underset{x \in X}{\in} B'_x = X \oplus B'$

Theo định nghĩa:

$X \; XB' = \{x / B'_x \subseteq X\} \subseteq \{x / B_x \subseteq X\} = X \quad B$.

****Mệnh đề 2.4* [Tính phân phối với phép ∪]:**

(i) $X \oplus (B \cup B') = (X \oplus B) \cup (X \oplus B')$

(ii) $XX(B \cup B') = (X \quad B) \cap (X \quad B')$

Chứng minh:

(i) $X \oplus (B \cup B') = (X \oplus B) \cup (X \oplus B')$

Ta có: $B \cup B' \supseteq B$

$X \oplus (B \cup B') \supseteq X \oplus B$ (tính gia tăng)

Tương tự:

$X \oplus (B \cup B') \supseteq X \oplus B'$

$X \oplus (B \cup B') \supseteq (X \oplus B) \cup (X \oplus B')$ (2.3)

Mặt khác,

$\forall y \in X \oplus (B \cup B') \Rightarrow \exists x \in X$ sao cho $y \in (B \cup B')_x$

$\Rightarrow \begin{cases} y \in B_x \\ y \in B'_x \end{cases} \Rightarrow \begin{cases} y \in X \oplus B \\ y \in X \oplus B' \end{cases}$

$\Rightarrow y \in (X \oplus B) \cup (X \oplus B')$

$\Rightarrow X \oplus (B \cup B') \subseteq (X \oplus B) \cup (X \oplus B')$ (2.4)

Từ (2.3) và (2.4) ta có: $X \oplus (B \cup B') = (X \oplus B) \cup (X \oplus B')$

(ii) $X \; X(B \cup B') = (X \quad B) \cap (X \quad B')$

Ta có: $B \cup B' \supseteq B$

$\Rightarrow$ $X \; X (B \cup B') \subseteq X \quad B$ (tính gia tăng)

Tương tự : $X \; X (B \cup B') \subseteq X \quad B'$

$\Rightarrow$ $X \; X(B \cup B') \subseteq (X \quad B) \cap (X \quad B')$ (2.5)

Mặt khác,

$\forall x \in (X \quad B) \cap (X \quad B')$

Suy ra, $\begin{cases} x \in X \quad B \\ x \in X \quad B' \end{cases} \Rightarrow \begin{cases} B_x \subseteq X \\ B'_x \subseteq X \end{cases}$

$\Rightarrow (B \cup B')_x \subseteq X$

$\Rightarrow x \in X \quad (B \cup B')$

$\Rightarrow X \quad (B \cup B') \supseteq (X \quad B) \cap (X \quad B')$ (2.6)

Từ (2.5) và (2.6) ta có: X ừ(B ∪ B') = (X B) ∩ (X B').

**** Ý nghĩa:***

Ta có thể phân tích các mẫu phức tạp trở thành các mẫu đơn giản thuận tiện cho việc cài đặt.

**** Mệnh đề 2.5* [Tính phân phối với phép ∩]:**

$(X \cap Y) \quad B = (X \quad B) \cap (Y \quad B)$

Chứng minh:

Ta có, $X \cap Y \subseteq X$

$\Rightarrow \quad (X \cap Y) \quad B \subseteq X \quad B$

Tương tự: $(X \cap Y) \quad B \subseteq Y \quad B$

$\Rightarrow \quad (X \cap Y) \quad B \subseteq (X \quad B) \cap (Y \quad B)$ (2.7)

Mặt khác,

$\forall x \in (X \quad B) \cap (Y \quad B)$

Suy ra $\begin{cases} x \in X \quad B \\ x \in Y \quad B \end{cases} \Rightarrow \quad B_x \subseteq \begin{cases} X \\ Y \end{cases}$

$\Rightarrow Bx \subseteq X \cap Y$

$\Rightarrow x \in (X \cap Y) \quad B$

$\Rightarrow (X \cap Y) \quad B \supseteq (X \quad B) \cap (Y \quad B)$ (2.8)

Từ (2.7) và (2.8) ta có: $(X \cap Y) \quad B = (X \quad B) \cap (Y \quad B)$.

**** Mệnh đề 2.6* [Tính kết hợp]**

(i) $(X \oplus B) \oplus B' = X \oplus (B \oplus B')$

(ii) $(X \quad B) \quad B' = X \quad (B \oplus B')$

Chứng minh:

(i) $(X \oplus B) \oplus B' = X \oplus (B' \oplus B)$

Ta có, $(X \oplus B) \oplus B' = (\underset{x \in X}{=} B_x) \oplus B'$

$= \underset{x \in X}{=} (B_x \oplus B') = \underset{x \in X}{=} (B \oplus B')_x$

$= X \oplus (B' \oplus B)$

(i) (X (B) B' = X (B ⊕ B')

Trước hết ta đi chứng minh: $B'_x \subseteq$ X B $\Leftrightarrow (B' \oplus B)_x \subseteq$ X

Thật vậy, do $B'_x \subseteq$ X B nên $\forall y \in B'_x \Rightarrow y \in$ X XB

$\Rightarrow B_y \subseteq X$

$\Rightarrow \underset{y \in B'_x}{\in} B_y \subseteq X$

$\Rightarrow (B' \oplus B)_x \subseteq$ X

Mặt khác, $(B' \oplus B)_x \subseteq$ X $\Leftrightarrow (B'_x \oplus$ B) $\subseteq$ X

$\Leftrightarrow \underset{y \in B'_x}{B_y} \subseteq$ X

$\Rightarrow \forall y \in B'_x$ ta có $B_y \subseteq$ X

$\Rightarrow$ hay $\forall y \in B'_x$ ta có y $\in$ X B

Do đó, $B'_x \subseteq$ X B

Ta có, (X T B) B' $= \{x / B_x \subseteq X\}$ B'

= {x/ $B'_x \subseteq$ X B}

= {x/ $(B' \oplus B)_x \subseteq$ X} (do chứng minh ở trên)

= X = (B ⊕ B') .

**** Định lý 2.1*** **[X bị chặn bởi các cận OPEN và CLOSE]**

Giả sử, X là một đối tượng ảnh, B là mẫu, khi đó, X sẽ bị chặn trên bởi tập CLOSE của X theo B và bị chặn dưới bởi tập OPEN của X theo B. Tức là:

(X ⊕ B) B $\supseteq$ X $\supseteq$ (X B) ⊕ B

Chứng minh:

Ta có: $\forall x \in X \Rightarrow B_x \subseteq X \oplus B$ (Vì $X \oplus B = \bigcup_{x \in X} B_x$)

$\Rightarrow x \in (X \oplus B)$ B (theo định nghĩa phép co)

$\Rightarrow (X \oplus B)$ B $\supseteq X$ (2.9)

Mặt khác,

$\forall y \in (X$ B) $\oplus$ B, suy ra:

$\exists x \in X$ B sao cho $y \in B_x$ (Vì (X(B) $\oplus$ B = $\bigcup_{x \in X \ominus B} B_x$)

$\Rightarrow B_x \subseteq X \Rightarrow y \in X$

Suy ra: $X \supseteq (X$ B) $\oplus$ B (2.10)

Từ (2.9) và (2.10) Ta có: $(X \oplus B)$ B $\supseteq X \supseteq (X$ B) $\oplus$ B .

***Hệ quả 2.1 [Tính bất biến] :**

(i) $((X \oplus B)$ B) $\oplus$ B = $X \oplus B$

(ii) ((X (B) $\oplus$ B) B = X B

Chứng minh:

(i) Thật vậy, từ định lý 2.1 ta có $X \subseteq (X \oplus B) \ominus B$

$\Rightarrow X \oplus B \subseteq ((X \oplus B)$ B) $\oplus$ B (do tính chất gia tăng) (2.11)

Mặt khác, cũng từ định lý 2.1 ta có (X ¡ B) $\oplus$ B $\subseteq X \forall X$

Do đó, thay X bởi $X \oplus B$ ta có, $((X \oplus B)$ B) $\oplus$ B $\subseteq X \oplus B$ (2.12)

Từ (2.11) và (2.12) Ta có: $((X \oplus B)$ B) $\oplus$ B = $X \oplus B$

(ii) Thật vậy, từ định lý 2.1 ta có (X ¡ B) $\oplus$ B $\subseteq X$

$\Rightarrow ((X$ B) $\oplus$ B) B $\subseteq X$ B (do tính chất gia tăng) (2.13)

Mặt khác, cũng từ định lý 2.1 ta có $X \subseteq (X \oplus B) \ominus B \forall X$

Do đó, thay X bởi X ởB ta có, X B $\subseteq ((X$ B) $\oplus$ B) B (2.14)

Từ (2.13) và (2.14) Ta có: ((X ừB) $\oplus$ B) B = X B (đpcm).

Chương 3:

BIÊN VÀ CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN

3.1. GIỚI THIỆU

Biên là vấn đề quan trọng trong trích chọn đặc điểm nhằm tiến tới hiểu ảnh. Cho đến nay chưa có định nghĩa chính xác về biên, trong mỗi ứng dụng người ta đưa ra các độ đo khác nhau về biên, một trong các độ đo đó là độ đo về sự thay đổi đột ngột về cấp xám. Ví dụ: Đối với ảnh đen trắng, một điểm được gọi là điểm biên nếu nó là điểm đen có ít nhất một điểm trắng bên cạnh. Tập hợp các điểm biên tạo nên biên hay đường bao của đối tượng. Xuất phát từ cơ sở này người ta thường sử dụng hai phương pháp phát hiện biên cơ bản:

Phát hiện biên trực tiếp: Phương pháp này làm nổi biên dựa vào sự biến thiên mức xám của ảnh. Kỹ thuật chủ yếu dùng để phát hiện biên ở đây là kỹ thuật lấy đạo hàm. Nếu lấy đạo hàm bậc nhất của ảnh ta có các kỹ thuật Gradient, nếu lấy đạo hàm bậc hai của ảnh ta có kỹ thuật Laplace. Ngoài ra còn có một số các tiếp cận khác

Phát hiện biên gián tiếp: Nếu bằng cách nào đó ta phân được ảnh thành các vùng thì ranh giới giữa các vùng đó gọi là biên. Kỹ thuật dò biên và phân vùng ảnh là hai bài toán đối ngẫu nhau vì dò biên để thực hiện phân lớp đối tượng mà khi đã phân lớp xong nghĩa là đã phân vùng được ảnh và ngược lại, khi đã phân vùng ảnh đã được phân lớp thành các đối tượng, do đó có thể phát hiện được biên.

Phương pháp phát hiện biên trực tiếp tỏ ra khá hiệu quả và ít chịu ảnh hưởng của nhiễu, song nếu sự biến thiên độ sáng không đột ngột, phương pháp tỏ ra kém hiệu quả, phương pháp phát hiện biên gián tiếp tuy khó cài đặt, song lại áp dụng khá tốt trong trường hợp này. Sự khác biệt cơ bản giữa hai phương pháp này là: Phương pháp phát hiện biên trực tiếp cho ta kết quả là ảnh biên, còn phương pháp phát hiện biên trực tiếp cho ta kết quả là đường biên.

3.2. CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN TRỰC TIẾP

3.2.1. Kỹ thuật phát hiện biên Gradient

Theo định nghĩa, gradient là một véctơ có các thành phần biểu thị tốc độ thay đổi giá trị của điểm ảnh, ta có:

$$\begin{cases} \dfrac{\partial f(x,y)}{\partial x} = fx \approx \dfrac{f(x+dx,y)-f(x,y)}{dx} \\ \dfrac{\partial f(x,y)}{\partial y} = fy \approx \dfrac{f(x,y+dy)-f(x,y)}{dy} \end{cases}$$

Trong đó, dx, dy là khoảng cách (tính bằng số điểm) theo hướng x và y.

**** Nhận xét:***

Tuy ta nói là lấy đạo hàm nhưng thực chất chỉ là mô phỏng và xấp xỉ đạo hàm bằng các kỹ thuật nhân chập vì ảnh số là tín hiệu rời rạc nên đạo hàm không tồn tại.

Ví dụ: Với dx = dy = 1, ta có:

$$\begin{cases} \dfrac{\partial f}{\partial x} \approx f(x+1,y) - f(x,y) \\ \dfrac{\partial f}{\partial y} \approx f(x,y+1) - f(x,y) \end{cases}$$

Do đó, mặt nạ nhân chập theo hướng x là $A = \begin{pmatrix} -1 & 1 \end{pmatrix}$

và hướng y là $B = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

Chẳng hạn:

$$I = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 3 \\ 0 & 3 & 3 & 3 \\ 0 & 3 & 3 & 3 \end{pmatrix}$$

Ta có,

$$I \otimes A = \begin{pmatrix} 0 & 0 & 0 & * \\ 3 & 0 & 0 & * \\ 3 & 0 & 0 & * \\ * & * & * & * \end{pmatrix}; \quad I \otimes B = \begin{pmatrix} 0 & 3 & 3 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ * & * & * & * \end{pmatrix}$$

$$I \otimes A + I \otimes B = \begin{pmatrix} 0 & 0 & 0 & * \\ 3 & 0 & 0 & * \\ 3 & 0 & 0 & * \\ * & * & * & * \end{pmatrix}$$

3.2.1.1. Kỹ thuật Prewitt

Kỹ thuật sử dụng 2 mặt nạ nhập chập xấp xỉ đạo hàm theo 2 hướng x và y là:

$$H_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

$$H_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Các bước tính toán của kỹ thuật Prewitt

+ ***Bước 1:*** Tính $I \otimes H_x$ và $I \otimes H_y$

+ ***Bước 2:*** Tính $I \otimes H_x + I \otimes H_y$

Ví dụ:

$$I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$I \otimes H_x = \begin{pmatrix} 0 & 0 & -10 & -10 & * & * \\ 0 & 0 & -15 & -15 & * & * \\ 0 & 0 & -10 & -10 & * & * \\ 0 & 0 & -5 & -5 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

$$I \otimes H_y = \begin{pmatrix} 15 & 15 & 10 & 5 & * & * \\ 0 & 0 & 0 & 0 & * & * \\ -15 & -15 & -10 & -5 & * & * \\ -15 & -15 & -10 & -5 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

$$I \otimes H_x + I \otimes H_y = -15 \begin{pmatrix} 15 & 15 & 0 & -5 & * & * \\ 0 & 0 & -15 & -15 & * & * \\ -15 & -20 & -15 & * & * & \\ -15 & -15 & -15 & -10 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

3.2.1.2. Kỹ thuật Sobel

Tương tự như kỹ thuật Prewitt kỹ thuật Sobel sử dụng 2 mặt nạ nhân chập theo 2 hướng x, y là:

$$H_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$H_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Các bước tính toán tương tự Prewitt

+ ***Bước 1:*** Tính $I \otimes H_x$ và $I \otimes H_y$

+ ***Bước 2:*** Tính $I \otimes H_x + I \otimes H_y$

3.2.1.3. Kỹ thuật la bàn

Kỹ thuật sử dụng 8 mặt nạ nhân chập theo 8 hướng 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0

$$H_1 = \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \qquad H_2 = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

$$H_3 = \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{pmatrix} \qquad H_4 = \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix}$$

$$\begin{pmatrix} -3 & -3 & -3 \\ & & \\ & & \end{pmatrix} \qquad \begin{pmatrix} -3 & -3 & -3 \\ & & \\ & & \end{pmatrix}$$

$$H_5 = \begin{matrix} -3 & 0 & 5 \\ -3 & 5 & 5 \end{matrix} \qquad H_6 = \begin{matrix} -3 & 0 & -3 \\ 5 & 5 & 5 \end{matrix}$$

$$H_7 = \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix} \qquad H_8 = \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix}$$

Các bước tính toán thuật toán La bàn

+ ***Bước 1:*** Tính $I \otimes H_i$; i = 1,8

+ ***Bước 2***: $\sum_{i=1}^{8} I \otimes H_i$

3.2.2. Kỹ thuật phát hiện biên Laplace

Các phương pháp đánh giá gradient ở trên làm việc khá tốt khi mà độ sáng thay đổi rõ nét. Khi mức xám thay đổi chậm, miền chuyển tiếp trải rộng, phương pháp cho hiệu quả hơn đó là phương pháp sử dụng đạo hàm bậc hai Laplace.

Toán tử Laplace được định nghĩa như sau:

Ta có: $$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) \approx \frac{\partial}{\partial x}\left(f(x+1,y) - f(x,y)\right)$$

$$\approx \left[f(x+1,y) - f(x,y)\right] - \left[f(x,y) - f(x-1,y)\right]$$

$$\approx f(x+1,y) - 2f(x,y) + f(x-1,y)$$

Tương tự,

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right) \approx \frac{\partial}{\partial y}\left(f(x,y+1) - f(x,y)\right)$$

$$\approx \left[f(x,y+1) - f(x,y)\right] - \left[f(x,y) - f(x,y-1)\right]$$

$$\approx f(x,y+1) - 2f(x,y) + f(x,y-1)$$

Vậy: ∇^2 f= f(x+1,y) + f(x,y+1) - 4f(x,y) + f(x-1,y) + f(x,y-1)

Dẫn tới:

$$H = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Trong thực tế, người ta thường dùng nhiều kiểu mặt nạ khác nhau để xấp xỉ rời rạc đạo hàm bậc hai Laplace. Dưới đây là ba kiểu mặt nạ thường dùng:

$$H_1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad H_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad H_3 = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

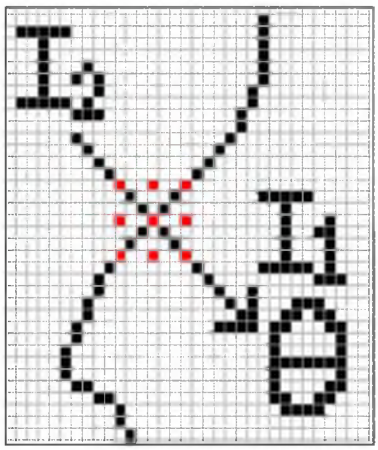
VD:

$$I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3.2.3. Kỹ thuật Canny

Đây là một thuật toán tương đối tốt, có khả năng đưa ra đường biên mảnh, và phát hiện chính xác điểm biên với điểm nhiễu.

Thuật toán



+ ***Bước 1:*** Làm trơn ảnh

Tính I ⊗ H, với:

$$H = \frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Gọi G là kết quả lọc nhiễu: G= I ⊗ H

+***Bước 2:*** Tính gradient của ảnh bằng mặt nạ PreWitt, kết quả đặt vào G_x,G_y.

$G_x = G \otimes H_x$, $G_y = G \otimes H_y$

+ ***Bước3:*** Tính gradient hướng tại mỗi điểm (i,j) của ảnh. Hướng này sẽ được nguyên hóa để nằm trong 8 hướng [0..7], tương đương với 8 lân cận của một điểm ảnh.

+ ***Bước 4:*** Dùng ràng buộc "loại bỏ những điểm không phải là cực đại" để xóa bỏ những điểm không là biên. Xét (i,j), θ là gradient hướng tại (i,j). I1, I2 là hai điểm lân cận của (i,j) theo hướng θ. Theo định nghĩa điểm biên cục bộ thì (i,j) là biên nếu I(i,j) cực đại địa phương theo hướng gradient → Nếu I(i,j) > I1 và I(i,j) > I2 thì mới giữ lại I(i,j), ngược lại xóa I(i,j) về điểm ảnh nền.

+ ***Bước 5:*** Phân ngưỡng. Với các điểm được giữ lại, thực hiện lấy ngưỡng gradient biên độ lần cuối để xác định các điểm biên thực sự.

3.3. PHÁT HIỆN BIÊN GIÁN TIẾP

3.3.1 Một số khái niệm cơ bản

***Ảnh và điểm ảnh**

Ảnh số là một mảng số thực 2 chiều (I_{ij}) có kích thước (M×N), trong đó mỗi phần tử I_{ij}(i = 1,...,M; j = 1,...,N) biểu thị mức xám của ảnh tại (i,j) tương ứng.

Ảnh được gọi là ảnh nhị phân nếu các giá trị I_{ij} chỉ nhận giá trị 0 hoặc 1.

Ở đây ta chỉ xét tới ảnh nhị phân vì ảnh bất kỳ có thể đưa về dạng nhị phân bằng kỹ thuật phân ngưỡng. Ta ký hiệu $\Im$ là tập các điểm vùng (điểm đen) và $\overline{\Im}$ là tập các điểm nền (điểm trắng).

***Các điểm 4 và 8-láng giềng**

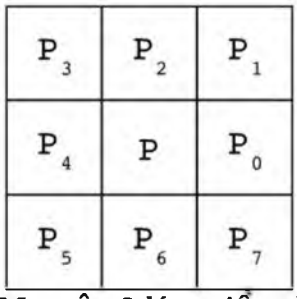
Giả sử (i,j) là một điểm ảnh, các điểm 4-láng giềng là các điểm kề trên, dưới, trái, phải của (i,j):

$$N_4(i,j) = \{(i',j') : |i-i'|+|j-j'| = 1\},$$

và những điểm 8-láng giềng gồm:

$$N_8(i,j) = \{(i',j') : \max(|i-i'|,|j-j'|) = 1\}.$$

Trong Hình 3.1 biểu diễn ma trận 8 láng giềng kề nhau, các điểm P_0, P_2, P_4, P_6 là các 4-láng giềng của điểm P, còn các điểm P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_7 là các 8-láng giềng của P.



Hình 3.1. Ma trận 8-láng giềng kề nhau

***Đối tượng ảnh**

Hai điểm P_s, $P_e \in E$, $E \subseteq \Im$ hoặc $\overline{\Im}$ được gọi là 8-liên thông (hoặc 4-liên thông) trong E nếu tồn tại tập các điểm được gọi là ***đường đi*** (i_o,j_o)... (i_n,j_n) sao cho $(i_o,j_o)= P_s$, $(i_n,j_n)= P_e$, $(i_r,j_r) \in E$ và (i_r,j_r) là 8-láng giềng (hoặc 4-láng giềng tương ứng) của (i_{r-1},j_{r-1}) với r = 1,2,...,n

Nhận xét: Quan hệ ***k-liên thông trong E*** (k=4,8) là một quan hệ phản xạ, đối xứng và bắc cầu. Bởi vậy đó là một quan hệ tương đương. Mỗi lớp tương đương được gọi là một thành phần k-liên thông của ảnh. Về sau ta sẽ gọi mỗi thành phần k-liên thông của ảnh là một đối tượng ảnh.

3.3.2. Chu tuyến của một đối tượng ảnh

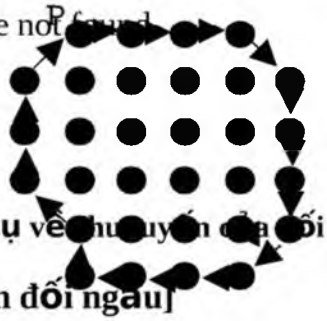
Định nghĩa 3.1: [Chu tuyến]

Chu tuyến của một đối tượng ảnh là dãy các điểm của đối tượng ảnh $P_1,\ldots,P_n$ sao cho P_i và P_{i+1} là các 8-láng giềng của nhau (i=1,...,n-1) và P_1 là 8-láng giềng của P_n, $\forall i \exists Q$ không thuộc đối tượng ảnh và Q là 4-láng giềng của P_i (hay nói cách khác $\forall i$ thì Pi là biên 4). Kí hiệu $<P_1P_2..P_n>$.

Tổng các khoảng cách giữa hai điểm kế tiếp của chu tuyến là độ dài của chu tuyến và kí hiệu Len(C) và hướng P_iP_{i+1} là hướng chẵn nếu P_i và P_{i+1} là các 4 – láng giềng (trường hợp còn lại thì P_iP_{i+1} là hướng lẻ).

Hình 3.2 dưới đây biểu diễn chu tuyến của ảnh, trong đó, P là điểm khởi đầu chu tuyến.

Error: Reference source not found



Hình 3.2. Ví dụ về chu tuyến của đối tượng ảnh

Định nghĩa 3.2 [Chu tuyến đối ngẫu]

Hai chu tuyến $C= <P_1P_2..P_n>$ và $C^{\perp}= <Q_1Q_2..Q_m>$ được gọi là đối ngẫu của nhau nếu và chỉ nếu $\forall i \exists j$ sao cho:

(i) P_i và Q_j là 4-láng giềng của nhau.

(ii) Các điểm P_i là vùng thì Q_j là nền và ngược lại.

Định nghĩa 3.3 [Chu tuyến ngoài]

Chu tuyến C được gọi là chu tuyến ngoài (Hình 3.3a) nếu và chỉ nếu

(i) Chu tuyến đối ngẫu $C^{\perp}$ là chu tuyến của các điểm nền

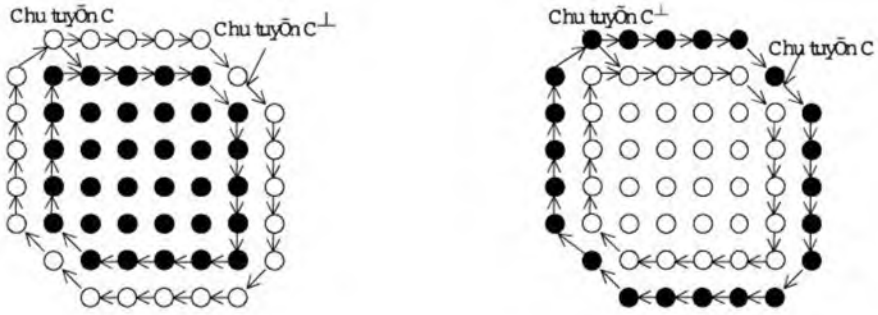
(ii) Độ dài của C nhỏ hơn độ dài $C^{\perp}$

Định nghĩa 3.4 [Chu tuyến trong]

Chu tuyến C được gọi là chu tuyến trong (Hình 3.3b) nếu và chỉ nếu:

(i) Chu tuyến đối ngẫu $C^{\perp}$ là chu tuyến của các điểm nền

(ii) Độ dài của C lớn hơn độ dài $C^{\perp}$



a) Chu tuyến ngoài **b) Chu tuyến trong**

Hình 3.3. Chu tuyến trong, chu tuyến ngoài

Định nghĩa 3.5 [Điểm trong và điểm ngoài chu tuyến]

Giả sử $C= <P_1P_2..P_n>$ là chu tuyến của một đối tượng ảnh và P là một điểm ảnh. Khi đó:

(i) Nếu nửa đường thẳng xuất phát từ P sẽ cắt chu tuyến C tại số lẻ lần, thì P được gọi là điểm trong chu tuyến C và kí hiệu in(P,C)

(ii) Nếu $P \notin C$ và P không phải là điểm trong của C, thì P được gọi là điểm ngoài chu tuyến C và kí hiệu out(P,C).

Bổ đề 3.1 [Chu tuyến đối ngẫu]

Giả sử $E \subseteq \Im$ là một đối tượng ảnh và $C= < P_1P_2..P_n>$ là chu tuyến của E, $C^{\perp}=<Q_1Q_2..Q_m>$ là chu tuyến đối ngẫu tương ứng. Khi đó:

(i) Nếu C là chu tuyến trong thì $in(Q_i,C)$ $\forall i$ (i=1,....,m)

(ii) Nếu C là chu tuyến ngoài thì $in(P_i,C^{\perp})$ $\forall i$ (i=1,...,n)

Bổ đề 3.2 [Phần trong/ngoài của chu tuyến]

Giả sử $E \subseteq \Im$ là một đối tượng ảnh và C là chu tuyến của E. Khi đó:

(i) Nếu C là chu tuyến ngoài thì $\forall x \in E$ sao cho $x \notin C$, ta có in(x,C)

(ii) Nếu C là chu tuyến trong thì $\forall x \in E$ sao cho $x \notin C$, ta có out(x,C)

Định lý 3.1 [Tính duy nhất của chu tuyến ngoài]

Giả sử $E \subseteq \Im$ là một đối tượng ảnh và C_E là chu tuyến ngoài của E. Khi đó C_E là duy nhất.

3.3.3. Thuật toán dò biên tổng quát

Biểu diễn đối tượng ảnh theo chu tuyến thường dựa trên các kỹ thuật dò biên. Có hai kỹ thuật dò biên cơ bản. Kỹ thuật thứ nhất xét ảnh biên thu được từ ảnh vùng sau một lần duyệt như một đồ thị, sau đó áp dụng các thuật toán duyệt cạnh dồ thị. Kỹ thuật thứ hai dựa trên ảnh vùng, kết hợp đồng thời quá trình dò biên và tách biên. Ở đây ta quan tâm cách tiếp cận thứ hai.

Trước hết, giả sử ảnh được xét chỉ bao gồm một vùng ảnh 8-liên thông $\Im$, được bao bọc bởi một vành đai các điểm nền. Dễ thấy $\Im$ là một vùng 4-liên thông chỉ là một trường riêng của trường hợp trên.

Về cơ bản, các thuật toán dò biên trên một vùng đều bao gồm các bước sau:

- Xác định điểm biên xuất phát
- Dự báo và xác định điểm biên tiếp theo
- Lặp bước 2 cho đến khi gặp điểm xuất phát

Do xuất phát từ những tiêu chuẩn và định nghĩa khác nhau về điểm biên, và quan hệ liên thông, các thuật toán dò biên cho ta các đường biên mang các sắc thái rất khác nhau.

Kết quả tác động của toán tử dò biên lên một điểm biên r_i là điểm biên r_{i+1} (8-láng giềng của r_i). Thông thường các toán tử này được xây dựng như một hàm đại số Boolean trên các 8-láng giềng của r_i. Mỗi cách xây dựng các toán tử đều phụ thuộc vào định nghĩa quan hệ liên thông và điểm biên. Do đó sẽ gây khó khăn cho việc khảo sát các tính chất của đường biên. Ngoài ra, vì mỗi bước dò biên đều phải kiểm tra tất cả các 8-láng giềng của mỗi điểm nên thuật toán thường kém hiệu quả. Để khắc phục các hạn chế trên, thay vì sử dụng một điểm biên ta sử dụng cặp điểm biên (một thuộc $\Im$, một thuộc $\overline{\Im}$), các cặp điểm này tạo nên tập nền vùng, kí hiệu là NV và phân tích toán tử dò biên thành 2 bước:

- Xác định cặp điểm nền vùng tiếp theo.
- Lựa chọn điểm biên

Trong đó bước thứ nhất thực hiện chức năng của một ánh xạ trên tập NV lên NV và bước thứ hai thực hiện chức năng chọn điểm biên.

Thuật toán dò biên tổng quát

Bước 1: Xác định cặp nền-vùng xuất phát

Bước 2: Xác định cặp nền-vùng tiếp theo

Bước 3: Lựa chọn điểm biên vùng

Bước 4: Nếu gặp lại cặp xuất phát thì dừng, nếu không quay lại bước 2.

Việc xác định cặp nền-vùng xuất phát được thực hiện bằng cách duyệt ảnh lần lượt từ trên xuống dưới và từ trái qua phải rồi kiểm tra điều kiện lựa chọn cặp nền-vùng. Do việc chọn điểm biên chỉ mang tính chất quy ước, nên ta gọi ánh xạ xác định cặp nền-vùng tiếp theo là toán tử dò biên.

Định nghĩa 3.6 [Toán tử dò biên]

Giả sử T là một ánh xạ như sau: T: NV $\rightarrow$ NV

(b,r) ((b',r')

Gọi T là một toán tử dò biên cơ sở nếu nó thoả mãn điều kiện: b',r' là các 8-láng giềng của r.

Giả sử (b,r) $\in$ NV; gọi K(b,r) là hàm chọn điểm biên. Biên của một dạng $\Im$ có thể định nghĩa theo một trong ba cách:

- Tập những điểm thuộc $\Im$ có mặt trên NV, tức là K(b,r)= r
- Tập những điểm thuộc $\overline{\Im}$ có trên NV, tức là K(b,r)= b
- Tập những điểm ảo nằm giữa cặp nền-vùng, tức là K(b,r) là những điểm nằm giữa hai điểm b và r.

Cách định nghĩa thứ ba tương ứng mỗi cặp nền-vùng với một điểm biên. Còn đối với cách định nghĩa thứ nhất và thứ hai một số cặp nền-vùng có thể có chung một điểm biên. Bởi vậy, quá trình chọn điểm biên được thực hiện như sau:

i:= 1; (b_i,r_i):= (b_o,r_o);

While $K(b_i,r_i)<>K(b_n,r_n)$ and $i\leq 8$ **do**

Begin

$(b_{i+1},r_{i+1})= T(b_i,r_i)$; i:= i+1;

End;

Điều kiện dừng

Cặp nền-vùng thứ n trùng với cặp nền vùng xuất phát: $(b_n,r_n)= (b_o,r_o)$

*** Xác định cặp nền – vùng xuất phát**

Cặp nền vùng xuất phát được xác định bằng cách duyệt ảnh lần lượt từ trên xuống dưới và từ trái sang phải điểm đem đầu tiên gặp được cùng với điểm trắng trước đó (theo hướng 4) để tạo nên cặp nền vùng xuất phát.

*** Xác định cặp nền vùng tiếp theo**

Đầu vào: pt, dir

Ví dụ: (3, 2) 4

```
Point orient []= {(1,0);(1;-1);(0;-1);(-1;-1);(-1;0);(-1,1);(0,1);(1,1)};

//Hàm tìm hướng có điểm đen gần nhất
BYTE GextNextDir(POINT pt, BYTE dir)
{
   BYTE pdir= (dir + 7)%8;
   do{
      if(getpixel(pt. x+orient [pdir]. x,pt.y+orient [pdir]. y))==BLACK)
         return pdir;
      pdir = (pdir + 7) %8;
   }while(pdir ! = dir);
   return. ERR; //Điểm cô lập
}

//Gán giá trị cho bước tiếp theo
pdir = GetNextDir(pt, dir);
if(pdir==ERR) //Kiểm tra có là điểm cô lập không?
   return. ERR; //Điểm cô lập
pt. x = pt. x + orient [pdir]. x;
pt. y = pt. y + orient [pdir]. y ;
```

Để tính giá trị cho hướng tiếp theo ta lập bảng dựa trên giá trị pdir đã tính được trước đó theo các khả năng có thể xảy ra:

pdir	Điểm trắng trước đó	Trắng so với đen mới
0	1	2
1	2	4
2	3	4
3	4	6
4	5	6
5	6	0
6	7	0
7	0	2

⇒ Do đó công thức để tính hướng tiếp theo sẽ là :

dir= ((pdir+3)/ 2 * 2)%8 ;

3.4. PHÁT HIỆN BIÊN DỰA VÀO TRUNG BÌNH CỤC BỘ

3.4.1. Biên và độ biến đổi về mức xám

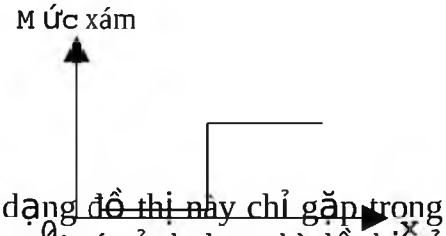
Như đã trình bày ở trên, trong thực tế người ta thường dùng hai phương pháp pháp hiện biên cơ bản là: Phát hiện biên trực tiếp và gián tiếp. Phần này đề cập đến kỹ thuật mới dựa vào trung bình cục bộ trên cơ sở đánh giá độ chênh lệch về giá trị mức xám của điểm ảnh so với các điểm lân cận do đó kết hợp được ưu điểm của cả hai khuynh hướng trực tiếp và gián tiếp.

Đối với các ảnh màu theo mô hình nào đó đều có thể chuyển sang mô hình gồm 3 thành phần mầu R, G, B. Sau đó dễ dàng chuyển các ảnh màu sang dạng ảnh đa cấp xám. Chẳng hạn:

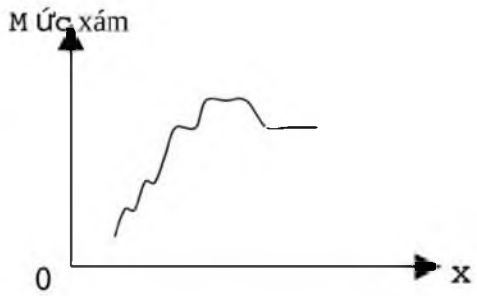
Gray = (R + G + B) / 3

Việc xử lý, thao tác trên các ảnh xám có một ưu điểm là dễ xử lý hơn các ảnh màu mà vẫn giữ được các đặc tính của ảnh. Các ảnh trắng đen tuy dễ xử lý nhất nhưng sẽ bị mất nhiều chi tiết sau khi chuyển đổi.

Một cách lý tưởng đồ thị biến thiên mức xám của điểm ảnh khi qua biên phải có dạng:



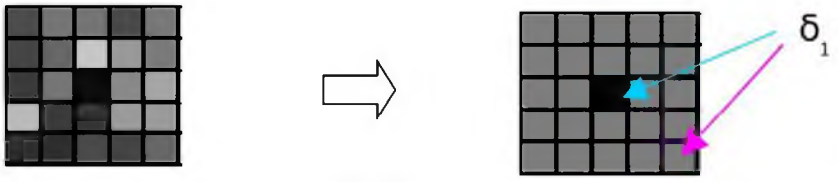
Trong thực tế dạng đồ thị này chỉ gặp trong các ảnh trắng đen (ảnh xám có hai màu), còn với các ảnh thực thì đồ thị của nó có dạng:



Khó khăn cho việc phân tích các ảnh thực là ở chỗ do sự biến thiên về mức xám của điểm ảnh không phải chỉ được thể hiện theo một hướng duy nhất mà phải xét theo cả tám hướng của các điểm ảnh láng giềng, tại các vùng biên và lân cận biên sự biến thiên mức xám của các điểm ảnh thường không đột ngột mà trải qua một khoảng biến thiên không đều nhưng có tốc độ biến thiên nhanh. Chúng ta có thể xác định được các đường biên như thế này bằng kỹ thuật Laplace nhưng như ở trên đã nói kỹ thuật này rất nhạy cảm với nhiễu mà nhiễu hầu như lại là vấn đề mà ở trong bức ảnh nào cũng có. Ngoài ra, trong thực tế khi dò biên cho các ảnh xám tùy theo mục đích xử lý sau này mà người ta có thể muốn lấy biên của tất cả các đối tượng trong ảnh hoặc chỉ một số đối tượng chính trong ảnh. Các kỹ thuật đạo hàm do sử dụng các mặt nạ là các ma trận nhân chập nên khó điều chỉnh độ chi tiết của ảnh biên thu được. Muốn làm được điều này lại phải tính toán lại các giá trị của các phần tử trong ma trận theo các công thức nhất định, rất phức tạp và tốn kém. Không những thế ảnh thu được sau khi lọc không làm mất đi được tất cả các điểm không thuộc đường biên mà chỉ làm nổi lên các điểm nằm trên biên và muốn nhận dạng được các đối tượng thì ta còn phải xử lý thêm một vài bước nữa thì mới thu được ảnh biên thực sự. Có thể nhận thấy là các thuật toán dò biên truyền thống mà chúng ta hay dùng vẫn chưa đạt được sự hoàn thiện như mong muốn [3,8].

3.4.2. Phát hiện biên dựa vào trung bình cục bộ

Ý tưởng chính của thuật toán được đề xuất là: Xác định tất cả các điểm nằm trên biên không theo hướng tìm kiếm và sử dụng các ma trận lọc, thông qua việc so sánh độ chênh lệch về mức xám của nó so với mức xám chung của các điểm ảnh lân cận (mức xám nền). Trước hết giá trị xám trung bình của các điểm ảnh nằm trong phạm vi của ma trận 3×3 hoặc 5×5 có tâm là điểm ảnh đang xét sẽ được tính toán. Nếu như độ chênh lệch mức xám giữa điểm đang xét với giá trị xám trung bình thỏa mãn lớn hơn một mức tối thiểu δ1 nào đó (PTB+ δ1< P) thì chúng ta sẽ coi nó là điểm biên và ghi nhận lại, còn các điểm không thỏa mãn điều kiện trên sẽ được coi là điểm nền.



a) Ma trận điểm ảnh trước khi lọc b) Ma trận điểm ảnh sau khi lọc

***Hình 3.4.* Ma trận điểm ảnh trước và sau lọc**

Thuật toán có thể được mô tả như sau:

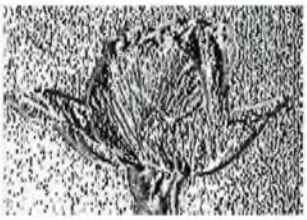
```
for (i=0; i< biHeight; i++)
for (j=0; j< biWidth; j++)
{
    tt_GrayScale=0;
    for (ii=i-1; ii<=i+1; ii++)
    for (jj=j-1; jj<=j+1; jj++)
       tt_GrayScale+=GetPoint(pOrgImg,ii,jj);
    if (tt_GrayScale>9*GetPoint(pOrgImg,i,j)+δ1)
       SetPoint(pBdImg,i,j,BLACK);
}
```

Trong đó:

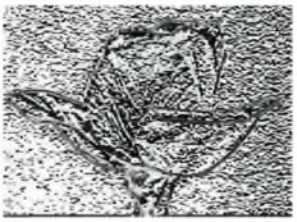
- biWidth, biHeight: là chiều rộng và chiều cao của ảnh tính theo đơn vị Pixel.
- pOrgImg, pBdImg: lần lượt là các con trỏ trỏ đến các vùng dữ liệu của ảnh gốc và ảnh biên.
- tt_GrayScale: là tổng giá trị độ xám của các điểm ảnh thuộc ma trận 3×3 có tâm là điểm ảnh đang xét.
- δ_1: là độ chênh lệch mức xám của điểm ảnh đang xét so với giá trị xám trung bình của ma trận.
- SetPoint() và GetPoint(): là các hàm đọc, ghi giá trị điểm ảnh.

Chúng ta có thể so sánh được hiệu quả của thuật toán phát hiện biên này so với các thuật toán phát hiện biên truyền thống thông qua các hình minh họa dưới đây. Hình 3.5a là ảnh gốc, Hình 3.5b là ảnh biên qua lọc Sobel Hx, Hình 3.5c là ảnh biên qua lọc Sobel Hy, Hình 3.5d là ảnh biên qua lọc Kirsh, Hình 3.5e là ảnh biên qua lọc Laplace. Hình 3.6 là các ảnh biên thu được khi sử dụng thuật toán phát hiện biên đề xuất dựa vào trung bình cục bộ với giá trị δ_1 khác nhau. Hình 3.6a là ảnh biên thu được với δ_1= 25, Hình 3.6b là ảnh biên thu được với δ1= 250.

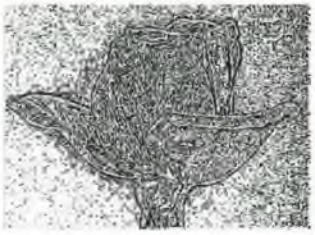
a) Ảnh gốc



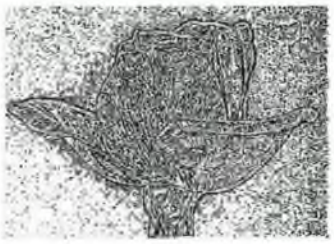
b) Ảnh qua lọc Sobel Hx



c) Ảnh qua lọc Sobel Hy

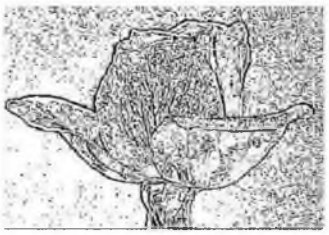


d) Ảnh qua lọc Kirsh

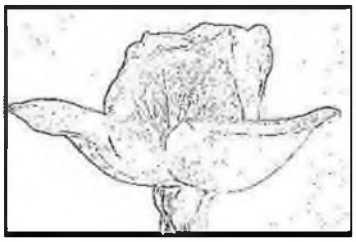


e) Ảnh qua lọc Laplace

***Hình 3.5.* Các ảnh biên theo các thuật toán phát hiện biên truyền thống**



a) Ảnh biên thu được với δ_1= 25



b) Ảnh biên thu được với δ_1= 250

***Hình 3.6.* Các ảnh biên kết quả thu được theo thuật toán đề xuất**

Nhận xét:

Chúng ta có nhận xét là ảnh gốc sử dụng trong chương trình có mầu nền khá tối và có rất nhiều nhiễu. Các bộ lọc sử dụng trong minh họa trên đều mắc phải vấn đề này. Thuật toán dò biên sử dụng trong chương trình tuy đã hạn chế được nhiều nhiễu so với việc sử dụng các bộ lọc và làm nổi rõ các đường biên nhưng vẫn không loại bỏ được hầu hết các nhiễu. Khi áp dụng thuật toán trên chúng ta vẫn có thể làm giảm bớt nhiễu đi nhiều hơn nữa bằng cách tăng giá trị của hệ số delta lên. Nhưng khi đó các đường biên thu được cũng bị đứt đoạn và mờ đi nhiều.

Thuật toán có độ phức tạp tỷ lệ với kích thước ảnh và kích thước cửa sổ. Với độ phức tạp của thuật toán là O(n2) nên nó thực hiện việc tìm biên khá nhanh, ảnh biên thu được chỉ gồm các điểm ảnh và điểm biên nên dễ xử lý, bản thân thuật toán này cũng ít chịu ảnh hưởng của nhiễu hơn là kỹ thuật Sobel mặc dù nó có khả năng phát hiện khá tốt các vùng biên

nhiễu. Nhưng cũng giống các phương pháp phát hiện biên trực tiếp khác là nó cho kết quả đường biên có độ dày không đều.

3.5. PHÁT HIỆN BIÊN DỰA VÀO CÁC PHÉP TOÁN HÌNH THÁI

3.5.1. Xấp xỉ trên và xấp xỉ dưới đối tượng ảnh

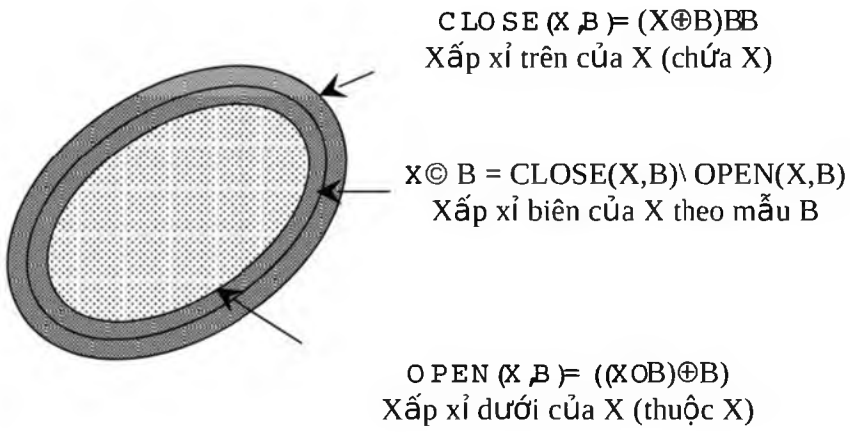
Biên là vấn đề quan trọng trong xử lý ảnh và nhận dạng, vì các đặc điểm trích chọn trong quá trình nhận dạng chủ yếu dựa vào biên. Trong thực tế người ta thường dùng hai phương pháp pháp hiện biên cơ bản là: Phát hiện biên trực tiếp và gián tiếp. Phần này đề cập đến một tiếp cận mới trong phát hiện biên dựa vào các phép toán hình thái thông qua các kỹ thuật xấp xỉ trên và xấp xỉ dưới đối tượng.

Trong [2,7] cũng đã đề cập đến kỹ thuật phát hiện biên dựa vào phép toán hình thái. Nhưng các kỹ thuật phát hiện biên trực tiếp, gián tiếp và dựa vào các phép toán hình thái kể trên đều xuất phát từ quan điểm biên của đối tượng là một tập hợp con của đối tượng. Trong thực tế chúng ta thường hiểu đường biên là khu vực ranh giới bao gồm cả hai phần thuộc đối tượng và không thuộc đối tượng. Ở phần dưới đây, chúng tôi đề xuất một kỹ thuật phát hiện biên dựa vào phép toán hình thái theo quan niệm này, xuất phát từ cơ sở định lý 2.1 đã được chứng minh ở trên.

Biên (hay đường biên) có thể hiểu đơn giản là các đường bao của các đối tượng trong ảnh chính là ranh giới giữa đối tượng và nền. Việc xem ranh giới là phần được tạo lập bởi các điểm thuộc đối tượng và thuộc nền cho phép ta xác định biên dựa trên các phép toán hình thái [2,7].

Theo định lý 2.1 ta có: $(X \oplus B) \ominus B \supseteq X \ \forall B$

Như vậy, tập CLOSE(X,B) = $(X \oplus B) \ominus B$ có thể được xem như là xấp xỉ trên của tập X theo mẫu B (Hình 3.7).



Hình 3.7. Xấp xỉ trên và dưới theo mẫu B của X

Cũng theo định lý 2.1 ta có, $(X \ominus B) \oplus B \subseteq X \quad \forall B$

Do vậy, tập OPEN(X,B) = (X⊖B)⊕B có thể được xem như là xấp xỉ dưới của tập X theo mẫu B.

Từ đó, tập CLOSE(X,B)\ OPEN(X,B) có thể được xem như là xấp xỉ biên của tập X theo mẫu và quá trình xấp xỉ biên của X theo mẫu B kí hiệu là X© B.

Để tăng độ chính xác, người ta thường xem B là dãy các phần tử cấu trúc.

$B = \{B_i, 1 \le i \le n\}$

Và xấp xỉ biên của X theo tập cấu trúc B được xác định:

$$X \copyright B = \bigcup_{i=1}^{n} (X \copyright B_i)$$

3.5.1. Thuật toán phát hiện biên dựa vào phép toán hình thái

Vào : Ảnh X và dãy mẫu $B = \{B_i, 1 \le i \le n\}$;

Ra : Biên của đối tượng theo mẫu B

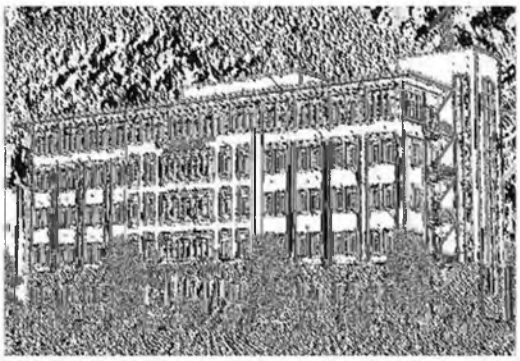
Phương pháp:

Bước 1: Tính $X \copyright B_i \; \forall i=1,n$

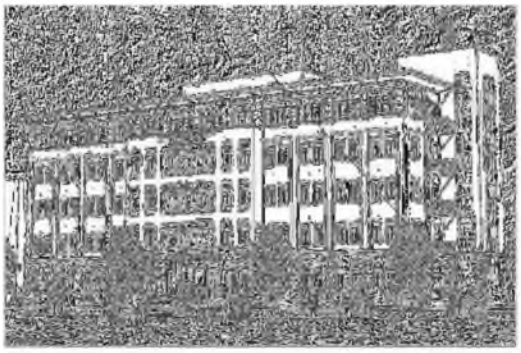
Bước 2: Tính $\bigcup_{i=1}^{n} (X \copyright B_i)$

Trong Hình 3.8a dưới đây là ảnh gốc với 256 mức xám, Hình 3.8b là ảnh biên thu được qua phát hiện biên bằng Sobel, Hình 3.8c là ảnh biên thu được qua phát hiện biên bằng Laplace. Hình 3.8d là ảnh biên kết quả thực hiện bởi thuật toán phát hiện biên bằng các phép toán hình thái với ngưỡng tách θ = 128 và các mẫu tách biên B_i là:

B_1 B_2 B_3 B_4



a) Ảnh gốc đa cấp xám



b) Ảnh biên thu được qua Sobel

c) Ảnh biên thu được qua Laplace

d) Ảnh biên kết quả dựa vào phép toán hình thái

Hình 3.8. Phát hiện biên bởi thuật toán dựa vào phép toán hình thái

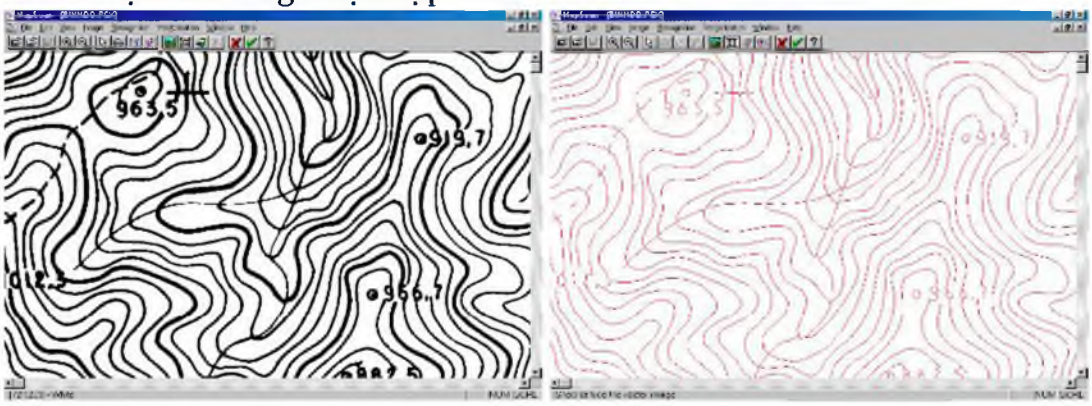
Chương 4:

XƯƠNG VÀ CÁC KỸ THUẬT TÌM XƯƠNG

4.1. GIỚI THIỆU

Xương được coi như hình dạng cơ bản của một đối tượng, với số ít các điểm ảnh cơ bản. Ta có thể lấy được các thông tin về hình dạng nguyên bản của một đối tượng thông qua xương.

Một định nghĩa xúc tích về xương dựa trên tính continuum (tương tự như hiện tượng cháy đồng cỏ) được đưa ra bởi Blum (1976) như sau: Giả thiết rằng đối tượng là đồng nhất được phủ bởi cỏ khô và sau đó dựng lên một vòng biên lửa. Xương được định nghĩa như nơi gặp của các vệt lửa và tại đó chúng được dập tắt.



a) Ảnh gốc b) Ảnh xương

***Hình 4.1.* Ví dụ về ảnh và xương**

Kỹ thuật tìm xương luôn là chủ đề nghiên cứu trong xử lý ảnh những năm gần đây. Mặc dù có những nỗ lực cho việc phát triển các thuật toán tìm xương, nhưng các phương pháp được đưa ra đều bị mất mát thông tin. Có thể chia thành hai loại thuật toán tìm xương cơ bản:

- Các thuật toán tìm xương dựa trên làm mảnh
- Các thuật toán tìm xương không dựa trên làm mảnh

4.2. TÌM XƯƠNG DỰA TRÊN LÀM MẢNH

4.2.1. Sơ lược về thuật toán làm mảnh

Thuật toán làm mảnh ảnh số nhị phân là một trong các thuật toán quan trọng trong xử lý ảnh và nhận dạng. Xương chứa những thông tin bất biến về cấu trúc của ảnh, giúp cho quá trình nhận dạng hoặc vectơ hoá sau này.

Thuật toán làm mảnh là quá trình lặp duyệt và kiểm tra tất cả các điểm thuộc đối tượng. Trong mỗi lần lặp tất cả các điểm của đối tượng sẽ được kiểm tra: nếu như chúng thoả mãn điều kiện xoá nào đó tuỳ thuộc vào mỗi thuật toán thì nó sẽ bị xoá đi. Quá trình cứ lặp lại cho đến khi không còn điểm biên nào được xoá. Đối tượng được bóc dần lớp biên cho đến khi nào bị thu mảnh lại chỉ còn các điểm biên.

Các thuật toán làm mảnh được phân loại dựa trên phương pháp xử lý các điểm là thuật toán làm mảnh song song và thuật toán làm mảnh tuần tự.

Thuật toán làm mảnh song song, là thuật toán mà trong đó các điểm được xử lý theo phương pháp song song, tức là được xử lý cùng một lúc. Giá trị của mỗi điểm sau một lần lặp chỉ phụ thuộc vào giá trị của các láng giềng bên cạnh (thường là 8-láng giềng) mà giá trị của các điểm này đã được xác định trong lần lặp trước đó. Trong máy có nhiều bộ vi xử lý mỗi vi xử lý sẽ xử lý một vùng của đối tượng, nó có quyền đọc từ các điểm ở vùng khác nhưng chỉ được ghi trên vùng của nó xử lý.

Trong thuật toán làm mảnh tuần tự các điểm thuộc đối tượng sẽ được kiểm tra theo một thứ tự nào đó (chẳng hạn các điểm được xét từ trái qua phải, từ trên xuống dưới). Giá trị của điểm sau mỗi lần lặp không những phụ thuộc vào giá trị của các láng giềng bên cạnh mà còn phụ thuộc vào các điểm đã được xét trước đó trong chính lần lặp đang xét.

Chất lượng của thuật toán làm mảnh được đánh giá theo các tiêu chuẩn được liệt kê dưới đây nhưng không nhất thiết phải thoả mãn đồng thời tất cả các tiêu chuẩn.

- Bảo toàn tính liên thông của đối tượng và phần bù của đối tượng
- Sự tương hợp giữa xương và cấu trúc của ảnh đối tượng
- Bảo toàn các thành phần liên thông
- Bảo toàn các điểm cụt
- Xương chỉ gồm các điểm biên, càng mảnh càng tốt
- Bền vững đối với nhiễu
- Xương cho phép khôi phục ảnh ban đầu của đối tượng
- Xương thu được ở chính giữa đường nét của đối tượng được làm mảnh
- Xương nhận được bất biến với phép quay.

4.2.2. Một số thuật toán làm mảnh

Trong phần này điểm qua một số đặc điểm, ưu và khuyết điểm của các thuật toán đã được nghiên cứu.

1º. Thuật toán làm mảnh cổ điển là thuật toán song song, tạo ra xương 8 liên thông, tuy nhiên nó rất chậm, gây đứt nét, xoá hoàn toàn một số cấu hình nhỏ.

2º. Thuật toán làm mảnh của Toumazet bảo toàn tất cả các điểm cụt không gây đứt nét đối tượng. Tuy nhiên, thuật toán có nhược điểm là rất chậm, rất nhạy cảm với nhiễu, xương chỉ là 4-liên thông và không làm mảnh được với một số cấu hình phức tạp

3º. Thuật toán làm mảnh của Y.Xia dựa trên đường biên của đối tượng, có thể cài đặt theo cả phương pháp song song và tuần tự. Tốc độ của thuật toán rất nhanh. Nó có nhược điểm là gây đứt nét, xương tạo ra là xương giả (có độ dày là 2 phần tử ảnh).

4º. Thuật toán làm mảnh của N.J.Naccache và R.Shinghal. Thuật toán có ưu điểm là nhanh, xương tạo ra có khả năng khôi phục ảnh ban đầu của đối tượng. Nhược điểm chính của thuật toán là rất nhạy với nhiễu, xương nhận được phản ánh cấu trúc của đối tượng thấp.

5º. Thuật toán làm mảnh của H.E.Lu P.S.P Wang tương đối nhanh, giữ được tính liên thông của ảnh, nhưng lại có nhược điểm là xương tạo ra là xương 4-liên thông và xoá mất một số cấu hình nhỏ.

6º. Thuật toán làm mảnh của P.S.P Wang và Y.Y.Zhang dựa trên đường biên của đối tượng, có thể cài đặt theo phương pháp song song hoặc tuần tự, xương là 8-liên thông, ít chịu ảnh hưởng của nhiễu. Nhược điểm chính của thuật toán là tốc độ chậm.

7º. Thuật toán làm mảnh song song thuần tuý nhanh nhất trong các thuật toán trên, bảo toàn tính liên thông, ít chịu ảnh hưởng của nhiễu. Nhược điểm là xoá hoàn toàn một số cấu hình nhỏ, xương tạo ra là xương 4-liên thông.

4.3. TÌM XƯƠNG KHÔNG DỰA TRÊN LÀM MẢNH

Để tách được xương của đối tượng có thể sử dụng đường biên của đối tượng. Với điểm p bất kỳ trên đối tượng, ta bao nó bởi một đường biên. Nếu như có nhiều điểm biên có cùng khoảng cách ngắn nhất tới p thì p nằm trên trục trung vị. Tập tất cả các điểm như vậy lập thành trục trung vị hay xương của đối tượng. Việc xác định xương được tiến hành thông qua
hai bước:

- **Bước thứ nhất**, tính khoảng cách từ mỗi điểm ảnh của đối tượng đến điểm biên gần nhất. Như vậy cần phải tính toán khoảng cách tới tất cả các điểm biên của ảnh.
- **Bước thứ hai,** khoảng cách ảnh đã được tính toán và các điểm ảnh có giá trị lớn nhất được xem là nằm trên xương của đối tượng.

4.3.1. Khái quát về lược đồ Voronoi

Lược đồ Voronoi là một công cụ hiệu quả trong hình học tính toán. Cho hai điểm P_i, P_j là hai phần tử của tập Ω gồm n điểm trong mặt phẳng. Tập các điểm trong mặt phẳng gần P_i hơn P_j là nửa mặt phẳng $H(P_i, P_j)$ chứa điểm P_i và bị giới hạn bởi đường trung trực của đoạn thẳng P_iP_j. Do đó, tập các điểm gần P_i hơn bất kỳ điểm P_j nào có thể thu được bằng cách giao n-1 các nửa mặt phẳng $H(P_i, P_j)$:

$$V(P_i) = \cap H(P_i, P_j)\ i \neq j\ (i= 1,...,n) \quad (4.1)$$

Định nghĩa 4.1 [Đa giác/Sơ đồ Voronoi]

Sơ đồ Voronoi của Ω là hợp của tất cả các $V(P_i)$

$$Vor(\Omega) = \cup V(P_i)\ P_i \in \Omega \text{ (là một đa giác)} \quad (4.2)$$

Định nghĩa 4.2 [Đa giác Voronoi tổng quát]

Cho tập các điểm Ω, đa giác Voronoi của tập con U của Ω được định nghĩa như sau:

$$V(U) = \{P| \exists v \in U, \forall w \in \Omega \setminus U : d(P,v) < d(P,w)\}$$

$$= \cup V(P_i)\ P_i \in U \quad (4.3)$$

4.3.2. Trục trung vị Voronoi rời rạc

Định nghĩa 4.3 [Bản đồ khoảng cách - Distance Map]

Cho đối tượng S, đối với mỗi $(x, y) \in S$, ta tính giá trị khoảng cách map(x, y) với hàm khoảng cách d(.,.) như sau:

$$\forall (x, y) \in S: map(x, y) = \min_i d[(x, y), (x_i, y_i)] \quad (4.4)$$

trong đó $(x_i, y_i) \in B(S)$ - tập các điểm biên của S

Tập tất cả các map(x, y), kí hiệu là DM(S), được gọi là bản đồ khoảng cách của S.

Chú ý: Nếu hàm khoảng cách d(.,.) là khoảng cách Euclide, thì phương trình (4.4) chính là khoảng cách ngắn nhất từ một điểm bên trong đối tượng tới biên. Do đó, bản đồ khoảng cách được gọi là bản đồ khoảng cách Euclide EDM(S) của S. Định nghĩa trên được dùng cho cả hình rời rạc lẫn liên tục.

Định nghĩa 4.4 [Tập các điểm biên sinh]

Cho map(x, y) là khoảng cách ngắn nhất từ (x, y) đến biên (theo định nghĩa 4.3). Ta định nghĩa: $map^{-1}(x, y) = \{p| \ p \in B(S), d(p, (x, y)):=map(x, y)\}$

Khi đó tập các điểm biên sinh $\hat{}B(S)$ được định nghĩa bởi:

$$\hat{}B(S) = \cup map^{-1}(x, y), (x, y) \in S \quad (4.5)$$

Do S có thể chứa các đường biên rời nhau, nên $\hat{}B(S)$ bao gồm nhiều tập con, mỗi tập mô tả một đường biên phân biệt:

$$\hat{}B(S)=\{B_1(S),..B_N(S)\} \quad (4.6)$$

Định nghĩa 4.5 [Trục trung vị Voronoi rời rạc (DVMA)]

Trục trung vị Voronoi rời rạc được định nghĩa là kết quả của sơ đồ Voronoi bậc nhất rời rạc của tập các điểm biên sinh giao với hình sinh S :

$$DVMA(\hat{}B(S)) = Vor(\hat{}B(S)) \cap S \quad (4.7)$$

4.3.3. Xương Voronoi rời rạc

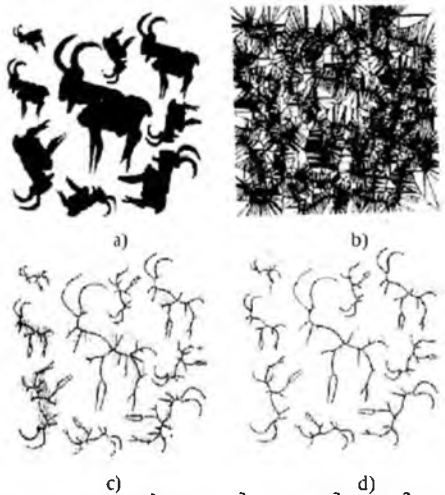
Định nghĩa 4.6 [Xương Voronoi rời rạc - DiscreteVoronoi Skeleton]

Xương Voronoi rời rạc theo ngưỡng T, kí hiệu là $Ske^{DVMA}(\hat{}B(S),T)$ (hoặc $Ske(\hat{}B(S),T)$) là một tập con của trục trung vị Voronoi:

$$Ske^{DVMA}(\hat{}B(S),T)= \{(x,y)| \ (x,y) \in DVMA(\hat{}B(S)), \Psi(x,y) > T\} \quad (4.8)$$

Ψ: là hàm hiệu chỉnh.

Dễ thấy nếu ngưỡng T càng lớn thì càng thì số lượng điểm tham gia trong xương Vonoroi càng ít (Hình 4.2).



Hình 4.2. **Xương Voronoi rời rạc ảnh hưởng của các hàm hiệu chỉnh khác nhau.**

(a) Ảnh nhị phân. (b) Sơ đồ Voronoi. (c) Hiệu chỉnh bởi hàm Potential, T=9.0. (d) Hiệu chỉnh bởi hàm Potential, T=18.0

4.3.4. Thuật toán tìm xương

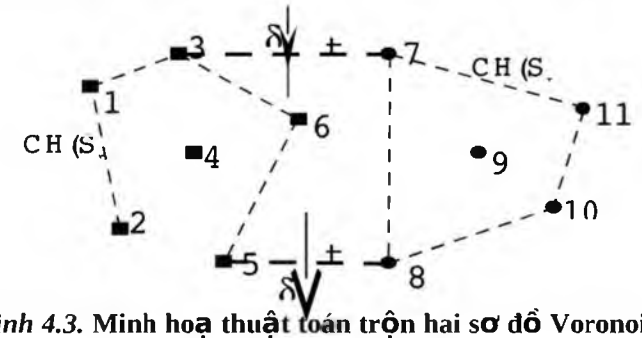
Trong mục này sẽ trình bày ý tưởng cơ bản của thuật toán tìm xương và mô tả bằng ngôn ngữ tựa Pascal.

Tăng trưởng: Việc tính toán sơ đồ Voronoi được bắt đầu từ một điểm sinh trong mặt phẳng. Sau đó điểm sinh thứ hai được thêm vào và quá trình tính toán tiếp tục với đa giác Voronoi đã tìm được với điểm vừa được thêm vào đó. Cứ như thế, quá trình tính toán sơ đồ Voronoi được thực hiện cho đến khi không còn điểm sinh nào được thêm vào. Nhược điểm của chiến lược này là mỗi khi một điểm mới được thêm vào, nó có thể gây ra sự phân vùng toàn bộ các đa giác Voronoi đã được tính.

Chia để trị: Tập các điểm biên đầu tiên được chia thành hai tập điểm có kích cỡ bằng nhau. Sau đó thuật toán tính toán sơ đồ Voronoi cho cả hai tập con điểm biên đó. Cuối cùng, người ta thực hiện việc ghép cả hai sơ đồ Voronoi trên để thu được kết quả mong muốn. Tuy nhiên, việc chia tập các điểm biên thành hai phần không phải được thực hiện một lần, mà được lặp lại nhiều lần cho đến khi việc tính toán sơ đồ Voronoi trở nên đơn giản. Vì thế, việc tính sơ đồ Voronoi trở thành vấn đề làm thế nào để trộn hai sơ đồ Voronoi lại với nhau.

Thuật toán sẽ trình bày ở đây là sự kết hợp của hai ý tưởng ở trên. Tuy nhiên, nó sẽ mang nhiều dáng dấp của thuật toán chia để trị.

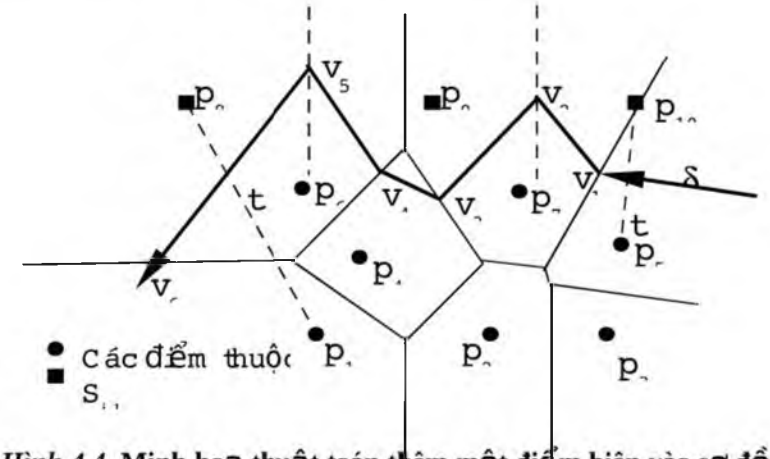
Hình 4.3 minh hoạ ý tưởng của thuật toán này. Mười một điểm biên được chia thành hai phần (bên trái: 1- 6, bên phải: 7-11) bởi đường gấp khúc δ, và hai sơ đồ Voronoi tương ứng $Vor(S_L)$ và $Vor(S_R)$. Để thu được sơ đồ Vornonoi $Vor(S_L \cup S_R)$, ta thực hiện việc trộn hai sơ đồ trên và xác định lại một số đa giác sẽ bị sửa đổi do ảnh hưởng của các điểm bên cạnh thuộc sơ đồ kia. Mỗi phần tử của δ sẽ là một bộ phận của đường trung trực nối hai điểm mà một điểm thuộc $Vor(S_L)$ và một thuộc $Vor(S_R)$. Trước khi xây dựng δ, ta tìm ra phần tử đầu và cuối của nó. Nhìn vào hình trên, ta nhận thấy rằng cạnh δ_1 và δ_5 là các tia. Dễ nhận thấy rằng việc tìm ra các cạnh đầu và cuối của δ trở thành việc tìm cạnh vào t_α và cạnh ra t_ω.



Hình 4.3. **Minh hoạ thuật toán trộn hai sơ đồ Voronoi**

Sau khi đã tìm được t_α và t_ω, các điểm cuối của t_α được sử dụng để xây dựng phần tử đầu tiên của δ (δ_1 trong hình trên). Sau đó thuật toán tìm điểm giao của δ với Vor(S_L) và Vor(S_R). Trong ví dụ trên, δ đầu tiên giao với V(3). Kể từ đây, các điểm nằm trên phần kéo dài δ sẽ gần điểm 6 hơn điểm 3. Do đó, phần tử tiếp theo δ_2 của δ sẽ thuộc vào đường trung trực của điểm 6 và điểm 7. Sau đó điểm giao tiếp theo của δ sẽ thuộc và Vor(S_L); δ bây giờ sẽ đi vào V(9) và δ_2 sẽ được thay thế bởi δ_3. Quá trình này sẽ kết thúc khi δ gặp phần tử cuối δ_5.

Trên đây chỉ là minh hoạ cho thuật trộn hai sơ đồ Voronoi trong chiến lược chia để trị. Tuy nhiên, trong thuật toán sẽ trình bày ở đây thì sự thực hiện có khác một chút. Tập các điểm ảnh không phải được đưa vào ngay từ đầu mà sẽ được quét vào từng dòng một. Giả sử tại bước thứ i, ta đã thu được một sơ đồ Voronoi gồm i-1 hàng các điểm sinh Vor(S_{i-1}). Tiếp theo, ta quét lấy một hàng L_i các điểm ảnh từ tập các điểm biên còn lại. Thực hiện việc tính sơ đồ Voronoi Vor(L_i) cho hàng này, sau đó trộn Vor(S_{i-1}) với Vor(L_i). Kết quả ta sẽ được một sơ đồ mới, và lại thực hiện việc quét hàng L_{i+1} các điểm sinh còn lại v.v.. Quá trình này sẽ kết thúc khi không còn điểm biên nào để thêm vào sơ đồ Voronoi. Do Vor(L_i) sẽ có dạng răng lược (nếu L_i có k điểm thì Vor(L_i) sẽ gồm k-1 đường thẳng đứng), nên việc trộn Vor(S_{i-1}) với Vor(L_i) có phần đơn giản hơn.



Hình 4.4. **Minh hoạ thuật toán thêm một điểm biên vào sơ đồ Voronoi**

Giải thuật trên có thể được mô tả bằng ngôn ngữ tựa Pascal như sau:

Procedure VORONOI

(*S_i: Tập các điểm của i dòng quét đầu tiên,

0 <= i <=i_{MAX},

Vor(S_i) sơ đồ Vorronoi của S_i *)

Begin

i:=0; S_i:=rỗng;

While ($i<i_{max} \wedge S_i \subset$ straight_line) **do**

Begin

(*Khởi tạo sơ đồ Voronoi cho đến khi nó chứa ít nhất một đỉnh*)

increment i;

GetScanLine L_i;

$Vor(S_i)$ = VoroPreScan($Vor(S_{i-1}, L_i)$);

End

While ($i < i_{max}$) **do**

Begin

Increment i;

GetScanLine L_i;

$Vor(L_i)$:= các đường trung trực sinh bởi các điểm sinh thuộc L_i

$Vor(S_i)$:= VoroLink($Vor(S_{i-1})$, $Vor(L_i)$);

End

End.

Giả sử xét trên hệ toạ độ thực. Ảnh vào được quét từ dưới lên. Toạ độ y (biến i) tương ứng với từng dòng quét được tăng dần theo từng dòng. Trong thủ tục trên, hàm quan trọng nhất là hàm VoroLink, hàm này thực hiện việc trộn sơ đồ Voronoi của L_{i-1} dòng đã được quét trước đó với sơ đồ Voronoi của dòng hiện tại thứ i. Trong vòng lặp trên, hàm VoroPreScan là một biến thể của hàm VoroLink, có nhiệm vụ khởi tạo sơ đồ Voronoi và thoát khỏi vòng lặp ngay khi nó thành lập được sơ đồ Voronoi chứa ít nhất một đỉnh. Hàm VoroLink thực hiện việc trộn hai sơ đồ Voronoi $Vor(S_{i-1})$ và $Vor(L_i)$ với nhau để thành $Vor(S_i)$.

Chương 5:

CÁC KỸ THUẬT HẬU XỬ LÝ

5.1. RÚT GỌN SỐ LƯỢNG ĐIỂM BIỂU DIỄN

5.1.1. Giới thiệu

Rút gọn số lượng điểm biểu diễn là kỹ thuật thuộc phần hậu xử lý. Kết quả của phần dò biên hay trích xương thu được 1 dãy các điểm liên tiếp. Vấn đề đặt ra là hiệu có thể bá bớt các điểm thu được để giảm thiểu không quan lưu trữ và thuận tiện cho việc đối sách hay không.

Bài toán:

Cho đường cong gồm n điểm trong mặt phẳng (x_1, y_1), (x_2, y_2)… (x_n,y_n). Hãy bỏ bớt 1 số điểm thuộc đường cong sao cho đường cong mới nhận được là $(X_{i1}; Y_{i1})$, $(X_{i2}; Y_{i2})$… $(X_{im}; Y_{im})$ "gần giống" với đường cong ban đầu.

*** Một số độ đo "gần giống"**

+ Chiều dài (chiều rộng) của hình chữ nhật nhá nhất chứa đường cong

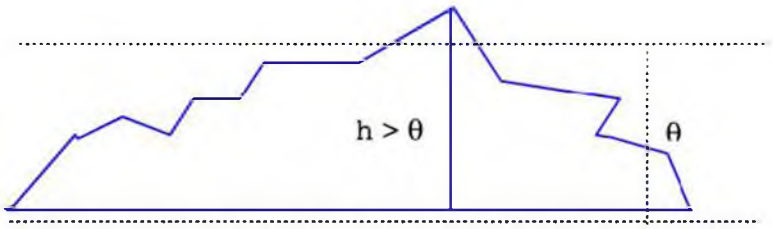
+ Khoảng cách lớn nhất từ đường cong đến đoạn thẳng nối 2 đầu mót của đường cong

+ Tỷ lệ giữa chiều dài và chiều rộng của hình chữ nhật nhá nhất chứa đường con

+ Số lần đường cong cắt đoạn thẳng nối 2 đầu mót

5.1.2. Thuật toán Douglas Peucker

5.1.2.1. Ý tưởng



Hình 5.1. Đơn giản hóa đường công theo thuật toán Douglas Peucker

Ý tưởng cơ bản của thuật toán Douglas-Peucker là xét xem khoảng cách lớn nhất từ đường cong tới đoạn thẳng nối hai đầu mút đường cong (xem Hình 5.1) có lớn hơn ngưỡng θ không. Nếu điều này đúng thì điểm xa nhất được giữ lại làm điểm chia đường cong và thuật toán được thực hiện

tương tự với hai đường cong vừa tìm được. Trong trường hợp ngược lại, kết quả của thuật toán đơn giản hoá là hai điểm đầu mút của đường cong.

Thuật toán Douglas-Peucker:

- Bước 1: Chọn ngưỡng θ.
- Bước 2: Tìm khoảng cách lớn nhất từ đường cong tới đoạn thẳng nối hai đầu đoạn đường cong h.
- Bước 3: Nếu h ≤ θ thì dừng.
- Bước 4: Nếu h > θ thì giữ lại điểm đạt cực đại này và quay trở lại bước 1.

Nhận xét: Thuật toán này tỏ ra thuận lợi đối với các đường cong thu nhận được mà gốc là các đoạn thẳng, phù hợp với việc đơn giản hoá trong quá trình vectơ các bản vẽ kỹ thuật, sơ đồ thiết kế mạch in v.v..

5.1.2.2. Chương trình

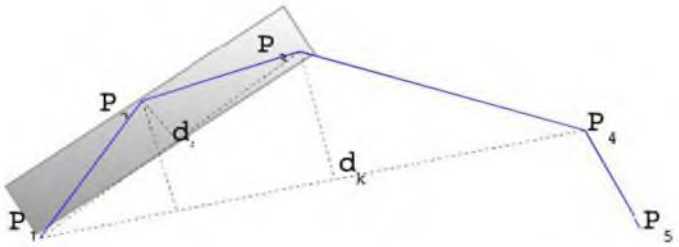
```
//Hàm tính đường cao từ dinh đến đoạn thẳng nối hai điểm dau, cuoi
float Tinhduongcao (POINT dau, POINT cuoi, POINT dinh)
{
   floot h;
   || tính đường cao
   returm h ;
}
//Hàm đệ quy nhằm đánh dấu loại bỏ các điểm trong đường cong
void DPSimple(POINT *pLINE,int dau,int cuoi,BOOL *chiso,float θ)
{
   int      i, index = dau;
   float   h, hmax = 0;
   for(i = dau + 1; i < cuoi; i++)
   {
      h= Tinhduongcao(pLINE[dau], pLINE[cuoi]; pLINE[i]);
      if(h > hmax)
      {
         hmax = h;
         index = i;
```

```
        }
    }
    if(hmax ≤ θ)
        for(i= dau + 1; i < cuoi, i++)
            chiso[i] = FALSE;
    else
    {
        DPSimple(PLINE, dau, index, chiso, θ);
        DPSimple(PLINE, index, cuoi, chiso, θ) ;
    }
}
//Hàm rút gọn số lượng điểm DouglasPeucker
int DouglasPeucker(POINT *pLINE, int n, float θ)
{
    int       i, j;
    BOOL  chiso [MAX_PT];
    for(i = 0; i < m; i++)  //Tất cả các điểm được giữ lại
        chiso[i] = TRUE;
    DPSimple(pLINE, 0, n – 1, chiso, θ);
    for(i = j = 0; i < n; i ++)
        if (chiso [i] ==TRUE)
            pLINE[j++] = pLINE[i];
    return j;
}
```

5.1.3. Thuật toán Band width

5.1.3.1. Ý tưởng

Trong thuật toán Band Width, ta hình dung có một dải băng di chuyển từ đầu mút đường cong dọc theo đường cong sao cho đường cong nằm trong di băng đó cho đến khi có điểm thuộc đường cong chạm vào biên của dải băng, điểm này sẽ được giữ lại. Quá trình này được thực hiện với phần còn lại của đường cong bắt đầu từ điểm vừa tìm được cho đến khi hết đường cong. Cụ thể như sau:



Hình 5.2. Đơn giản hóa đường cong với thuật toán Band Width

Bắt đầu bằng việc xác định điểm đầu tiên trên đường cong và coi đó như là một điểm chốt (P_1). Điểm thứ ba (P_3) được coi là điểm động. Điểm giữa điểm chốt và điểm động (P_2) là điểm trung gian. Ban đầu khoảng cách từ điểm trung gian đến đoạn thẳng nối điểm chốt và điểm động được tính toán và kiểm tra. Nếu khoảng cách tính được này nhỏ hơn một ngưỡng θ cho trước thì điểm trung gian có thể bỏ đi, tiến trình tiếp tục với điểm chốt là điểm chốt cũ, điểm trung gian là điểm động cũ và điểm động là điểm kế tiếp sau điểm động cũ. Trong trường hợp ngược lại, khoảng cách tính được lớn hơn ngưỡng θ cho trước thì điểm trung gian sẽ được giữ lại, tiến trình tiếp tục với điểm chốt là điển trung gian, điểm trung gian là điểm động cũ và điểm động là điểm kế tiếp sau điểm động cũ. Tiến trình được lặp cho đến hết đường cong (Hình 5.2 minh họa thuật toán Band-Width).

Thuật toán Band-Width:

- Bước 1: Xác định điểm đầu tiên trên đường cong và coi đó như là một điểm chốt (P_1). Điểm thứ ba (P_3) được coi là điểm động. Điểm giữa điểm chốt và điểm động (P_2) là điểm trung gian.
- Bước 2: Tính khoảng cách từ điểm trung gian đến đoạn thẳng nối hai điểm chốt và điểm động.
- Bước 3: Kiểm tra khoảng cách tìm được nếu nhỏ hơn một ngưỡng θ cho trước thì điểm trung gian có thể bỏ đi. Trong trường hợp ngược lại điểm chốt chuyển đến điểm trung gian.
- Bước 4: Chu trình được lặp lại thì điểm trung gian được chuyển đến điểm động và điểm kế tiếp sau điểm động được chỉ định làm điểm động mới..

Nhận xét: Thuật toán này tăng tốc độ trong trường hợp đường ống chứa nhiều điểm, điều đó có nghĩa là độ lệch giữa các điểm trong đường thẳng là nhỏ, hay độ dày nét của đường được véctơ hoá là mảnh.

5.1.3.2. Chương trình

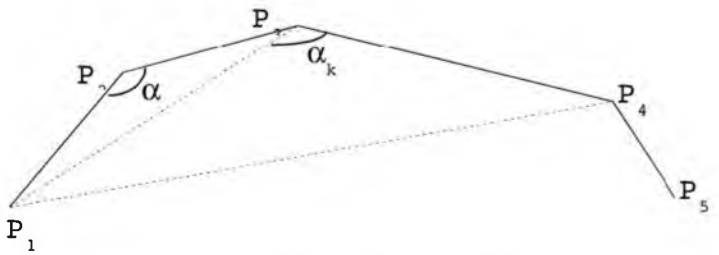
```
//Hàm tính đường cao từ đỉnh đến đoạn thẳng nối hai điểm dau, cuoi
float Tinhduongcao(POINT dau, POINT cuoi, POINT dinh)
{
   floot h;
   || tính đường cao
   returm h ;
}
//Hàm đệ quy nhằm đánh dấu loại bỏ các điểm trong đường cong
void BWSimple(POINT *pLINE, int chot, int tg, BOOL *chiso,
                    float θ, int n)
{
   if(Tinhduongcao(pLINE[chot], pLINE[tg+1], pLINE[tg]) ≤ θ)
      chiso[tg] = 0;
   else
      chot = tg;
   tg = tg + 1
if(tg < n - 1)
   BWSimple (pLINE, chot, tg, chiso, θ, n) ;
}
//Hàm rút gọn số lượng điểm BandWidth
int BandWidth(POINT *pLINE, int n, floot θ)
{
   int       i, j;
   BOOL  chiso [MAX_PT];
   for (i = 0; i < n; i++)
      chiso[i]= TRUE; //Tất cả các điểm được giữ lại
   BWSimple(pLINE, 0, 1, chiso, θ, n);
   for(i= j= 0; i < n; i++)
   if(chiso [i]== TRUE)
```

```
        pLINE [j ++1] = pLINE [i];
    return j;
}
```

5.1.4. Thuật toán Angles

5.1.4.1. Ý tưởng

Tương tự như thuật toán Band Width nhưng thay việc tính toán khoảng cách bởi tính góc. Cụ thể thuật toán bắt đầu với điểm đầu đường cong (P_1) là điểm chốt.



Hình 5.3. Đơn giản hóa đường cong với thuật toán Angles

Điểm thứ 3 của đường cong (P_3) là điểm động, điểm giữa điểm chốt và điểm động (P_2) là điểm trung gian

Góc tạo bởi điểm chốt, trung gian, động với điểm trung gian là đỉnh việc tính toán và kiểm tra

Nếu thì điểm trung gian có thể bỏ đi trong trường hợp ngược lại điểm chốt sẽ là điểm trung gian cũ và quá trình lặp với điểm trung gian là điểm động cũ, điểm động mới là điểm kế tiếp sau điểm động cũ. Tiến trình thực hiện cho đến hết đường cong.

5.1.4.2. Chương trình

```
//Hàm tính đường cao từ đỉnh đến đoạn thẳng nối hai điểm dau, cuoi
float Tinhgoc(POINT dau, POINT cuoi, POINT dinh)
{
   float θ;
   || tinhgoc (tự viết)
   return θ;
}
//Hàm đệ quy nhằm đánh dấu loại bỏ các điểm trong đường cong
void ALSimple(POINT *pLINE,int chot,int tg,BOOL *chiso,float θ,int n)
{
```

```
    if(Tinhgoc(pLINE[chot], pLINE[tg], pLINE[tg+1]) > θ)
        chiso[tg] = FALSE;
    else
        chot = tg;
    tg = tg + 1;
    if(tg < n - 1)
        ALSimple(pLINE, chot, tg, chiso, θ, n);
}
//Hàm rút gọn số lượng điểm Angles
int Angles(POINT *pLINE, int n, float θ)
{
    int i, j, chiso [MAX];
    for (i = 0; i < n; i++) //Tất cả các điểm được giữ lại
        chiso[i]= TRUE;
    ALSiple (PLINE, 0, 1 chiso, θ, n) ;
    for (i = j = 0; i < n; i++)
        if (chiso ==TRUE)
            pLINE[j++]= pLINE [i];
    return j;
}
```

**** Chú ý:***

Với θ= 0 thuật toán DouglasPeucker và BandWidth sẽ bỏ đi các điểm giữa thẳng hàng. Thuật toán Angles phải có θ= 180^o để bỏ đi các điểm giữa thẳng hàng.

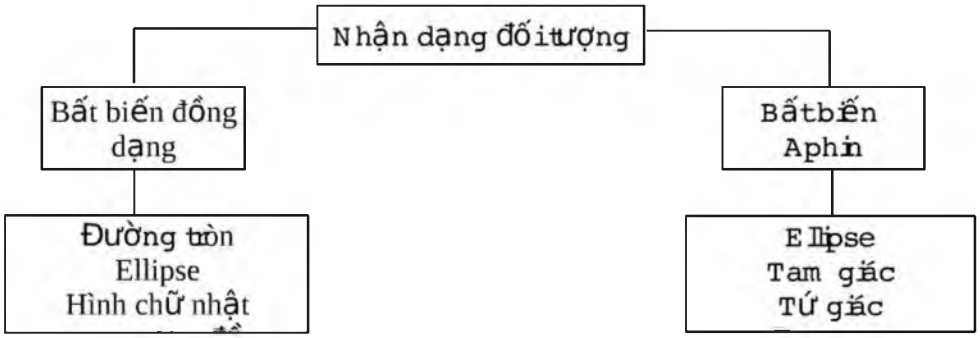
5.2. XẤP XỈ ĐA GIÁC BỞI CÁC HÌNH CƠ SỞ

Các đối tượng hình học được phát hiện thường thông qua các kỹ thuật dò biên, kết quả tìm được này là các đường biên xác định đối tượng. Đó là, một dãy các điểm liên tiếp đóng kính, sử dụng các thuật toán đơn giản hoá như Douglas Peucker, Band Width, Angle v.v.. ta sẽ thu được một polyline hay nói khác đi là thu được một đa giác xác định đối tượng dấu. Vấn đề là ta cần phải xác định xem đối tượng có phải là đối tượng cần tách hay không? Như ta đã biết một đa giác có thể có hình dạng tựa như

một hình cơ sở, có thể có nhiều cách tiếp cận xấp xỉ khác nhau. Cách xấp xỉ dựa trên các đặc trưng cơ bản sau:

Đặc trưng toàn cục: Các mô men thống kê, số đo hình học như chu vi, diện tích, tập tối ưu các hình chữ nhật phủ hay nội tiếp đa giác v.v..

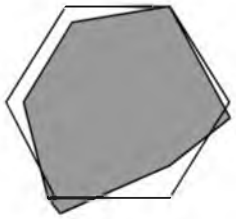
Đặc trưng địa phương: Các số đo đặc trưng của đường cong như góc, điểm lồi, lõm, uốn, cực trị v.v..



Hình 5.4. Sơ đồ phân loại các đối tượng theo bất biến

Việc xấp xỉ tỏ ra rất có hiệu quả đối với một số hình phẳng đặc biệt như tam giác, đường tròn, hình chữ nhật, hình vuông, hình ellipse, hình tròn và một đa giác mẫu.

5.2.1 Xấp xỉ đa giác theo bất biến đồng dạng



Hình 5.5. Xấp xỉ đa giác bởi một đa giác mẫu

Một đa giác với các đỉnh $V_0,..,V_{m-1}$ được xấp xỉ với đa giác mẫu $U_0,..,U_{n-1}$ với độ đo xấp xỉ như sau:

$$E(V,U) = \min_{0 \le d \le m-1} \sqrt{\frac{\Delta_d}{n}},$$

Trong đó

$$\Delta_d = \min_{0\le\theta\le 2\pi, a\in R^2} \sum_{j=0}^{n-1} \left\| kR_\theta U_j + a - V_{(j+d) \bmod m} \right\|^2 , \quad k = \sqrt{\frac{area(V_0 \cdots V_{m-1})}{area(U_0 \cdots U_{n-1})}}$$, với R_θ là phép quay quanh gốc toạ độ một góc θ.

Trong đó, Δ_d được tính hiệu quả bằng công thức sau:

$$\Delta_d = \sum_{j=0}^{n-1} |V_{(j+d) \bmod m}|^2 - \frac{1}{n} |\sum_{j=0}^{n-1} V_{(j+d) \bmod m}|^2 + k^2 \sum_{j=0}^{n-1} |U_j|^2 - 2k |\sum_{j=0}^{n-1} U_j \overline{V}_{(j+d) \bmod m}|$$

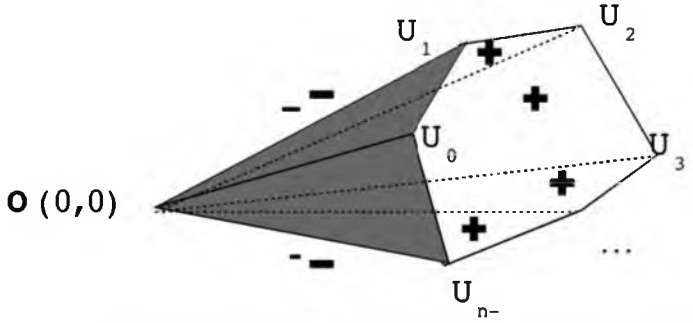
Ở đây U_j, V_j được hiểu là các số phức tại các đỉnh tương ứng. Khi m >> n thì độ phức tạp tính toán rất lớn. Với các hình đặc biệt như hình tròn, ellipse, hình chữ nhật, hình xác định duy nhất bởi tâm và một đỉnh (đa giác đều) ta có thể vận dụng các phương pháp đơn giản hơn như bình phương tối thiểu, các bất biến thống kê và hình học.

Định nghĩa 5.1

Cho đa giác Pg có các đỉnh U_0, U_1,..., U_n($U_0 \equiv U_n$) Khi đó mô men bậc p+q được xác định như sau:

$$M_{pq} = \iint_{Pg} x^p y^q dxdy .$$

Trong thực hành để tính tích phân trên người ta thường sử dụng công thức Green hoặc có thể phân tích phần bên trong đa giác thành tổng đại số của các tam giác có hướng ΔOU_iU_{i+1}.

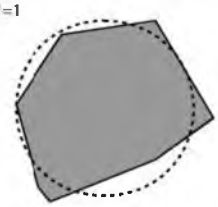


Hình 5.6. Phân tích miền đa giác thành tổng đại số các miền tam giác

5.2.1.1. Xấp xỉ đa giác bằng đường tròn

Dùng phương pháp bình phương tối thiểu, ta có độ đo xấp xỉ:

$$E(Pg,Cr)= \min_{a,b,c\in R} \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^2 + y_i^2 + ax_i + by_i + c)^2}$$



Hình 5.7. Xấp xỉ đa giác bằng đường tròn

5.2.1.2. Xấp xỉ đa giác bằng ellipse

Cũng như đối với đường tròn phương trình xấp xỉ đối với ellipse được cho bởi công thức:

$$E(Pg,El)= \min_{a,b,c,d,e\in R} \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^2 + ay_i^2 + bx_i y_i + cx_i + dy_i + e)^2}$$

Một biến thể khác của phương pháp bình phương tối thiểu khi xấp xỉ các đường cong bậc hai được đưa ra trong [7].

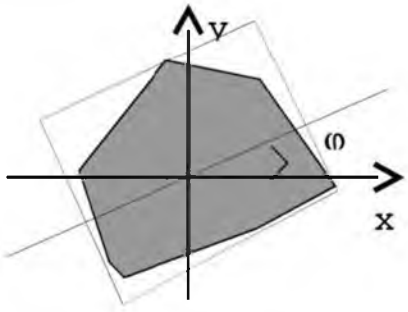
5.2.1.3. Xấp xỉ đa giác bởi hình chữ nhật

Sử dụng tính chất diện tích bất biến qua phép quay, xấp xỉ theo diện tích như sau: Gọi $\mu_{11}, \mu_{20}, \mu_{02}$ là các mô men bậc hai của đa giác (tính theo diện tích). Khi đó góc quay được tính bởi công thức sau:

$$tg2\varphi = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}.$$

Gọi diện tích của hình chữ nhật nhỏ nhất có các cạnh song song với các trục quán tính và bao quanh đa giác Pg là S.

Kí hiệu $E(Pg, Rect)= \sqrt{S - area(Pg)}$



Hình 5.8. Xấp xỉ đa giác bằng hình chữ nhật

5.2.1.4. Xấp xỉ đa giác bởi đa giác đều n cạnh

Gọi $M(x_0,y_0)$ là trọng tâm của đa giác, lấy một đỉnh **Q** tuỳ ý của đa giác, xét đa giác đều n cạnh Pg' tạo bởi đỉnh Q với tâm là M.

Kí hiệu $E(Pg, Pg')= \sqrt{|area(Pg) - area(Pg')|}$

$E(Pg, E_n)=\min E(Pg,Pg')$ khi Q chạy khắp các đỉnh của đa giác.

5.2.2 Xấp xỉ đa giác theo bất biến aphin

Trong [7] đưa ra mô hình chuẩn tắc về bất biến aphin, cho phép chúng ta có thể chuyển bài toán xấp xỉ đối tượng bởi bất biến aphin về bài toán xấp xỉ mẫu trên các dạng chuẩn tắc. Như vậy có thể đưa việc đối sánh các đối tượng với mẫu bởi các bất biến đồng dạng, chẳng hạn việc xấp xỉ bởi tam giác, hình bình hành, ellipse tương đương với xấp xỉ tam giác đều, hình vuông, hình tròn v.v... Thủ tục xấp xỉ theo bất biến aphin một đa giác với hình cơ sở được thực hiện tuần tự như sau:

+ ***Bước 0:***

Phân loại bất biến aphin các dạng hình cơ sở

Dạng hình cơ sở	Dạng chuẩn tắc
Tam giác	Tam giác đều
Hình bình hành	Hình vuông
Ellipse	Đường tròn
...	...

+ ***Bước 1***:

Tìm dạng chuẩn tắc cơ sở Pg' thoả mãn điều kiện:

$$\begin{cases} m_{01} = m_{10} = 0 & \text{(phép tịnh tiến)} \\ m_{02} = m_{20} = 1 & \text{(phép co dãn theo hai trục x, y)} \\ m_{13} = m_{31} = 0 \end{cases} \qquad (**)$$

+ ***Bước 2***:

Xác định biến đổi aphin T chuyển đa giác thành đa giác Pg ở dạng chuẩn tắc (thoả mãn tính chất (**)).

Xấp xỉ đa giác Pg với dạng chuẩn tắc cơ sở Pg' tìm được ở bước 1 với độ đo xấp xỉ E(Pg,Pg').

+ ***Bước 3***:

Kết luận, đa giác ban đầu xấp xỉ T-1(Pg') với độ đo xấp xỉ E(Pg,Pg').

Đối với bước 1 trong [7] đã đưa ra hai ví dụ sau:

Ví dụ 1:

Tồn tại duy nhất tam giác đều ΔP1P2P3 thoả mãn tính chất (**) là

P1=(0,-2α),P2= $(\sqrt{3}\alpha,\alpha)$, P3= $(-\sqrt{3}\alpha,\alpha)$, $\alpha = \frac{\sqrt[4]{2}\sqrt[8]{3}}{\sqrt{3}}$.

Ví dụ 2:

Tồn tại hai hình vuông P1P2 P3 P4 thoả mãn tính chất (**)

Hình vuông thứ nhất có 4 đỉnh tương ứng là (-p,-p),(-p,p), (p,-p),(p,p), với p=$\sqrt[4]{\frac{3}{4}}$

Hình vuông thứ hai có 4 đỉnh tương ứng là (-p,0),(p,0), (0,-p),(0,p), với p=$\sqrt[4]{3}$.

5.3. BIẾN ĐỔI HOUGH

5.3.1. Biến đổi Hongh cho đường thẳng

Bằng cách nào đó ta thu được một số điểm vấn đề đặt ra là cần phải kiểm tra xem các điểm có là đường thẳng hay không

Bài toán:

Cho n điểm (x_i; y_i) i = 1, n và ngưỡng θ hãy kiểm tra n điểm có tạo thành đường thẳng hay không?

**** Ý tưởng***

Giả sử n điểm nằm trên cùng một đường thẳng và đường thẳng có phương trình

y = ax + b

Vì (x_i, y_i) i = 1, n thuộc đường thẳng nên $y_1 = ax_1 + b$, ∀i = 1, n

$\Leftrightarrow b = - x_1a + y_1$; ∀i = 1, n

Như vậy, mỗi điểm (x_i; y_i) trong mặt phẳng sẽ tương ứng với một số đường thẳng $b = - x_ia + y_i$ trong mặt phẳng tham số a, b. n điểm (x_i; y_i) i = 1, n thuộc đường thẳng trong mặt phẳng tương ứng với n đường thẳng trong mặt phẳng tham số a, b giao nhau tại 1 điểm và điểm giao chính là a, b. Chính là hệ số xác định phương trình của đường thẳng mà các điểm nằm vào.

**** Phương pháp:***

- Xây dựng mảng chỉ số [a, b] và gán giá trị 0 ban đầu cho tất cả các phân tử của mảng

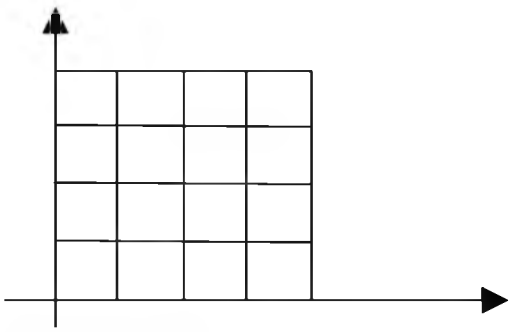
- Với mỗi $(x_i; y_i)$ và ∀a, b là chỉ số của phần tử mảng thoả mãn $b = - x_i a + y_i$ tăng giá trị của phân tử mảng tương ứng lên 1

- Tìm phần tử mảng có giá trị lớn nhất nếu giá trị lớn nhất tìm được so với số phân tử lớn hơn hoặc bằng ngưỡng θ cho trước thì ta có thể kết luận các điểm nằm trên cùng 1 đường thẳng và đường thẳng có phương trình
$y = ax + b$ trong đó a, b tương ứng là chỉ số của phần tử mảng có giá trị lớn nhất tìm được:

Ví dụ:

Cho 5 điểm (0, 1); (1, 3); (2, 5); (3, 5); (4, 9) và θ = 80%. Hãy kiểm tra xem 5 điểm đã cho có nằm trên cùng một đường thẳng hay không? Hãy cho biết phương trình đường thẳng nếu có?

- Lập bảng chỉ số [a, b] và gán giá trị 0



+ (0, 1): b = 1

+ (1, 3): b = -a + 3

+ (2, 5): b = -2a + 5

+ (3, 5): b = -3a + 5

+ (4, 9): b = -4a + 9

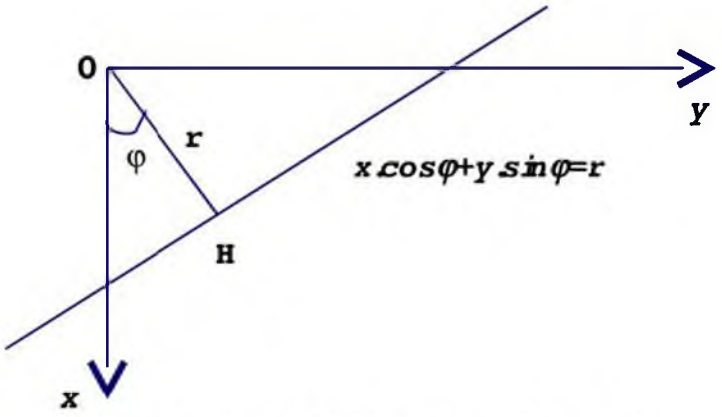
- Tìm phần tử lớn nhất có giá trị 4

4/5 = 80%

- Kết luận: 5 điểm này nằm trên cùng 1 đường thẳng

Phương trình: y = 2x + 1

5.3.2. Biến đổi Hough cho đường thẳng trong tọa độ cực



Hình 5.9. Đường thẳng Hough trong toạ độ

Mỗi điểm (x,y) trong mặt phẳng được biểu diễn bởi cặp (r,φ) trong tọa độ cực.

Tương tự mỗi đường thẳng trong mặt phẳng cũng có thể biểu diễn bởi một cặp (r,φ) trong tọa độ cực với r là khoảng cách từ gốc tọa độ tới đường thẳng đó và φ là góc tạo bởi trục 0X với đường thẳng vuông góc với nó, hình 5.9 biểu diễn đường thẳng hough trong tọa độ Decard.

Ngược lại, mỗi một cặp (r,φ) trong toạ độ cực cũng tương ứng biểu diễm một đường thẳng trong mặt phẳng.

Giả sử M(x,y) là mộ điểm thuộc đường thẳng được biểu diễn bởi (r,φ), gọi H(X,Y) là hình chiếu của gốc toạ độ O trên đường thẳng ta có:

X= r. cosφ và Y= r.sinφ

Mặt khác, ta có: $\overrightarrow{OH}.\overrightarrow{HA}=0$

Từ đó ta có mối liên hệ giữa (x,y) và (r,φ) như sau: x*cosφ+y*sinφ= r.

Xét n điểm thẳng hàng trong tọa độ Đề các có phương trình $x*\cos\varphi_0+y*\sin\varphi_0= r_0$. Biến đổi Hough ánh xạ n điểm này thành n đường sin trong tọa độ cực mà các đường này đều đi qua (r_0,φ_0). Giao điểm (r_0,φ_0) của n đường sin sẽ xác định một đường thẳng trong hệ tọa độ đề các. Như vậy, những đường thẳng đi qua điểm (x,y) sẽ cho duy nhất một cặp (r,φ) và có bao nhiêu đường qua (x,y) sẽ có bấy nhiêu cặp giá trị (r,φ).

Chương 6:

ỨNG DỤNG XỬ LÝ ẢNH

6.1. PHÁT HIỆN GÓC NGHIÊNG VĂN BẢN DỰA VÀO CHU TUYẾN

Góc nghiêng văn bản là một bài toán kinh điển trong xử lý ảnh văn bản. Một hệ thống xử lý ảnh văn bản thường phải giải quyết bài toán phát hiện góc nghiêng như một bước đầu tiên và tất yếu. Chính vì vậy, cùng với sự phát triển của xử lý ảnh nói chung và xử lý ảnh văn bản nói riêng, bài toán góc nghiêng văn bản cũng được quan tâm ngày càng nhiều và dưới nhiều góc độ khác nhau. Có rất nhiều hướng tiếp cận cho bài toán góc nghiêng văn bản từ trước tới nay. Các thuật toán phát hiện góc nghiêng thường được xây dựng cho các hệ thống phân tích ảnh văn bản khác nhau nên chỉ giải quyết cho những loại ảnh văn bản cụ thể. Có thể chia ra một số hướng tiếp cận cơ bản cho bài toán góc nghiêng văn bản như sau:

- Các thuật toán dựa vào phân tích hình chiếu (Projection Profile)
- Các thuật toán dựa vào biến đổi Hough (Hough Transform)
- Các thuật toán phân tích láng giềng (Nearest Neighbour Clustering)
- Phương pháp dùng các phép toán hình thái

Dựa vào tính chất mỗi đối tượng ảnh có duy nhất một chu tuyến ngoài và quan niệm con người nhận ra độ nghiêng của văn bản dựa vào cỡ chữ chiếm chủ đạo trong văn bản. Mục này đề cập đến việc tính toán kích thước chủ đạo của các đối tượng ảnh trong văn bản thông qua kỹ thuật tính biểu đồ tần xuất kích thước hình chữ nhật nhỏ nhất bao quanh đối tượng ảnh. Việc xác định góc nghiêng văn bản sẽ được xác định nhờ phép biến đổi Hough cho những điểm giữa đáy của hình chữ nhật nhỏ nhất bao quanh đối tượng ảnh cho các đối tượng ảnh có kích thước chủ đạo.

6.1.1. Tính toán kích thước chủ đạo của các đối tượng ảnh

Như đã nói từ các phần trên, góc nghiêng được xác định dựa vào biến đổi Hough. Ở đây, chúng ta chỉ áp dụng biến đổi Hough cho những điểm giữa đáy của các hình chữ nhật ngoại tiếp các đối tượng có kích thước chủ đạo trong ảnh. Như vậy, công việc đầu tiên cần thực hiện là xác định được các hình chữ nhật ngoại tiếp các đối tượng hay nói cách khác là xác định biên các đối tượng.

Hình 6.1. **Các hình chữ nhật ngoại tiếp đối tượng ảnh**

Ở đây, ta dùng thuật toán dò biên đã được cải tiến trong chương 2 để xác định biên cho các đối tượng trong ảnh văn bản. Hình chữ nhật ngoại tiếp đối tượng sẽ được xác định ngay sau khi dò được biên cho đối tượng đó.

Một cách trực tiếp giống như một số thuật toán khác, ta có thể dùng biến đổi Hough áp dụng lên đáy của các hình chữ nhật ngoại tiếp các đối tượng này và ước lượng góc nghiêng cho văn bản. Tuy nhiên, ở đây biến đổi Hough được áp dụng sau khi đã loại bớt đi một số đối tượng bằng các ngưỡng kích thước.

Hình 6.2. **Ví dụ về một ảnh văn bản nghiêng với nhiều loại đối tượng**

Mục đích của việc dùng ngưỡng là dựa vào thước đo kích thước để phân loại đối tượng. Nói cách khác, dùng ngưỡng phân loại ta có thể phân biệt được một cách tương đối những đối tượng là ký tự và đối tượng phi ký tự. Nhờ biết phân biệt đối tượng, ta sẽ chỉ làm việc với các đối tượng có kích thước chủ đạo trong ảnh do đó độ chính xác của thuật toán được cải thiện đáng kể.

Giả sử ta có một ảnh văn bản nghiêng như hình vẽ trên đây. Rõ ràng đây là một ảnh văn bản phức tạp với nhiều đối tượng phi ký tự và số ký

tự chữ cái trong ảnh trên là rất ít. Mặc dù vậy, chúng ta vẫn cho rằng ảnh bị nghiêng. Vậy ta đã căn cứ vào đâu khi kết luận ảnh bị nghiêng? Trong một ảnh văn bản, thông thường các đối tượng ký tự chiếm nhiều hơn những đối tượng khác. Xuất phát từ quan điểm nhìn nhận sự vật của mắt người và đặc thù trên đây của ảnh văn bản, gợi ý cho chúng ta một hướng giải quyết bài toán góc nghiêng là xác định các đối tượng chủ đạo trong ảnh chính là các ký tự, và căn cứ vào chúng để ước lượng góc nghiêng.

Ý tưởng để xác định các đối tượng có kích thước chủ đạo trong ảnh là dùng kỹ thuật lập biểu đồ tần xuất hay Histogram kích thước để ước lượng một ký tự có tần số xuất hiện nhiều nhất trong văn bản mà ta gọi là đối tượng chuẩn. Với mỗi một ảnh đầu vào, ta sẽ xác định một đối tượng chuẩn riêng và tự động trong chương trình. Sau đó, lấy đối tượng này làm chuẩn và so sánh các đối tượng còn lại với nó. Những đối tượng có kích thước xấp xỉ bằng kích thước của đối tượng này sẽ được chọn để áp dụng biến đổi Hough. Một đối tượng được xem là xấp xỉ bằng kích thước của đối tượng khác nếu chênh lệch kích thước giữa chúng bé hơn một ngưỡng được định nghĩa trước.

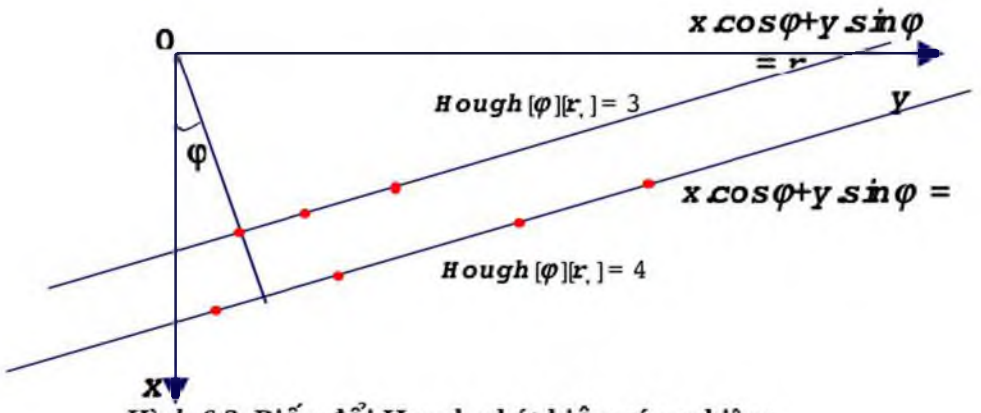
6.1.2. Biến đổi Hough và phát hiện góc nghiêng văn bản

6.1.2.1. Áp dụng biến đổi Hough trong phát hiện góc nghiêng văn bản

Ý tưởng của việc áp dụng biến đổi Hough trong phát hiện góc nghiêng văn bản là dùng một mảng tích luỹ để đếm số điểm ảnh nằm trên một đường thẳng trong không gian ảnh. Mảng tích luỹ là một mảng hai chiều với chỉ số hàng của mảng cho biết góc lệch φ của một đường thẳng và chỉ số cột chính là giá trị r khoảng cách từ gốc toạ độ tới đường thẳng đó. Sau đó tính tổng số điểm ảnh nằm trên những đường thẳng song song nhau theo các góc lệch thay đổi. Góc nghiêng văn bản tương ứng với góc có tổng giá trị mảng tích luỹ cực đại.

Theo biến đổi Hough, mỗi một đường thẳng trong mặt phẳng tương ứng được biểu diễn bởi một cặp (r,φ). Giả sử ta có một điểm ảnh (x,y) trong mặt phẳng. Vì qua điểm ảnh này có vô số đường thẳng, mỗi đường thẳng lại cho một cặp (r,φ) nên với mỗi điểm ảnh ta sẽ xác định được một số cặp (r,φ) thoả mãn phương trình Hough.

Hình vẽ dưới đây minh hoạ cách dùng biến đổi Hough để phát hiện góc nghiêng văn bản. Giả sử ta có một số điểm ảnh. Đây là những điểm giữa đáy các hình chữ nhật ngoại tiếp các đối tượng đã được lựa chọn từ các bước trước. Ở đây, ta thấy trên mặt phẳng có hai đường thẳng song song nhau. Đường thẳng thứ nhất có ba điểm ảnh nên giá trị mảng tích luỹ bằng 3. Đường thẳng thứ hai có giá trị mảng tích luỹ bằng 4. Do đó, tổng giá trị mảng tích lũy cho cùng góc φ trường hợp này bằng 7.



Hình 6.3. **Biến đổi Hough phát hiện góc nghiêng**

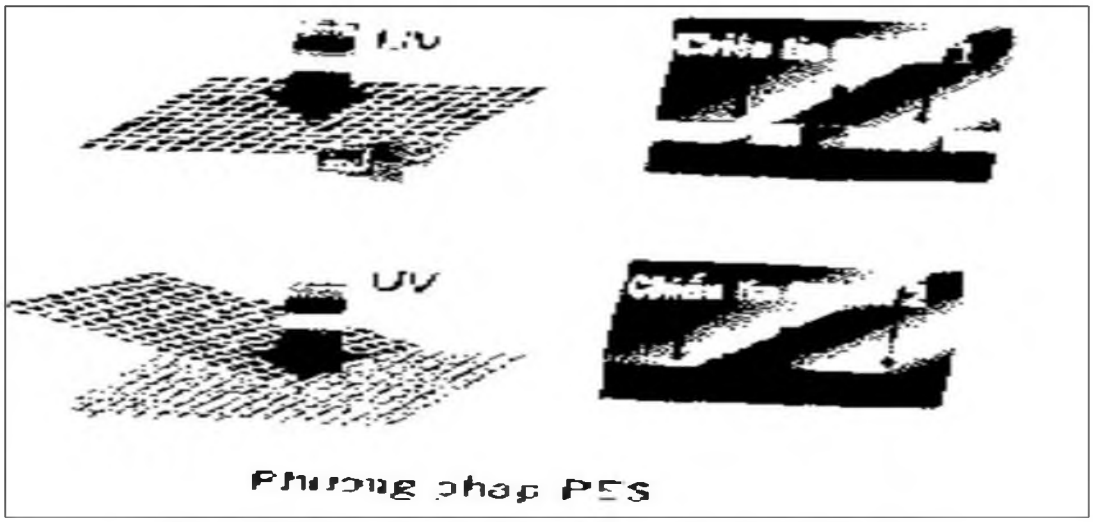
6.1.2.2. Thuật toán phát hiện và hiệu chỉnh góc nghiêng văn bản

a) Xử lý ngoại lệ

Sau giai đoạn tiền xử lý ảnh ta thu được ảnh trung gian TempImage. Thuật toán phát hiện góc nghiêng sẽ làm việc với ảnh trung gian này để tìm ra góc nghiêng cho văn bản và sau đó dùng thuật toán quay ảnh để quay ảnh ban đầu với góc nghiêng vừa tìm được.

Tuy nhiên, một điểm cần được xét đến trong thuật toán phát hiện góc nghiêng là xử lý những ảnh văn bản phức tạp hoặc các trường hợp ngoại lệ. Ta sẽ lần lượt đưa ra các phương án xử lý cho các trường hợp này.

Ảnh có quá ít ký tự



Hình 6.4. **Ví dụ về một ảnh nghiêng có ít ký tự chữ cái**

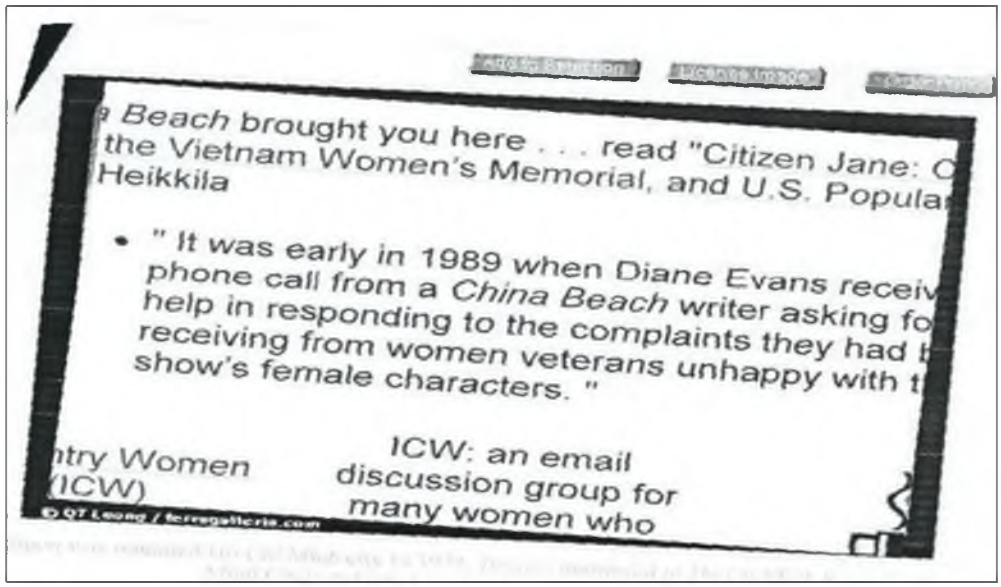
Trường hợp thứ nhất là trong ảnh có quá ít ký tự chữ cái chưa đủ để xác định được góc nghiêng. Các đối tượng trong ảnh chủ yếu là ảnh hoặc nhiễu, đặc biệt các ký tự nghiêng các góc khác nhau do đặc thù riêng của ảnh. Hình vẽ dưới đây minh hoạ một ảnh văn bản nghiêng với số ký tự rất ít.

Vì vậy, ta chỉ đưa ra kết luận về góc nghiêng cho văn bản trong trường hợp số lượng các đối tượng này phải lớn hơn một ngưỡng nào đó. Trong chương trình số lượng này được chọn bằng 70 đối tượng.

Các đối tượng bao nhau

Trường hợp ngoại lệ khác là các đối tượng bao nhau. Đây là một cản trở đối với những thuật toán xác định góc nghiêng khác đặc biệt là những thuật toán theo phương pháp phân tích láng giềng thân cận như đã được đề cập ở trên.

Mặc dù số ký tự trong văn bản có thể rất nhiều nhưng các ký tự hầu hết bị chứa trong các đối tượng khác lớn hơn nhiều chẳng hạn như ảnh hay bảng biểu. Hình 5.5 dưới đây minh họa cho cho trường hợp các ký tự bị bao bởi đối tượng ảnh. Khi đó, nhiệm vụ là phải nhận ra được sự bao hàm giữa các đối tượng và tách, lấy được các đối tượng ký tự bị bao bởi các đối tượng lớn hơn.



***Hình 6.5.* Ví dụ về văn bản nghiêng có các đối tượng bao nhau**

Ở đây, ta dùng một kỹ thuật bóc dần những đối tượng lớn ngoại cỡ để xác định những ký tự trong đó. Một đối tượng được gọi là có kích thước ngoại cỡ được quy ước là đối tượng có chiều rộng và chiều cao lớn hơn 200 pixel. Nếu trong quá trình dò biên ta gặp một đối tượng như vậy, ta sẽ cách ly nó ra khỏi tập đối tượng đang xét. Các đối tượng này sẽ được dùng đến nếu cuối cùng số đối tượng được chọn để áp dụng biến đổi Hough bé hơn 70. Ta xem như đối tượng này là một ảnh và tiếp tục duyệt các đối tượng bên trong nó để lấy ra những đối tượng ký tự.

b) Thuật toán phát hiện góc nghiêng văn bản dựa vào biên

Giả sử ảnh đầu vào là ảnh màu (Image). Thuật toán phát hiện và chỉnh sửa góc nghiêng văn bản được thực hiện theo các bước chính sau:

Bước 1: Tiền xử lý ảnh màu Image được ảnh trung gian TempImage

Bước 2: Xác định chu tuyến ngoài cho các đối tượng:

Duyệt ảnh từ trên xuống dưới, từ trái sang phải, điểm ảnh hiện tại là (x,y):

- Nếu (x,y) có màu khác màu nền và chưa xét Label [x][y]=0 :

  - Tăng giá trị nhãn lên một đơn vị: label=label+1.

  - Gọi hàm xác định chu tuyến DetectAnObject với điểm xuất phát (x,y), rec dùng lưu hình chữ nhật chứa đối tượng, hàm trả về -1 nếu đối tượng cô lập, 1 nếu đối tượng có kích thước bình thường và 0 trong trường hợp ngược lại có kích thước kỳ lạ.

  - Nếu hàm chu tuyến trả về 1 :

    + Tăng số đối tượng: Id=Id+1.

    + Lưu lại Rec[Id] =rec.

    + Duyệt từ phải sang trái, tìm điểm cùng hàng có nhãn bằng label và nhảy tới đó.

  - Ngược lại nếu hàm DetectAnObject trả về 0:

    + Nếu rec.Wid > 200 và rec.Hei > 200 (kích thước quá lớn) thiết lập màu nền cho các điểm biên lấy điểm (x,y+1) làm điểm xét tiếp theo.

    + Ngược lại, duyệt từ phải sang trái tìm điểm cùng hàng đầu tiên có nhãn bằng label và nhảy tới đó.

- Nếu (x,y) có màu khác màu nền và đã xét, Label [x][y] > 0, duyệt từ phải sang trái tìm điểm đầu tiên cùng hàng có nhãn bằng Label [x][y] và nhảy tới đó.

Bước 3: Dùng mảng Rec[N] xác định các giá trị ngưỡng trung bình WidAvr, HeiAvr và PrmAvr.

Bước 4: Áp dụng biến đổi Hough

Với mỗi phần tử Rec[i] của mảng Rec:

Nếu Rec[i].Pmr< 6*PrmAvr và Rec[i].Wid< 4*WidAvr và Rec[i].Hei< 4*HeiAvr , áp dụng biến đổi Hough cho điểm giữa đáy của hình chữ nhật.

Bước 5: Dùng mảng kết quả Hough[360][Dis] ước lượng góc nghiêng cho văn bản:

- Gán giá trị cực đại các phần tử của mảng Hough[360][Dis] cho max.

- Gán maxtotal = 0.

- Với mỗi hàng i của mảng

- Khởi tạo cho tổng các giá trị của hàng: total = 0.

- Với mỗi giá trị cột j, nếu Hough[i][j] > max/2, tăng tổng total = total+Hough[i][j].

- Nếu total> maxtotal:

+ maxtotal = total.

+ Góc lệch φ = i.

Bước 6: Quay lại ảnh Image với góc lệch φ vừa xác định được từ bước 5.

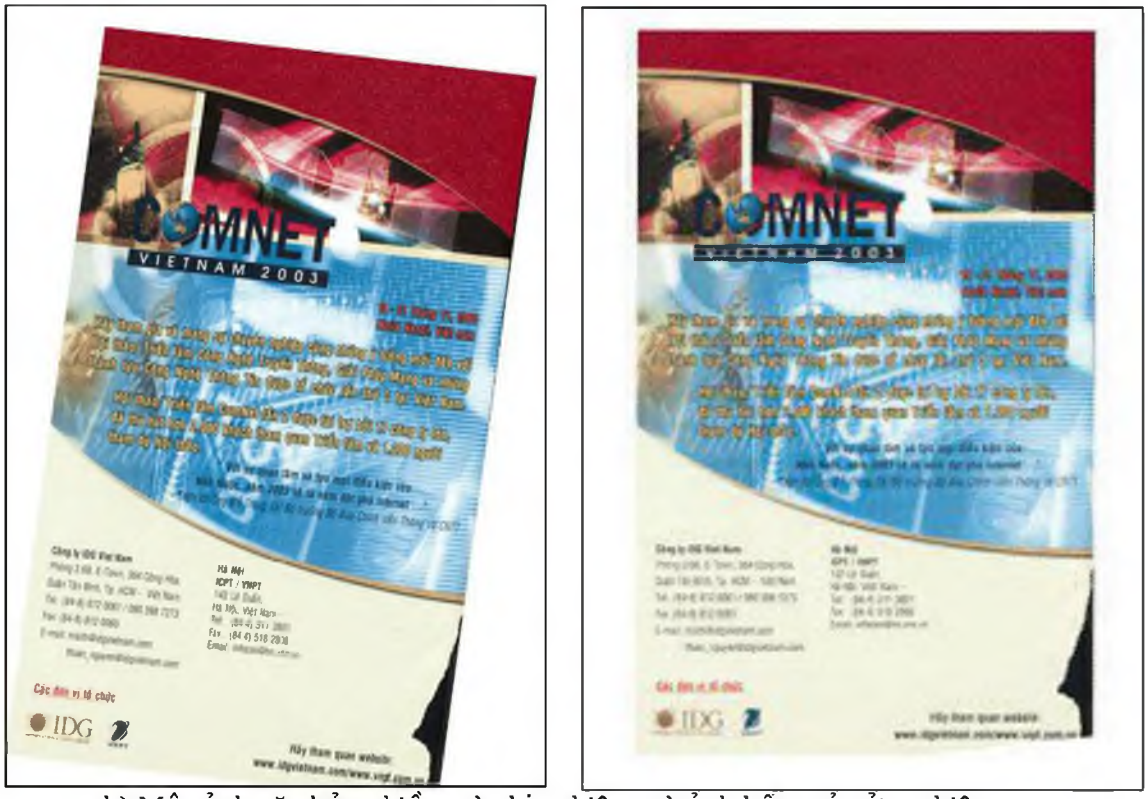
6.1.2.3. Thực nghiệm và kết quả

Chúng tôi đã cài đặt thuật toán đề xuất bằng công cụ ngôn ngữ lập trình Visual C^{++} và thử nghiệm nhiều với bộ test đa dạng. Kết quả thử nghiệm đã chứng minh tính chính xác của thuật toán và khả năng làm việc với nhiều loại văn bản khác nhau cũng như các góc lệch khác nhau. Kết quả thuật toán làm việc tốt ngay cả với ảnh màu có các đối tượng ảnh có kích thước co cụm và có độ lệch dưới 15°, dưới đây là một số hình ảnh mà thuật toán thực hiện được.

Ưu điểm nổi bật của thuật toán là việc phân biệt các ký tự chữ cái và những đối tượng phi ký tự như nhiễu, đối tượng đồ họa, đường thẳng, v.v.. do đó độ chính xác của thuật toán tăng lên. Để loại trừ các đối tượng phi ký tự này, thuật toán đã tự động xác định các ngưỡng kích thước. Ba ngưỡng kích thước được dùng là chiều rộng, chiều cao và chu vi của một đối tượng.

Hình 6.6 cho thấy thuật toán ngoài việc có thể thực hiện đối với ảnh màu do bổ sung thêm khâu tiền xử lý trước đó, còn có khả năng thực hiện việc phát hiện góc nghiêng và tiến hành quay ảnh với nhiều đối tượng khác nhau, ngay cả đối với ảnh bản đồ là đối tượng mà các hệ thống khác thường không thực hiện được như MapScan và VnDOCR trong khi việc quay này là cần thiết để có thể nhận được các chữ của bản đồ trong quá trình véctơ hóa tự động.

a) Một ảnh bản đồ có ít ký tự nghiêng góc và ảnh kết quả sau khi sửa nghiêng



b) Một ảnh văn bản nhiều màu bị nghiêng và ảnh kết quả sửa nghiêng

***Hình 6.6.* Một số loại ảnh màu mà thuật toán thực hiện hiệu chỉnh**

6.2. PHÂN TÍCH TRANG TÀI LIỆU

Một trong những vấn đề cơ bản của nhận dạng các trang văn bản nói chung và các trang văn bản có lẫn các đối tượng khác như ảnh, sơ đồ, biểu đồ v.v.. (Hình 6.7) là phải phân tích được chúng. Chỉ khi nào phân tích được chúng một cách chính xác, mới có thể tiến hành nhận dạng các thông tin trong các đối tượng một cách chính xác và sau quá trình nhận dạng trang văn bản mới được trả lại cấu trúc của nó một cách chính xác.

Cảng Marseille khói lửa

•DANH ĐỨC (fax từ Paris)

9 giờ tối 14.6 trên đường về nhà, khi tôi từ hầm xe điện lên đã nghe trên một đài FM tin hooligans Anh quậy phá ở cảng Marseille, nơi sẽ diễn ra trận Anh - Tunisie ngày 15.6.98. Theo tin ban đầu, khoảng 300 hooligans Anh tụ tập ở Vieux Port (tức khu bến cảng cổ) trước một pub (quán rượu) mang tên "OM Cafe" nguyên là địa điểm tụ tập của các cổ động viên đội OM, giăng biểu ngữ và hô khẩu hiệu "đả cáo bọn Tunisie". Thông tin ban đầu cho biết cảnh sát đã can thiệp bằng lựu đạn cay song không giải tỏa được đám hooligans ngày càng đông hơn, và còn hứa hẹn "đêm nay sẽ nổi lửa" ở Marseille đấy. Sáng ra, tình hình Marseille chỉ còn là "ngàn ngang", nhưng tổn thất, trong đó có 37 người bị thương nặng.

chiến, cứ ngồi yên trong các xe bít bùng, không chương mặt ra kẻo bị xem là khiêu khích hoặc giải tán sớm thì bị gọi là "quân phiệt" và càng làm cho bạn động sửng bùng nổ mà nguyên cớ sẽ đổ thừa cho cảnh sát. Khi vụ đụng độ đầu tiên xảy ra, cảnh sát phong tỏa ngay khu cảng cổ, bắt giữ ngay số người ẩu đả. Cảnh sát trưởng Marseille, ông Daniel Herbst vẫn còn tin rằng có thể nắm được tình hình, thông báo với cảnh

Hooligans Anh đốt cờ Tunisie trên đường phố Marseille

trung. "Tiểu đoàn" ủng hộ viên Anh cuồng kích nhất tấn công dài dài đã chiến trang bị khiên, dùi cui, súng phóng lựu (đạn khói)... bằng các lon bia (bia mua trong

Hình 6.7. **Trang văn bản có lẫn ảnh**

Mục này đề cập đến việc phân tích văn bản theo tiếp cận dưới lên nhờ việc sử dụng khoảng cách Hausdorff giữa các đối tượng ảnh. Ban đầu các đối tượng ảnh sẽ được tách bởi chu tuyến ngoài. Các đối tượng có kích thước hình chữ nhật phủ nhỏ hơn một ngưỡng nào đó sẽ được nhóm với nhau theo lân cận gần nhất dựa vào việc sử dụng khoảng cách Hausdorff để tạo ra các khối. Các đối tượng ảnh còn lại sẽ được tiếp tục phân tích như là đối với một trang văn bản có kích thước nhỏ hơn.

6.2.1. Quan hệ Q_θ

Định nghĩa 5.1: [Liên kết Q_θ]

Cho trước ngưỡng θ, hai đối tượng ảnh U, $V \subseteq \Im$ hoặc $\overline{\Im}$ được gọi là liên kết theo θ và kí hiệu $Q_\theta(U,V)$ nếu tồn tại dãy các đối tượng ảnh X_1, X_2,... X_n sao cho:

(i) $U \equiv X_1$

(ii) $V \equiv X_n$

(iii) $h(X_i, X_{i+1}) < \theta \ \forall i, 1 \le i \le n-1$

Mệnh đề 6.1: Quan hệ liên kết Q_θ là một quan hệ tương đương

Chứng minh:

(i) Phản xạ: $U \subseteq \Im$ hoặc $\overline{\Im}$ ta có $h(U,U)= 0 < \theta$

(ii) Đối xứng: Giả sử có $Q_\theta(U,V)$ cần phải chứng minh $Q_\theta(V,U)$

Thật vậy, theo giả thiết tồn tại dãy đối tượng ảnh $X_1, X_2,..., X_n$ sao cho:

$U \equiv X_1, V \equiv X_n, h(X_i, X_{i+1}) < \theta \ \forall i, 1 \leq i \leq n-1$

Khi đó, với dãy đối tượng ảnh $Y_1, Y_2,..., Y_n$ mà: $Y_i \equiv X_{n-i+1} \ \forall i, 1 \leq i \leq n$ ta có:

$V \equiv Y_1, U \equiv Y_n, h(Y_i, Y_{i+1}) < \theta \ \forall i, 1 \leq i \leq n-1.$

Suy ra, $Q_\theta(V,U)$.

(iii) Bắc cầu: Giả sử ta có $Q_\theta(U,V)$ và $Q_\theta(V,T)$ ta cần chứng minh $Q_\theta(U,T)$

Thật vậy, vì $Q_\theta(U,V)$ nên tồn tại dãy đối tượng ảnh $X_1, X_2,...,X_n$ sao cho:

$U \equiv X_1, V \equiv X_n, h(X_i, X_{i+1}) < \theta \ \forall i, 1 \leq i \leq n-1$

$Q_\theta(V,T)$ nên tồn tại dãy đối tượng ảnh $Y_1, Y_2,...,Y_m$ sao cho:

$U \equiv Y_1, T \equiv T_m, h(Y_i, Y_{i+1}) < \theta \ \forall i, 1 \leq i \leq m-1$

Khi đó, dãy các đối tượng ảnh $Z_1, Z_2,...,Z_n, Z_{n+1},..., Z_{n+m}$ ở đây:

$Z_i \equiv X_i \ \forall i, 1 \leq i \leq n$ và $Z_{n+i} \equiv Y_i \ \forall i, 1 \leq i \leq m$ có các tính chất:

$U \equiv Z_1, T \equiv Z_{n+m}, h(Z_i, Z_{i+1}) < \theta \ \forall i, 1 \leq i \leq n+m-1$

Suy ra $Q_\theta(U,T)$.

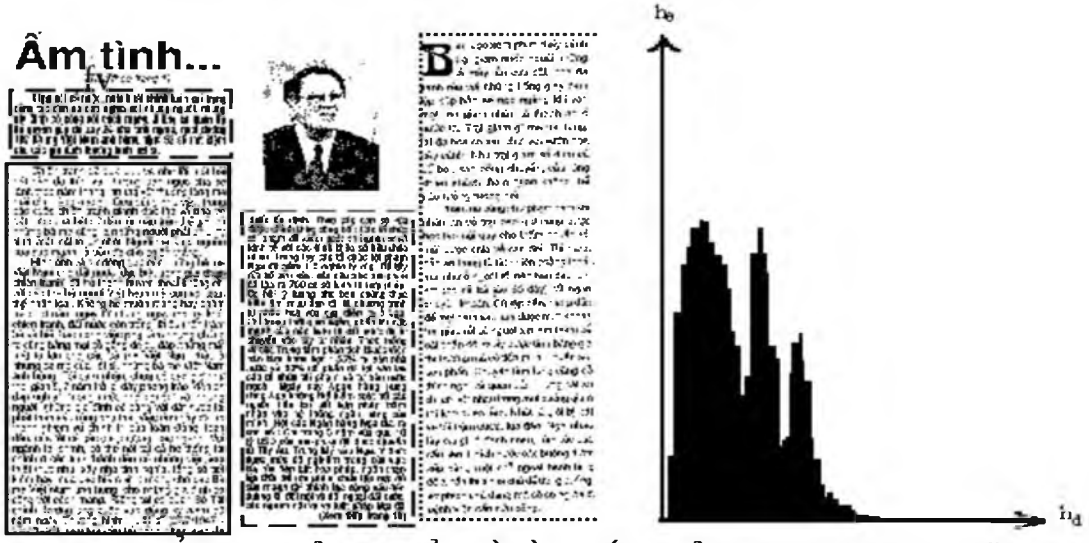
6.2.2. Phân tích trang văn bản nhờ khoảng cách Hausdorff bởi quan hệ Q_θ

Thông thường, việc tiến hành phân tích định dạng trang thường được tiến hành sau khi ảnh được xác định góc nghiêng và quay về góc 0.

Phân tích định dạng trang có thể thực hiện từ dưới lên hay từ trên xuống. Với phân tích từ trên xuống, một trang được chia từ những phần lớn thành các phần con nhỏ hơn, ví dụ nó có thể được chia thành một số cột văn bản. Sau đó mỗi cột có thể được chia thành các đoạn, mỗi đoạn lại được chia thành các dòng văn bản... Tiếp cận theo hướng này có các phương pháp: Sử dụng các phép chiếu nghiêng, gán nhãn chức năng, phân tích khoảng trống trắng v.v.. Ưu điểm lớn nhất của các phương pháp phân tích từ trên xuống là nó dùng cấu trúc toàn bộ trang để giúp cho phân tích định dạng được nhanh chóng. Đây là cách tiếp cận hiệu quả cho hầu hết các dạng trang. Tuy nhiên, với các trang không có các biên tuyến tính và có sơ đồ lẫn cả bên trong và quanh văn bản, các phương pháp này có thể không thích hợp. Ví dụ, nhiều tạp chí tạo văn bản quanh quanh một sơ đồ

ở giữa, vì thế văn bản đi theo những đường cong của đối tượng trong sơ đồ chứ không theo đường thẳng.

Phân tích định dạng từ dưới lên bắt đầu với những phần nhỏ và nhóm chúng vào những phần lớn hơn kế tiếp tới khi mọi khối trên trang được xác định. Tuy nhiên không có một phương pháp tổng quát nào điển hình cho mọi kỹ thuật phân tích dưới lên. Trong [1] các tác giả đã đề xuất thuật toán phân tích trang văn bản hỗn hợp thành các thành phần pageANALYSIS theo tiếp cận dưới lên nhờ việc sử dụng khoảng cách Hausdorff giữa các đối tượng ảnh thông qua quan hệ Q_θ. Ban đầu các đối tượng ảnh sẽ được cô lập bởi chu tuyến ngoài (đường biên kín nhỏ nhất chứa mọi điểm ảnh của đối tượng ảnh). Các đối tượng có kích thước hình chữ nhật phủ nhỏ hơn một ngưỡng θ nào đó sẽ được nhóm với nhau theo lân cận gần nhất dựa vào việc sử dụng khoảng cách Hausdorff để tạo ra các khối, các đối tượng ảnh còn lại sẽ được tiếp tục phân tích như là đối với một trang văn bản. Trong đó, ngưỡng θ thường được xác định theo kinh nghiệm người sử dụng. Trong phần dưới đây chúng tôi đề xuất việc lựa chọn ngưỡng một cách tự động dựa vào biểu đồ tần xuất (histogram).



Hình 6.8: **Ảnh văn bản và biểu đồ tần xuất khoảng cách Hausdorff giữa các đối tượng ảnh**

Do thuật toán phân tích trang văn bản pageANALYSIS [1] dựa khoảng cách Hausdorff bởi quan hệ Q_θ là quá trình duyệt tìm các lớp tương đương theo khoảng cách θ. Các đối tượng ảnh trong cùng một khối văn bản có những đặc trưng tương đối giống nhau về kích thước và khoảng cách giữa chúng với các đối tượng lân cận. Hơn nữa, một trang văn bản lại thường có một vài dạng đối tượng chỉ đạo. Do đó, ta có thể lựa chọn ngưỡng θ ban đầu thông qua việc đánh giá biểu đồ tần xuất khoảng cách Hausdorff giữa các đối tượng ảnh (hình 5.8).

Từ biểu đồ tần xuất khoảng cách Hausdorff giữa các đối tượng ảnh của ảnh văn bản cần phân tích. Ngưỡng θ được lựa chọn trong các giá trị hθ tương ứng là các đỉnh trong biểu đồ tần xuất đó chính là các giá trị ứng với nhiều phần tử cùng loại nhất. Với ngưỡng θ đã chọn ta tiến hành phân vùng theo tiếp cận dưới lên nhờ việc sử dụng khoảng cách Hausdorff giữa các đối tượng ảnh thông quan quan hệ Q_θ. Kết quả thu được và tập hợp các hình chữ nhật rời nhau thể hiện các vùng trong ảnh.

Việc lựa chọn ngưỡng θ phù hợp nhất sẽ được tiến hành thông qua việc đánh giá sự sai lệch của văn bản so với mẫu. Với mỗi ngưỡng θ, ta sẽ tìm được mẫu tương ứng có độ lệch nhỏ nhất. Ngưỡng θ và văn bản mẫu tương ứng với độ sai lệch nhỏ nhất trong số các độ lệch sẽ được lựa chọn. Nếu sai số nhỏ nhất chấp nhận được (nhỏ hơn một ngưỡng cho trước nào đó) thì số vùng của văn bản sẽ được xác định tương ứng với số vùng của văn bản mẫu được lựa chọn. Khi đó, văn bản sẽ được phân tích trang dựa theo các thuộc tính của văn bản mẫu. Trong trường hợp ngược lại có thể xem văn bản không thuộc tập văn bản mẫu và do vậy có thể tiến hành bổ sung văn bản đang xét vào tập mẫu.

6.2.3. Phân tích trang văn bản dựa vào mẫu

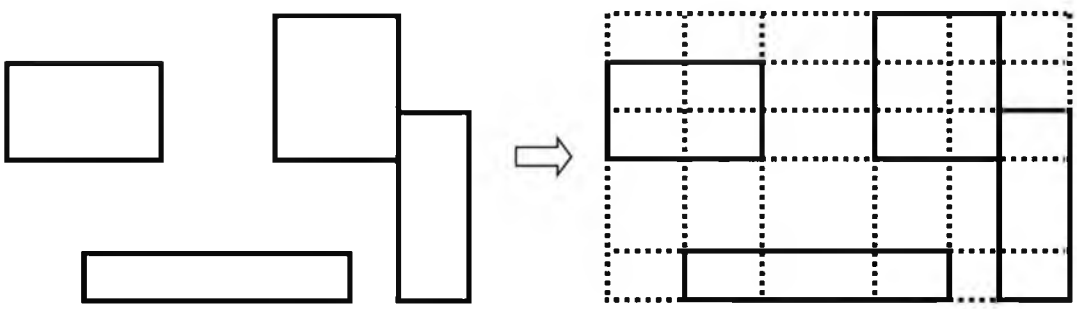
6.2.3.1. Đánh giá độ lệch cấu trúc văn bản theo mẫu

Quá trình phân tích trang văn bản dựa theo các mẫu đã có sẽ được thực hiện thông qua việc đánh giá sự sai lệch của của văn bản so với văn bản mẫu. Văn bản mẫu có độ lệch nhỏ nhất so với văn bản cần hiệu chỉnh sẽ được lựa chọn. Nếu độ lệch nhỏ nhất tìm được nằm trong phạm vi cho phép thì có thể xem văn bản cần hiệu chỉnh thuộc trong số mẫu đã có, trường hợp ngược lại văn bản được xem như là mẫu mới và được bổ sung vào tập văn bản mẫu.

Việc đánh giá độ sai lệch của văn bản so với văn bản mẫu sẽ được tiến hành thông qua việc xây dựng lưới tựa các vùng chữ nhật cơ bản của mẫu và đánh giá độ lệch của vùng so với lưới. Độ lệch của văn bản so với mẫu sẽ được đánh giá dựa trên sự tương đồng của cả văn bản và mẫu so với lưới tương ứng.

Việc xây dựng lưới tựa các vùng hình chữ nhật

tìm được trong văn bản thông qua việc chọn ngưỡng θ dựa vào biểu đồ tần xuất hay các vùng văn bản chữ nhật trong mẫu. Lưới là tập các toạ độ ngang dọc, hình 3 thể hiện ví dụ minh hoạ việc xây dựng lưới từ tập các hình chữ nhật.



***Hình 6.9:* Xây dựng lưới tựa các hình chữ nhật**

Độ lệch của một vùng c_k so với ô lưới $M_{Grid}(i,j)$ được tính bởi công thức:

$$Intersect(c_k, M_{Grid}(i,j)) = \begin{cases} 1 & \text{Nếu } c_k \cap M_{Grid}(i,j) \neq \varnothing \\ 0 & \text{Nếu ngược lại} \end{cases}$$

và độ lệch của một vùng c_k so với lưới M_{Grid} được xác định bởi tổng độ lệch của vùng so với các ô của của lưới M_{Grid}:

$$Segments(c_k, M_{Grid}) = \sum_{i=1}^{n_h} \sum_{j=1}^{n_v} Intersect(c_k, M_{Grid}(i,j))$$

Gọi tập hợp các vùng nằm trong lưới mà có độ lệnh khác 0 là $C_{M_{Grid}}$ ta có:

$$C_{M_{Grid}} = \{c_k \mid Segments(c_k, M_{Grid}) > 0\}$$

Khi đó, độ lệnh của văn bản so với ô lưới (i,j) được xác định bởi công thức:

$$N_{M_{Grid}}(i,j) = \sum_{c_k \in C_{M_{Grid}}} \left(Intersect(c_k, M_{Grid}(i,j)) \times \frac{1}{Segments(c_k, M_{Grid})} \right)$$

và độ của văn bản so với lưới được xác định là tổng độ lệch của văn bản so với từng ô của lưới là:

$$N_{M_{Grid}} = \sum_{i=1}^{n_h} \sum_{j=1}^{n_v} N_{M_{Grid}}(i,j)$$

Độ lệch của văn bản so với mẫu được đánh giá bằng tỷ số giữa tổng độ lệch của các vùng trong văn bản và mẫu đối với từng ô của lưới kết hợp giữa hai lưới được xây dựng từ các vùng của văn bản và mẫu trên tổng số vùng của văn bản và mẫu:

$$S = \frac{\sum_{i=1}^{n_h}\sum_{j=1}^{n_v}\left|N_{MG_{Grid}}(i,j) - N'_{MG_{Grid}}(i,j)\right|}{n_c + n'_c}$$

Trong đó:

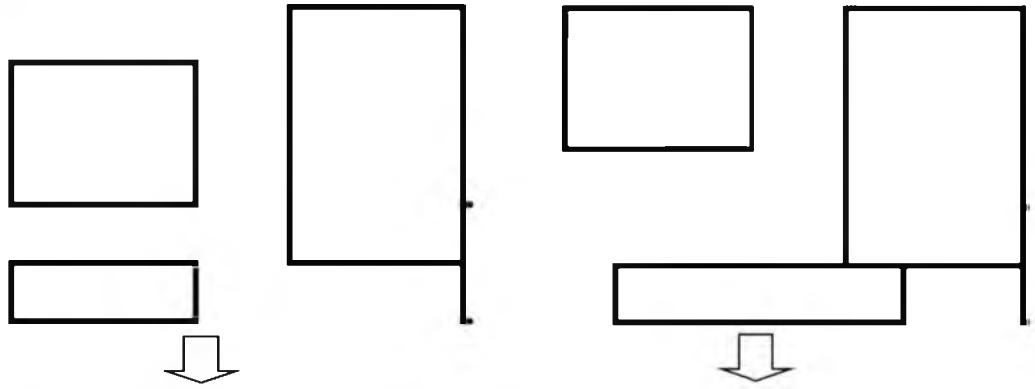
MG_{Grid}- là lưới kết hợp từ hai lưới được xây dựng từ các hình chữ nhật vùng của văn bản và mẫu

n_c, n'_c - là số vùng của văn bản và số vùng của mẫu

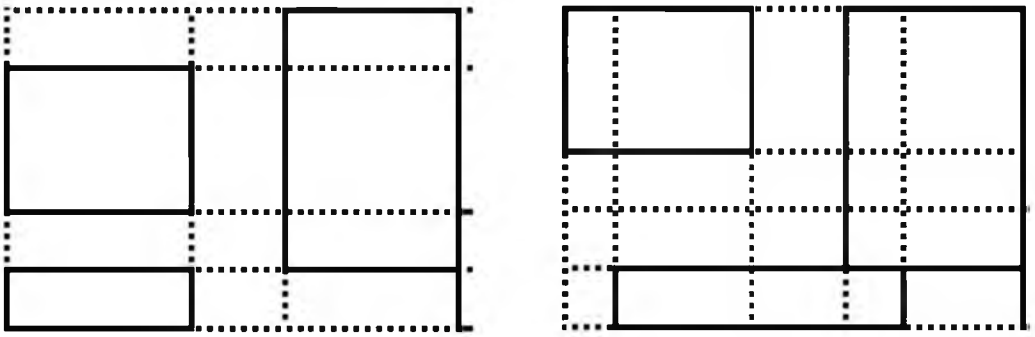
$N_{MG_{Grid}}(i,j), N'_{MG_{Grid}}$ - là độ lệnh của văn bản và mẫu so với ô lưới (i,j)

***Ví dụ minh hoạ đánh giá độ lệch văn bản so với mẫu**

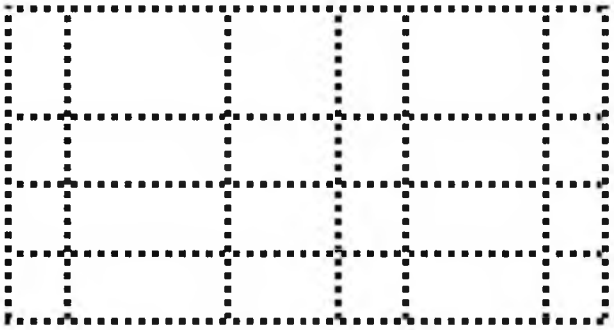
Cấu trúc văn bản, cấu trúc mẫu và lưới tựa hình chữ nhật xây dựng tương ứng



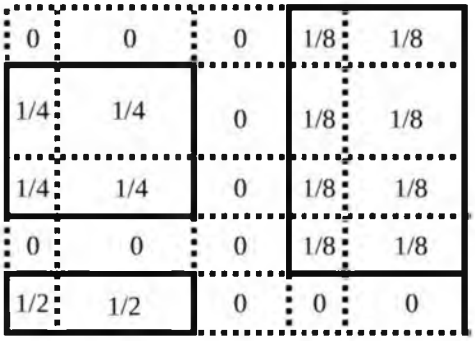
Lưới tựa hình chữ nhật tương ứng

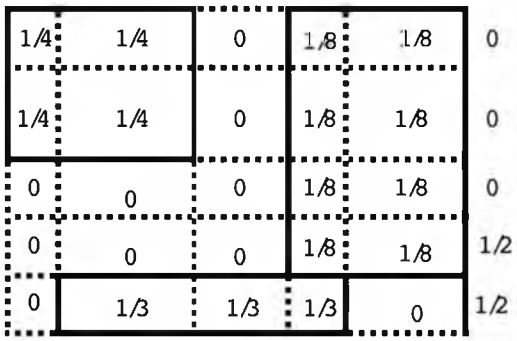


Lưới xây dựng kết hợp từ các lưới tựa vùng chữ nhật văn bản và mẫu



Khi đó, giá trị độ lệch của văn bản và mẫu so với các ô lưới được tính theo công thức $N_{M_{Grid}}(i,j)$ là:





và do đó, độ lệch của văn bản so với mẫu được tính theo công thức là:

$$S = \frac{4*1/4+1/2+1/6+2/3}{4+4} = \frac{5}{16} = 0,3125$$

6.2.3.2. Thuật toán phân tích trang văn bản dựa vào mẫu

Dưới đây, chúng tôi trình bày thuật toán phân tích trang văn bản pageANALYSIS* dựa vào mẫu nhờ kỹ thuật phân tích trang văn bản pageANALYSIS [1] theo tiếp cận dưới lên nhờ sử dụng quan hệ Q_θ và việc đánh giá độ lệch cấu trúc văn bản theo mẫu ở mục trên.

Vào: Ảnh văn bản I cần phân tích,

Tập cấu trúc văn bản mẫu tempStructs

Ngưỡng Tolerance

Ra: Cấu trúc trang văn bản cần phân tích pageStruct

Phương pháp: Thuật toán gồm các bước cơ bản sau

❶ Tính biểu đồ tần xuất theo khoảng cách Hausdorff

+ Tách các đối tượng dựa vào chu tuyến ngoài

+ Tính khoảng cách Hausdorff giữa các đối tượng

+ Xây dựng biểu đồ tần xuất theo khoảng cách đã tính

❷ Với biểu đồ tần xuất đã xây dựng lựa chọn ngưỡng θ

❼ Phân tích trang văn bản theo thuật toán pageANALYSIS theo quan hệ Qθ với ngưỡng θ lựa chọn dựa vào biểu đồ tần xuất ở bước 2

❻ Đánh giá lệch của cấu trúc trang văn bản vừa được phân tích ở bước 3 với các cấu trúc trang văn bản mẫu và tìm ra cấu trúc trang tương ứng có độ lệch nhỏ nhất.

❺ Lặp lại bước 2 đến bước 4 chừng nào còn lựa chọn được θ theo các đỉnh biểu đồ tần xuất theo khoảng cách Hausdorff giữa các đối
tượng ảnh.

❻ Chọn ra mẫu có độ lệch nhỏ nhất trong số các độ lệch nhỏ nhất tìm được trong bước 4 ứng với các θ lựa chọn.

❼ Kiểm tra nếu độ lệch nhỏ nhất tìm được trong bước 6 nhỏ hơn ngưỡng Tolerance thì có thể kết luận văn bản cần phân tích có dạng là mẫu có độ lệch nhỏ nhất tương ứng và cấu trúc trang phân tích thu được cấu trúc tương ứng thu được ở bước 2 sau bước phân tích theo thuật toán pageANALYSIS theo quan hệ Qθ. Trong trường hợp ngược lại có thể kết luận văn bản không nằm trong các mẫu văn bản cho trước, để nâng cao chất lượng cho bước sau có thể bổ sung thêm văn bản với các cấu trúc tìm được tương ứng vào tập mẫu cấu trúc văn bản.

Mệnh đề 6.2: Thuật toán phân tích trang văn bản pageANALYSIS* dựa vào mẫu là dừng và cho kết quả đúng.

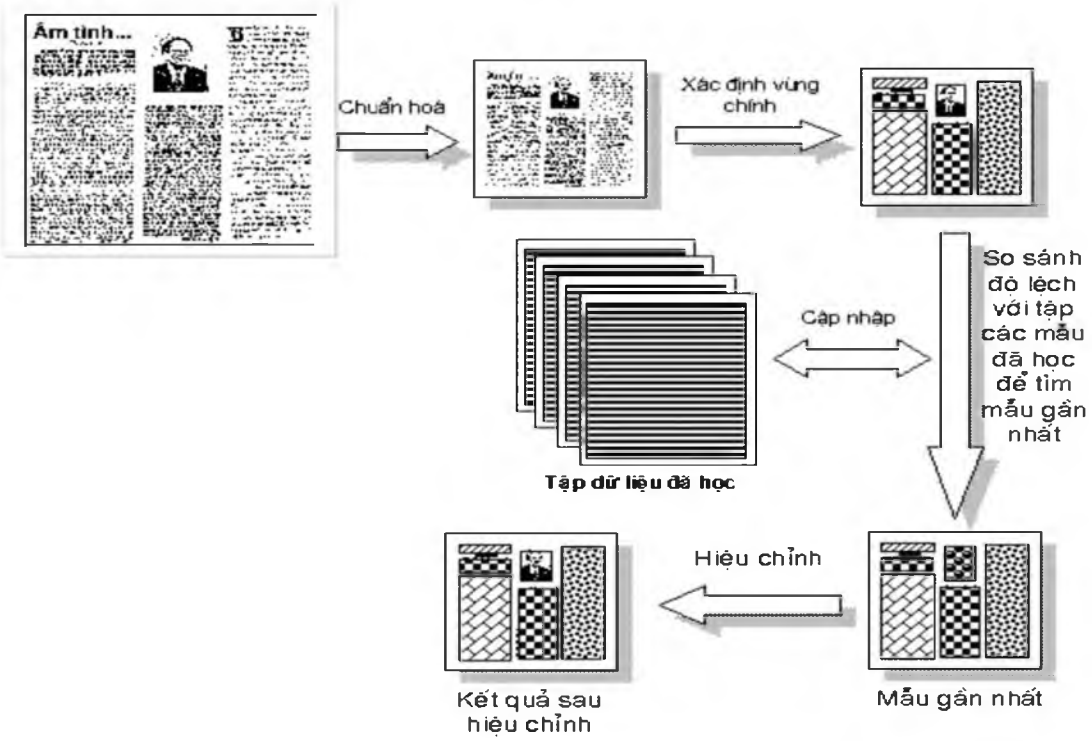
Chứng minh:

Vì số điểm của chu tuyến và đối tượng xác định bởi chu tuyến là hữu hạn nên bước xét duyệt chu tuyến là dừng do đó bước cô lập các đối tượng sẽ dừng. Số các đối tượng thu được là hữu hạn nên việc tính biểu đồ tần xuất theo khoảng cách Hausdorff là dừng. Do đó, các bước lựa chọn ngưỡng θ dựa vào các đỉnh của biểu đồ tần xuất là hữu hạn.

Vì thuật toán phân tích trang văn bản pageANALYSIS* dựa vào mẫu nhờ kỹ thuật phân tích trang văn bản pageANALYSIS theo tiếp cận dưới lên nhờ sử dụng quan hệ Q_θ và việc đánh giá độ lệch cấu trúc văn bản theo mẫu. Tính đúng đắn của thuật toán pageANALYSIS đã được chỉ ra trong [1] và từ mục 6.2.3.1 ta thấy tính đúng đắn của việc đánh giá độ lệch văn bản theo mẫu dẫn đến tính đúng đắn của thuật toán pageANALYSIS*.

Tổng hợp các bước ở trên ta có thuật toán pageANALYSIS* là dừng và cho kết quả đúng .

Các bước tiến hành phân phân vùng và đối sánh mẫu



Hình 6.10: **Các bước tiến hành phân vùng và đối sánh mẫu**

6.3. CẮT CHỮ IN DÍNH DỰA VÀO CHU TUYẾN

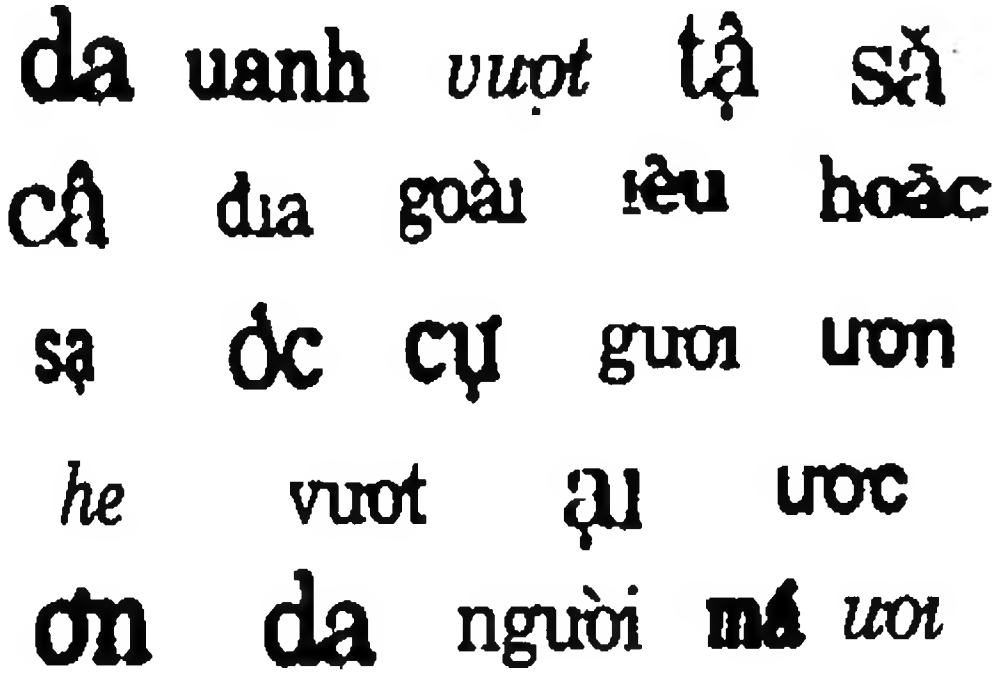
Mỗi một từ hay một cụm chữ dính có thể được xem như một đối tượng ảnh. Mục này đề xuất một thuật toán cắt chữ Việt in dính nhờ việc sử dụng tính chất của chu tuyến và vị trí tương hỗ giữa chúng. Việc sử dụng tính chất của chu tuyến, vị trí tương hỗ giữa chúng và một số tính chất khác liên quan trong việc cắt chữ dính, qua thực nghiệm thể hiện hiệu quả hơn đối với lớp các chữ Việt in dính đã được áp dụng các phương pháp truyền thống.

6.3.1. Đặt vấn đề

Một trong những điểm mấu chốt của vấn đề nhận dạng chữ in nói chung và chữ Việt in nói riêng là phải cô lập được chúng. Việc cô lập các chữ được tiến hành sau khi tiến hành phân tích trang để tách các khối, từ các khối tách ra các dòng, từ các dòng tách ra các từ và các từ tách ra các ký tự.

Phương pháp chiếu ngang VPP (Vertical Project Profile) hay biểu đồ tần suất ngang VH (Vertical Histogram) là phương pháp khá phổ biến được áp dụng trong các hệ OCR (Optical Character Recognition). Nhưng phương

pháp này thường bị hạn chế khi các ký tự bị dính (touch) hay bị chèn (overlap) [35,36,54], đặc biệt đối với chữ Việt với các tầng mũ và dấu điều này thường hay xảy ra.



Hình 6.11. **Ví dụ về chữ Việt in bị dính**

Hình 6.11 là một số ví dụ về kí tự chữ Việt in bị dính và bị chèn mà phần mềm VnDOCR(*) phiên bản 2.0 sử dụng phép cắt theo VPP nên phép cắt không chính xác. Đối với những chữ chất lượng xấu như thế, những chỗ dính có diện tích tiếp xúc lớn hơn diện tích tiếp xúc của các phần của một chữ.

Việc khắc phục nhược điểm này của các phương pháp VPP và VH được nhiều tác giả giải quyết cho đến những thời điểm hiện nay thể hiện tính thời sự của vấn đề trong đó có việc sử dụng chu tuyến kết hợp với các phương pháp cho chữ dính và đặc biệt là chữ viết tay.

Mặt khác, trong các nghiên cứu trước các tác giả đã chỉ ra rằng mỗi một đối tượng ảnh được giới hạn bởi duy nhất một chu tuyến ngoài và các chu tuyến trong. Trong phạm vi nghiên cứu này mỗi một từ hay một cụm chữ dính là một thành phần liên thông trong ảnh và do đó nó là một đối tượng ảnh.

Xuất phát từ hoàn cảnh đó, mục này trình bày một kỹ thuật cắt chữ Việt in dính, nhờ việc sử dụng các tính chất của chu tuyến và vị trí tương hỗ giữa chúng, với đối tượng trực tiếp là các chữ in dính mà phần mềm VnDOCR2.0 sau khi sử dụng phương pháp cắt chữ VPP không cắt được.

(*) Phần mềm đoạt giải nhất "Sáng tạo khoa học kỹ thuật Việt Nam" (giải VIFOTECH trước đây) năm 1999.

6.3.2. Một số khái niệm cơ bản

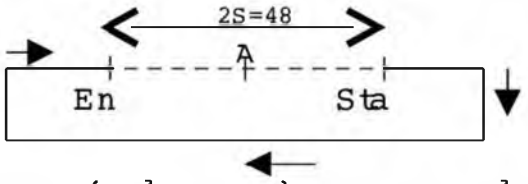
Cặp điểm thắt: Là cặp điểm nằm trên chu tuyến ngoài của một từ hay một cụm chữ dính mà khoảng cách giữa chúng nhỏ hơn nhiều so với khoảng cách dọc theo chu tuyến giữa chúng.

Khoảng cách dọc theo chu tuyến ngoài giữa hai điểm p_i và p_j được định nghĩa là: $dist_B(p_i, p_j)$ là khoảng cách nhỏ nhất từ p_i đến p_j dọc theo chu tuyến. Giá trị w_{ij} cũng có thể được tính toán bởi hàm năng lượng biên W: B(E) $\rightarrow$ R dọc theo biên B(E) của E. Ta có: $dist_B(p_i,p_j)= \min(|W(p_i)-W(p_j)\}, \Sigma_W-|W(p_i)-W(p_j)|)$

Ở đây, Σ_W – tổng năng lượng biên dọc theo B(E).

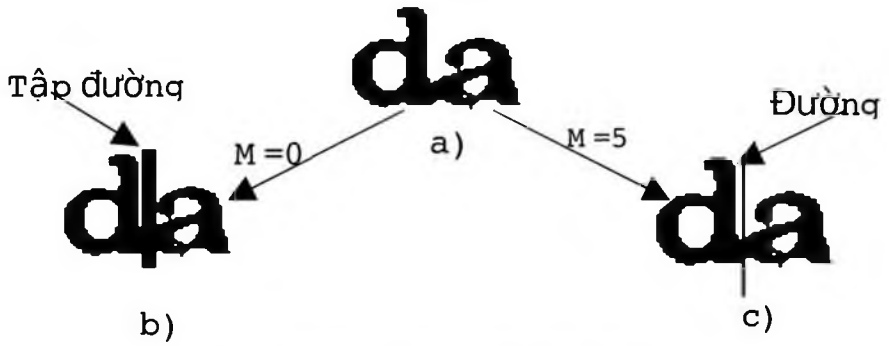
Ngưỡng thắt (L): Là tỷ số khoảng cách tối đa giữa 2 điểm dọc theo chu tuyến và khoảng cách giữa chúng trong cặp điểm thuộc chu tuyến đang xét được tạm coi là cặp điểm thắt.

Ngưỡng quét (S): Là số điểm cách điểm đang xét về 2 phía trên đường đi đang xét. Thí dụ: S= 24, và điểm quét tại điểm A thì ngưỡng quét được mô tả như hình dưới đây:



Hình 6.12. **Số điểm cách về hai phía tại điểm quét A**

Mật độ cắt (M): Là khoảng cách tương đối giữa 2 cặp điểm thắt liền nhau về 2 phía. Thí dụ M = 5 thì tất cả các điểm cắt liền kề nhau sẽ cách nhau 1 khoảng tối thiểu là 5 điểm ảnh. Mật độ cắt rất quan trọng. Thật vậy, trong một từ có thể có nhiều điểm thắt và các điểm thắt đó có thể rất gần nhau (cách nhau khoảng 1,2,3,5,6 pixel) điều này làm cho nét cắt sẽ không mảnh vì vậy cần phải có khoảng cách giữa các điểm thắt. Ví dụ: Khi M = 0 thì chương trình sẽ nhận ra 1 loạt các điểm thắt và đưa ra các quyết định cắt như trong Hình 5.13.b. Khi M= 5 thì chương trình đưa ra quyết định cắt như trong Hình 5.13.c



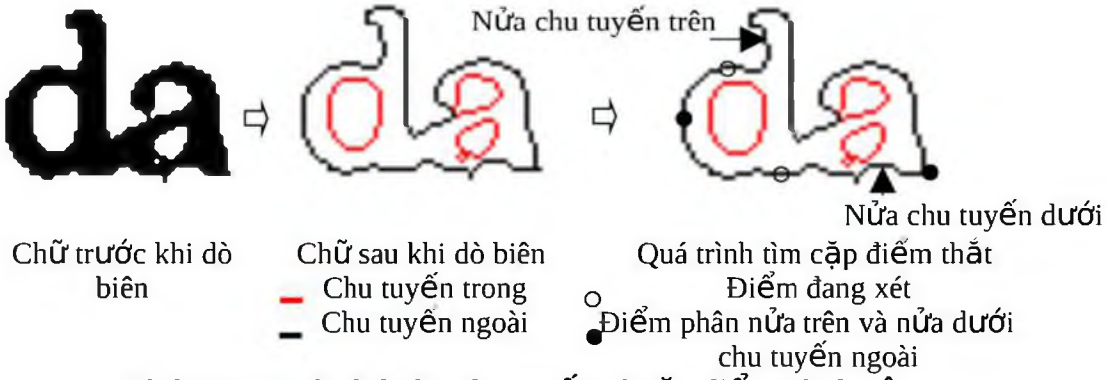
Hình 5.13. **Ví dụ về đường cắt và mật độ cắt**

6.3.3. Thuật toán cắt chữ in dính dựa vào chu tuyến

6.3.3.1. Phân tích bài toán

Ý tưởng chính của phần này về việc cắt chữ in dính là "**gần nhà, xa ngõ**", dựa vào các đặc điểm của các cặp điểm thắt để ra quyết định cắt.

Với mỗi một từ hay một cụm chữ dính, trước tiên ta tiến hành tách ra các chu tuyến, sau đó dựa vào chu tuyến ngoài để tìm ra các cặp điểm thắt. Mỗi chu tuyến đều tồn tại một chu tuyến đối ngẫu. Trong các nghiên cứu trước các tác giả đã chỉ ra rằng mỗi đối tượng ảnh, ở đây là một từ hay một cụm chữ dính, tồn tại duy nhất một chu tuyến ngoài.



Hình 6.14. **Quá trình tìm chu tuyến và cặp điểm xét duyệt**

Xuất phát từ tư tưởng trên, với mỗi một từ hay cụm chữ dính ban đầu sẽ được tiến hành dò biên để tìm ra các chu tuyến và tính chất tương ứng, các thông tin về chu tuyến ngoài và chu tuyến trong tìm được cùng các thông tin về ngưỡng quét, ngưỡng thắt, mật độ quét được lưu lại.

Quá trình tìm các cặp điểm thắt của đối tượng bắt đầu từ việc xét duyệt các điểm thuộc chu tuyến ngoài. Cặp điểm xét duyệt được xác định bởi một điểm nằm nửa phần trên chu tuyến và điểm còn lại nằm ở phần nửa dưới chu tuyến ngoài (Hình 6.14). Việc phân loại nửa trên và nửa dưới chu tuyến xác định bởi điểm bên trái nhất và điểm bên phải nhất của đối tượng.

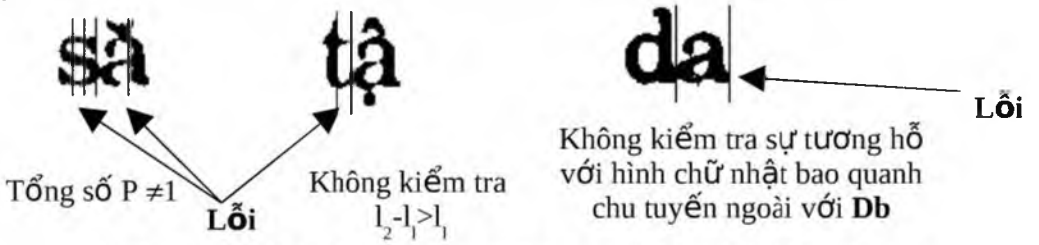
Với mỗi điểm xét duyệt nằm nửa trên chu tuyến ngoài ta tìm một điểm tương ứng nằm nửa dưới chu tuyến có khoảng cách nhỏ nhất đến điểm đang xét và thỏa mãn ngưỡng quét (S). Điều này nhằm tránh việc xét những điểm biên liền sát với điểm đang xét dọc theo chu tuyến.

Nếu khoảng cách từ điểm đang xét đến điểm dưới chu tuyến tìm được có tỷ số so với khoảng cách hai điểm này tính dọc theo chu tuyến nhỏ hơn ngưỡng (L) thì cặp điểm đang xét được xem là cặp điểm thắt và được lưu trữ lại.

Tiếp theo, các cặp điểm thắt tìm được sẽ được kiểm tra xem có thỏa mãn điều kiện là cặp điểm cắt hay không? hay nói cách khác đường nối giữa 2 điểm có là đường phân ranh giới của các chữ dính hay không?

Trước hết, vị trí của từng cặp điểm thắt sẽ được xét duyệt, nếu độ lệch hoành độ của chúng ở ***quá xa*** hay ***quá gần*** (nhằm tránh trường hợp cắt ngang) thì cặp điểm đó sẽ bị loại. Trong trường hợp ngược lại, cặp điểm biên (kí hiệu là Db) sẽ được truyền theo lời gọi hàm **FindCutPoint(...)**. Hàm **FindCutPoint(...)** sẽ thực hiện tìm đếm số điểm thuộc nửa dưới của chu tuyến ngoài có hoành độ ***gần với*** hoành độ của điểm thuộc nửa trên chu tuyến ngoài trong cặp điểm đang xét (kí hiệu số điểm biên tìm được là B). Nếu số điểm B tìm được > 1 thì hàm **FindCutPoint(…)** trả về giá trị FALSE, do đó cặp điểm thắt Db sẽ bị loại khỏi quá trình xét duyệt sau đó.

Ngược lại, hàm **FindCutPoint(…)** tính độ cao l_1 của 2 điểm biên trong Db, và tính độ cao l_2 của điểm chuẩn trong Db với điểm B vừa tìm được. Nếu l_2-l_1>l_1 thì hàm **FindCutPoint**(...) cũng sẽ trả về giá trị FALSE. Ngược lại, cặp điểm Db sẽ được kiểm tra xem có gần với cạnh trái và cạnh phải của hình chữ nhật bao quanh chu tuyến ngoài đang xét hay không. Nếu chúng quá gần thì hàm **FindCutPoint(..)** trả về giá trị FALSE, ngược lại hàm **FindCutPoint(..)** trả về giá trị TRUE. Tiếp nữa, hàm **FindCutPoint(…)** sử dụng đến mật độ M nhằm xem xét khoảng cách tương đối giữa các cặp điểm thắt theo hoành độ. Nếu chúng lớn hơn điều kiện mật độ cắt (M) hàm **FindCutPoint(…)** trả về giá trị TRUE. Do đặc điểm của chữ Việt là các chữ có râu đều có râu nằm về bên phải nên thứ tự ưu tiên đối với mật độ cắt là từ ***phải sang trái*** để tránh trường hợp khi cắt các chữ "ư" "ơ" v.v.. để lại nét cho chữ viết sau.



Hình 6.15. **Một số hình ảnh minh hoạ về các điều kiện ở trên**

Cặp điểm này được gọi là **cặp điểm cắt**. Để tăng độ chính xác cho quá trình cắt, hoành độ của 2 điểm trong cặp điểm thắt được lấy trung bình, sau đó, hoành độ mới này sẽ hợp với tung độ của điểm chuẩn trong cặp điểm thắt để hình thành lên một điểm mới (gọi là điểm cắt tạm thời) và nó được lưu giữ nhằm phục vụ cho các bước tiếp theo trong quá trình cắt chữ.

Cuối cùng, sau khi đã xác định được tập điểm cắt tạm thời trong tập các cặp điểm biên gần nhau, hàm **VerifyCutPoint()** sẽ được gọi nhằm thẩm tra lại dựa theo quá trình phân tích độ rộng của kí tự trong từ và vị trí tương hỗ giữa chu tuyến trong và chu tuyến ngoài trên cơ sở phân tích các mẫu cơ bản nhờ sử dụng các quy tắc, các dấu hiệu như:

l: Số chu trình (loops)

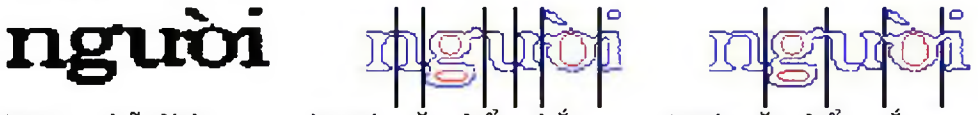
j: Số điểm khớp (junction points)

e: Số điểm ngoặt (turning points)

f: Số điểm kết thúc (end points)

t: Hướng (trên, dưới, trái, phải)

Để biểu diễn các ký tự, từ đó tìm ra điểm cắt phù hợp, chẳng hạn các điểm cắt thường được giới hạn bởi trước và sau chu tuyến trong của đối tượng (Hình 6.16).



a) Cụm chữ dính b) Các cặp điểm thắt tìm được c) Các cặp điểm cắt sau khi tính tới vị trí tương hỗ của chu tuyến trong

Hình 6.16. **Thực hiện VerifyCutPoint có tính đến vị trí tương hỗ của chu tuyến trong**

6.3.3.2. Thuật toán CutCHARACTER cắt chữ in dính dựa vào chu tuyến

Ban đầu các chu tuyến và các tính chất tương ứng của chu tuyến sẽ được phát hiện. Bước tiếp theo là dựa vào nửa chu tuyến ngoài trên và nửa chu tuyến ngoài dưới của cụm chữ dính để tìm ra các cặp điểm thắt theo điều kiện về các cặp điểm thắt. Các cặp điểm thắt sẽ được chính xác hóa nhờ các thông tin về điểm cùng hoành độ, mật độ cắt, độ rộng của chữ v.v.. và cuối cùng là quyết định cặp điểm thắt là cặp điểm cắt nhờ sử dụng các quy tắc, dấu hiệu như: Số chu trình (loops), số điểm khớp (junction points), số điểm ngoặt (turning points), số điểm kết thúc (end points), hướng (trên, dưới, trái, phải) và vị trí tương hỗ của chu tuyến trong.

Thuật toán ***CutCHARACTER*** tìm cặp điểm cắt dựa vào chu tuyến (theo tính liên thông) nên sẽ khắc phục được các lỗi cắt chữ bị chèn (overlap) của thuật toán cắt chữ VPP hay VH.

Do các cặp điểm thắt được tìm nằm ở nửa trên và nửa dưới của chu tuyến ngoài của cụm từ, chữ dính cần tách, nên thuật toán ***CutCHARACTER*** có khả năng cắt được các chữ dính không theo chiều thẳng đứng (cắt xiên) và do đó mở ra khả năng ứng dụng cho việc cắt chữ viết tay.

Chúng tôi đã áp dụng kỹ thuật cắt chữ dính nhờ việc sử dụng tính chất của chu tuyến và vị trí tương hỗ giữa chúng đối với lớp các ký tự mà phần mềm VnDOCR 2.0 sau khi sử dụng phương pháp Vertical Project Profile (VPP) hay Vertical Histogram (VH) là phương pháp khá phổ biến được áp dụng trong các hệ OCR (Optical Character Recognition).

Trong số gần 90 cụm từ mà VnDOCR không cắt được, chúng tôi đã cho áp dụng kỹ thuật cắt chữ này và đã cắt được hơn 80 cụm chữ dính. Điều đó mở ra khả năng tích hợp của kỹ thuật với các phương pháp VPP và VH truyền thống.

Các cụm chữ dính mà thuật toán cắt được tương ứng với các dạng phông chữ kiểu không chân và có chân Arial, Avant, Times, Courier,... với kích thước của các ký tự từ 8 đến 72 điểm và các thuộc tính như bình thường, đậm, nghiêng, hay kết hợp đậm-nghiêng.

6.4. NHẬN DẠNG CHỮ VIẾT

Cơ sở nhận dạng ký tự là phân biệt và phân tích các mẫu cơ bản để biểu diễn các ký tự. Các đoạn cung của các ký tự được đặc trưng bởi độ dài và độ cong, các đoạn thẳng được ký hiệu bởi hướng, đầu và cuối điểm của chúng để biểu diễn các ký tự (xem [1,3,25,44,60,61]). Vì các tham số này không thể dùng một cách có hiệu quả để tổng quát hóa cấu trúc chữ. Phương pháp chung được dùng cho việc nhận dạng ký tự được xét như sau:

Cho tập hợp η={A,B,...X,Y,Z}. Bài toán đặt ra là phân hoạch η thành các tập con ωi , i=1,...,26, sao cho:

$\omega i \cap \omega j = \varnothing$, $\forall i,j = 1,\ldots,26$, $i \neq j$ và $\bigcup_{i=1}^{26} \omega_i = \eta$.

Bằng cách sử dụng một loạt các quy tắc, các dấu hiệu như:

- l : Số chu trình (loops)
- j: Số điểm khớp (junction points)
- e: Số điểm ngoặt (turning points)

- f: Số điểm kết thúc (end points)
- t: Hướng (trên, dưới, trái, phải)

Ta phân hoạch tập đối tượng đã cho thành các tập con. áp dụng tiếp các quy tắc, dấu hiệu này, để phân tiếp các tập con thành các tập nhỏ hơn. Thí dụ với tập đã cho, dùng quy tắc e (số điểm ngoặt) ta phân thành 3 tập nhỏ:

η_1= {A,D,O,P,Q,R}, η_2= {B}, η_3= {C,E,F,..} tương ứng với số điểm ngoặt khác nhau.

Nếu chưa đủ độ tin cậy, ta dùng thêm hướng t để phân tiếp. Thí dụ, dùng thêm t cho tập η_1 ta thu được 5 tập nhỏ:

η_{11}= {A,R}, η_{12}= {D}, η_{13}= {O}, η_{14}= {Q}, η_{15}= {P}.

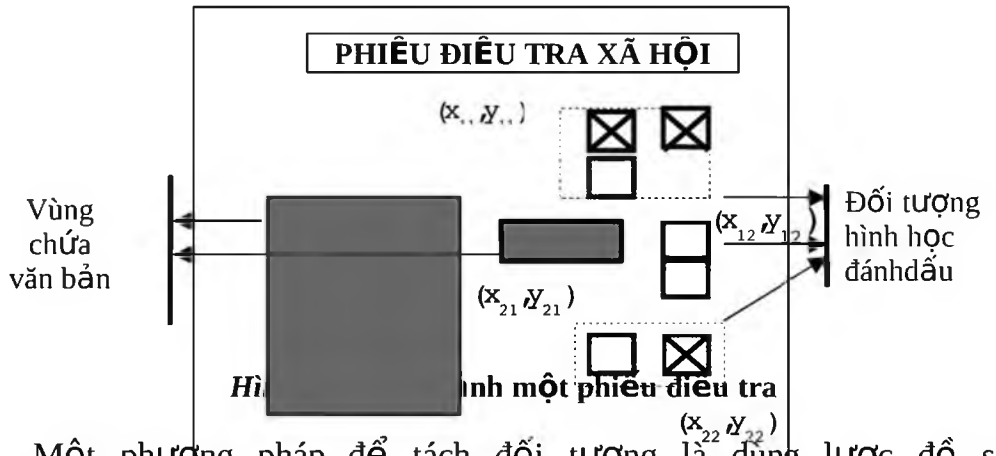
Nếu các tập thu được chưa phải là đơn nhất, ta áp dụng thêm các quy tắc khắc như j để làm mịn nó. Với tập η_{11}, áp dụng quy tắc j ta chia nó thành 2 tập {A} và {R} vì chữ A có 2 điểm khớp (chạc 3) mà chữ R chỉ có một. Cuối cùng, ta có một phân hoạch không gian theo yêu cầu và mỗi ký tự có thể biểu diễn bởi η_{K_i}, trong đó K_i là xâu các số nguyên, K là xâu các ký tự *j, l,e* và *t*, mỗi ký tự nhận giá trị bằng số lượng của chúng.

6.5. TÁCH CÁC ĐỐI TƯỢNG HÌNH HỌC TRONG PHIẾU ĐIỀU TRA DẠNG DẤU

6.5.1. Giới thiệu

Có thể thấy rõ một phiếu điều tra không đơn thuần chỉ có các vùng chứa đối tượng hình học, mà còn cả vùng chứa ký tự, vùng ảnh v.v.. Hơn nữa, bản thân trong vùng chứa đối tượng hình học lại có thể có nhiều loại đối tượng, với các đặc tính và định hướng ngang dọc khác nhau (Hình 5.16). Chính vì vậy giai đoạn tách từng đối tượng ra khỏi vùng khá quan trọng ảnh hưởng tới chất lượng toàn bộ hệ thống nhận dạng. Khi cô lập đúng một đối tượng ra khỏi tổng thể vùng thì khi đó mới có thể tiến hành tách dấu chứa trong nó một cách chính xác.

Để tiến hành nhận dạng các phiếu điều tra và kết nối các phiếu điều tra thành một cơ sở dữ liệu, cần định vị chính xác các vị trí của đối tượng (ô dấu). Có 2 pha cần thiết: Hiệu chỉnh các vị trí của phiếu điều tra trùng với phiếu mẫu và tách các ô trong phiếu mẫu chính xác. Để giải quyết bài toán đầu tiên có thể hiệu chỉnh góc lệch bằng biến đổi Hough, phép chiếu nghiêng và nối tâm các của các chữ gần nhau. Sau đó dùng biểu đồ tần suất và phương pháp bình phương tối thiểu để hiệu chỉnh lề. Bài toán thứ 2 có thể sử dụng chu tuyến trong của đối tượng hình học.



Hình một phiếu điều tra

Một phương pháp để tách đối tượng là dùng lược đồ sáng (Histogram). Phương pháp này đòi hỏi các đối tượng trên một dòng và các dòng chứa các hình phải tách rời nhau một khoảng đủ lớn. Điều này tạo điều kiện cho việc tìm ra dòng phân cách giữa các dòng hình và giữa các hình trên cùng một dòng. Song trên thực tế việc tìm đường phân cách giữa chúng theo nghĩa thông thường là rất khó do hình nhoè hai hình có thể dính lại với nhau bởi nhiễu. Hơn nữa, chính một số dạng mẫu dấu điều tra lại được thiết kế dính liền nhau giống như biểu bảng (Hình 6.17). Mục này trình bày phương pháp tách các đối tượng hình học mềm dẻo hơn bằng việc phát hiện các đối tượng hình học một cách tự động nhờ sử dụng chu tuyến trong và một số kiểu xấp xỉ đa giác bởi các hình cơ sở.

6.5.2. Tách các đối tượng nhờ sử dụng chu tuyến

Một trong những bước cơ bản của việc tách các đối tượng hình học là phải xác định được chúng. Mệnh đề 6.3 dưới đây cho ta điều kiện cần để xác định đối tượng cần tách.

Mệnh đề 6.3

Tồn tại một chu tuyến trong bên trong các đối tượng hình học dạng dấu (dùng để đánh dấu).

Chứng minh:

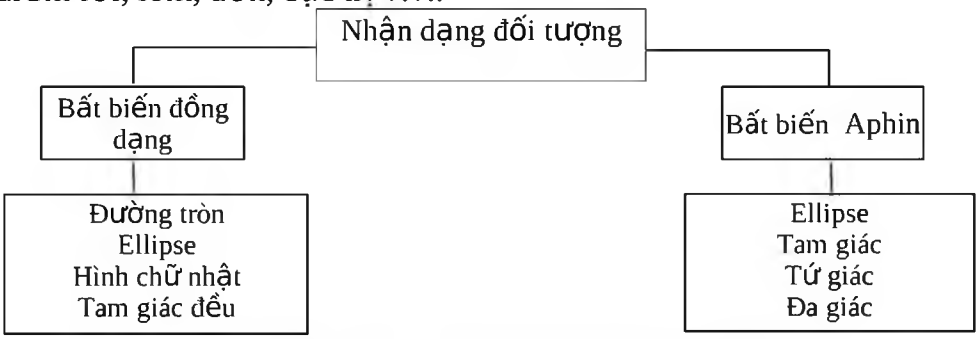
Do đặc tính của các đối tượng hình học dạng dấu là nơi để người ta đánh dấu vào trong do đó tồn tại vùng trắng(vùng để đánh dấu) bên trong đối tượng hình học dạng dấu. Theo định nghĩa chu tuyến trong các đối tượng này đều tồn tại chu tuyến trong .

Do bản chất của công việc nhận dạng dấu là chỉ quan tâm đến đối tượng có được đánh dấu hay không (bên trong). Hơn nữa, do cấu trúc của phiếu điều tra rất có thể nhiều đối tượng cần nhận dạng lại được bố trí kề liền nhau (Hình 6.17), do đó ta chỉ cần quan tâm đến chu tuyến trong của đối tượng, chu tuyến xác định đối tượng dấu.

Chu tuyến trong xác định đối tượng tìm được là một dãy các điểm liên tiếp đóng kín. Sử dụng các thuật toán đơn giản hoá như Douglas Peucker, Band Width, Angle v.v.. ta sẽ thu được một polyline hay nói khác đi là thu được một đa giác xác định đối tượng dấu. Vấn đề là ta cần phải xác định xem đối tượng có phải là đối tượng cần tách hay không? Như ta đã biết một đa giác có thể có hình dạng tựa như một hình cơ sở, có thể có nhiều cách tiếp cận xấp xỉ khác nhau. Cách xấp xỉ dựa trên các đặc trưng cơ bản sau:

Đặc trưng toàn cục: Các mô men thống kê, số đo hình học như chu vi, diện tích, tập tối ưu các hình chữ nhật phủ hay nội tiếp đa giác v.v..

Đặc trưng địa phương: Các số đo đặc trưng của đường cong như góc, điểm lồi, lõm, uốn, cực trị v.v..



***Hình 6.18.* Sơ đồ phân loại các đối tượng theo bất biến**

Dựa trên các cách tiếp cận được đưa ra trong các tài liệu chúng tôi lựa chọn tiếp cận với các đặc trưng toàn cục và đưa ra sơ đồ xấp xỉ theo các bất biến đồng dạng và bất biến aphin. Việc xấp xỉ tỏ ra rất có hiệu quả đối với một số hình phẳng đặc biệt như tam giác, đường tròn, hình chữ nhật, hình vuông, hình ellipse, hình tròn và một đa giác mẫu.

6.6. TÁCH BẢNG DỰA TRÊN TẬP CÁC HÌNH CHỮ NHẬT RỜI RẠC

Một trong những vấn đề cơ bản của nhận dạng các trang văn bản nói chung và các trang văn bản ở dạng bảng nói riêng là phải phân tích được chúng. Đối với các trang văn bản thông thường thì phải hiểu phạm vi, cấu trúc của các khối văn bản. Trong các trang hoặc khối văn bản ở dạng bảng thì phải hiểu và phân tích được các ô chứa trong bảng. Vì chỉ khi nào phân tích được bảng một cách chính xác thì khi đó mới có thể tiến hành nhận dạng các thông tin trong các ô trong nó một cách chính xác và cũng chỉ có phân tích được bảng một cách chính xác thì sau quá trình nhận dạng các ô mới được trả lại cấu trúc của nó một cách chính xác.

Country	Address	Telephone	Fax
Algeria	12 Phan Chu Trinh	8253865	8260830
Argentina	8th floor, Daeha Centre, 360 Kim Ma St.	8315578 / 8315262	8315262
Australia	Van Phuc Quarter	8317755	8317711
Bangladesh	Apt 101 - 108 A1 Van Phuc Diplomatic Quarter	8231625	8231628
Belgium	48 Nguyen Thai HocSt	8452263	8457165
Brazil	14 Thuy Khue St	8432544	8432542
Bulgaria	Van Phuc Quarter	8252908	8460856
Cambodia	71 A Tran Hung Dao St.	8253788	8265225
Canada	31 Hung Vuong St.	8235500	8235333
China	46 Hoang Dieu St.	8453736	8252826
Cuba	65 Ly Thuong Kiet St	8252426	8252426

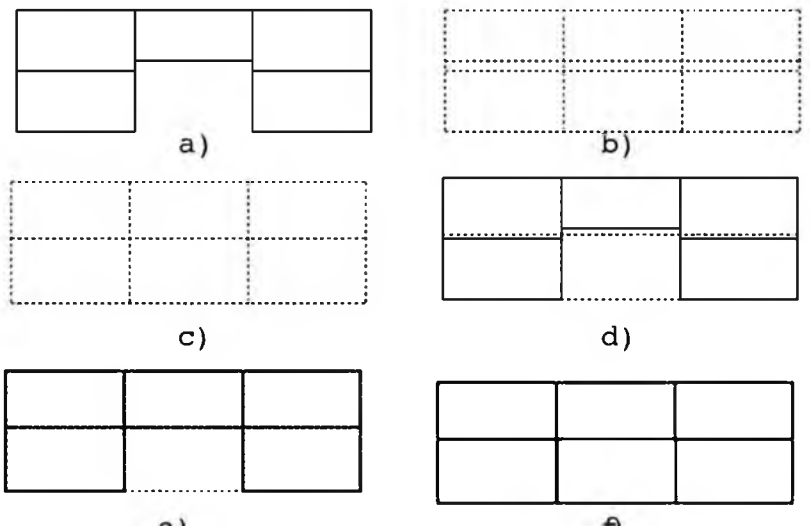
Hình 6.19. **Khối văn bản ở dạng bảng**

Trong mục này chúng tôi đưa ra cách phân tích bảng theo tiếp cận từ dưới lên. Ban đầu các ô trong bảng sẽ được phát hiện nhờ kỹ thuật tách đối tượng hình học với dạng đối tượng là hình chữ nhật. Dựa trên các hình chữ nhật tìm được, chúng tôi xây dựng lưới tựa hình chữ nhật. Sau đó tiến hành hiệu chỉnh lưới dựa trên khoảng cách ngưỡng cho trước. Từ tập lưới đã sửa tiến hành hiệu chỉnh lại tập hình chữ nhật. Dựa vào tập hình chữ nhật và lưới có thể phát hiện ra các ô thiếu. Các tiếp cận này có thể phân tích và hiệu chỉnh đối với các bảng có các ô được ghép (merge) với nhau và cả những bảng có đường nét không đủ. Bên cạnh đó, chúng tôi cũng đưa ra các đánh giá như thế nào là tập hình chữ nhật có thể chuyển đổi thành bảng. Với việc đánh giá như vậy có thể dẫn tới việc nhận dạng đối tượng bảng một cách tự động.

6.6.1. Phân tích bài toán

Để phân tách được bảng chúng tôi thực hiện phân tích từ dưới lên. Đầu tiên, tiến hành tách ra các chu tuyến sau đó dựa vào chu tuyến trong để nhận ra các hình chữ nhật. Chu tuyến là dãy liên tiếp các điểm biên của ảnh. Mỗi chu tuyến đều tồn tại một chu tuyến đối ngẫu. Nếu chu tuyến có độ dài nhỏ hơn chu tuyến đối ngẫu thì ta gọi nó là chu tuyến bên trong. Trong trường hợp ngược lại thì đó là chu tuyến ngoài. Từ các chu tuyến trong tiến hành nhận dạng để tìm ra tập các hình chữ nhật.

Quá trình xây dựng bảng được tiến hành từ tập các hình chữ nhật tách được từ ảnh. Tập các hình chữ nhật là tập liên thông hoặc được lựa chọn trong một hình chữ nhật. Dựa vào tập các hình chữ nhật chúng tôi xây dựng được các lưới tựa các hình chữ nhật (lưới là tập các toạ độ ngang dọc).

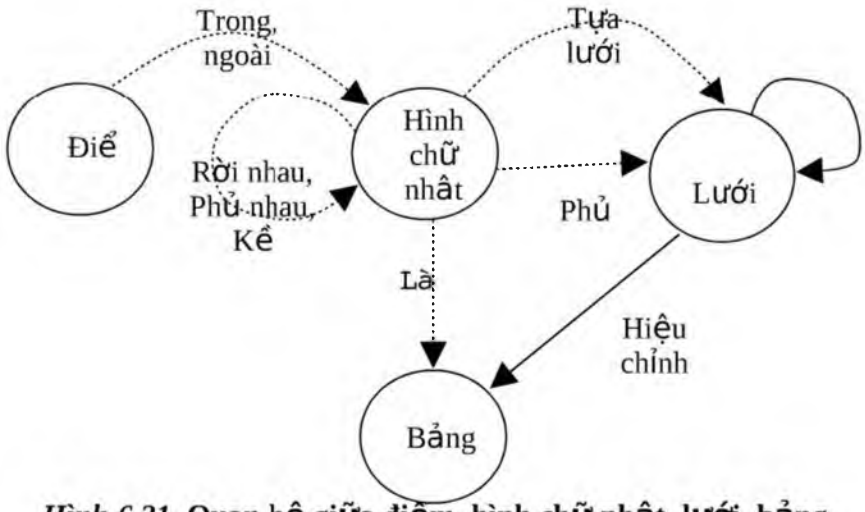


a) Tập hình chữ nhật ban đầu, b) Dựng lưới dựatrên tập các hình chữ nhật,
c) Hiệu chỉnh lưới, d) Hiệu chỉnh bảng dựa trên lưới, e) Xác định ô khuyết thiếu
f) Sau khi đã bổ sung thêm ô.

***Hình 6.20*. Quá trình hiệu chỉnh bảng từ tập hình chữ nhật**

Bước tiếp theo là tiến hành hiệu chỉnh tập lưới sao cho bất cứ hai toạ độ ngang hoặc dọc không quá gần nhau. Dựa vào lưới đã hiệu chỉnh có thể hiệu chỉnh tập hình chữ nhật sao cho tất cả các đỉnh của tập các hình chữ nhật nằm trên lưới, cách hiệu chỉnh được tiến hành cho từng hình chữ nhật. Việc hiệu chỉnh các hình chữ nhật nằm trên lưới sẽ cho phép ta phát hiện và bổ sung những hình chữ nhật còn khuyết.

Việc đánh giá khả năng tách bảng được thực hiện như sau: Tính tỉ số giữa số ô lưới được phủ bởi các hình chữ nhật vừa được hiệu chỉnh với tổng số ô lưới được tạo ra. Nếu tỷ số này lớn hơn ngưỡng cho trước thì việc chuyển đổi bảng được coi như thành công.



***Hình 6.21*. Quan hệ giữa điểm, hình chữ nhật, lưới, bảng**

Để thực hiện việc chuyển đổi từ tập các hình chữ nhật thành bảng chúng ta cần xác định các mối quan hệ nội tại trong các hình như tập các điểm tạo ra một ảnh, tập các hình vuông, lưới và bảng dựa trên lưới.

Thuật toán xác định và hiệu chỉnh đối tượng bảng

Ban đầu, các ô trong bảng sẽ được phát hiện nhờ kỹ thuật tách đối tượng hình học. với dạng đối tượng là hình chữ nhật. Dựa trên các hình chữ nhật tìm được, xây dựng lưới tựa hình chữ nhật, sau đó tiến hành hiệu chỉnh lưới dựa trên khoảng cách ngưỡng cho trước. Từ tập lưới đã sửa

tiến hành hiệu chỉnh lại tập hình chữ nhật. Dựa vào tập hình chữ nhật và lưới có thể phát hiện ra các ô thiếu. Trên cơ sở đó đánh giá như thế nào là tập hình chữ nhật có thể chuyển đổi thành bảng.

Với việc đánh giá như vậy có thể dẫn tới việc nhận dạng đối tượng bảng một cách tự động. Qua thực nghiệm chúng tôi thấy cách tiếp cận này có thể phân tích và hiệu chỉnh đối với các bảng có các ô được nối (merge) với nhau và cả những bảng có đường nét không đủ.

6.7. PHÁT HIỆN ĐỐI TƯỢNG CHUYỂN ĐỘNG

Hướng tiếp cận phổ biến để phát hiện đối tượng chuyển động là so sánh khung hình hiện tại với khung hình liền trước. Kỹ thuật này rất tốt trong việc nén video khi mà chúng ta chỉ cần đánh giá sự thay đổi và chỉ cần ghi lại những thay đổi, mà không cần phải ghi lại toàn bộ khung hình đó [5,6]. Nhưng kỹ thuật này chưa được tốt cho các ứng dụng phát hiện đối tượng chuyển động. Dưới đây, chúng ta sẽ xem xét trường hợp phát hiện đối tượng chuyển động dựa trên hướng tiếp cận trừ khung hình hiện tại với khung hình liền trước.

6.7.1. Phát hiện đối tượng chuyển động dựa theo hướng tiếp cận trừ khung hình liền kề

Giả sử, chúng ta có tệp ảnh động video định dạng avi, với các thông số thuộc tính như Hình 6.22.

Các khung hình được lấy từ file video có dạng RGB 24 bit, và được chuyển sang ảnh 256 cấp xám. Chúng ta gọi khung hình hiện thời là Ic, khung hình liền trước Ip. Khung hình Igc, Igp cùng là ảnh xám 256 màu, được chuyển như sau:

//Ảnh hiện thời

$ColorI_c=(I_c(i,j).Red + I_c(i,j).Green + I_c(i,j).Blue)/3;$

$Ig_c(i,j).Blue= ColorI_c;$

$Ig_c(i,j).Green= ColorI_c;$

$Ig_c(i,j).Red= ColorI_c;$

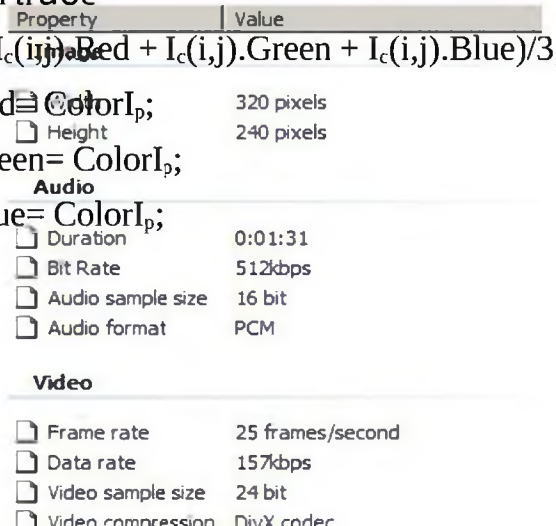
//Ảnh liền trước

$ColorI_p=(I_c(i,j).Red + I_c(i,j).Green + I_c(i,j).Blue)/3;$

$Ig_p(i,j).Red= ColorI_p;$

$Ig_p(i,j).Green= ColorI_p;$

$Ig_p(i,j).Blue= ColorI_p;$



Hình 6.22. **Thuộc tính của file video dạng avi**

Tiếp theo, Igc, Igp được trừ theo từng điểm ảnh, và được so sánh với ngưỡng. Nếu giá trị tuyệt đối nhỏ hơn giá trị ngưỡng thì coi là điểm giống nhau, ngược lại coi là khác nhau. Tức là, tại vị trí i,j:

```
if(abs(ColorIp- ColorIc)<lThreshold)   //giống nhau
{
      Iwb(i,j).Red=0;
      Iwb(i,j).Green=0;
      Iwb(i,j).Blue=0;
} else   //khác nhau
{
      Iwb(i,j).Red=255;
      Iwb(i,j).Green=255;
      Iwb(i,j).Blue=255;
}
```

Iwb là ảnh đen trắng thể hiện vùng khác nhau giữa 2 khung hình, những điểm khác nhau sẽ có màu trắng, ngược lại có màu đen. Dưới đây là một số hình minh họa của kỹ thuật trừ ảnh liền kề với ngưỡng được đặt là 20.

a) Ảnh khung hình liền trước I_p

b) Ảnh khung hình hiện thời I_c

c) Ảnh đen trắng Iwb

***Hình 6.23.* Ảnh khung hình liền trước và hiện thời có sai khác ít.**
a) Ảnh khung hình liền trước, b) Ảnh khung hình hiện thời, c) Ảnh đen trắng dùng kỹ thuật trừ ảnh liền kề, với ngưỡng là 20.

a) Ảnh khung hình liền trước I_p

b) Ảnh khung hình hiện thời I_c

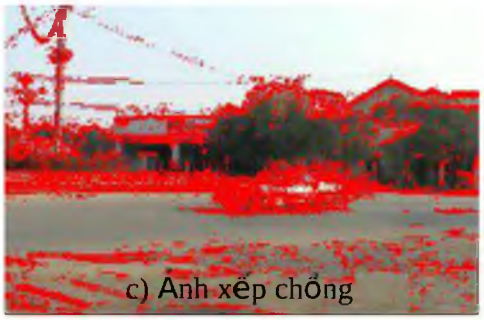
c) Ảnh đen trắng Iwb

***Hình 6.24.* Ảnh khung hình liền trước và hiện thời có sai khác đáng kể.**
a) Ảnh liền trước, b) Ảnh hiện thời, c) Ảnh đen trắng dùng kỹ thuật trừ ảnh liền kề, với ngưỡng là 20.

a) Ảnh khung hình liền trước I_p

b) Ảnh khung hình hiện thời I_c

***Hình 6.25*. a) Ảnh liền trước, b) Ảnh liền sau, c) Ảnh xếp chồng b) lên a) điểm khác nhau sẽ có màu đỏ.**



c) Ảnh xếp chồng

Qua Hình 6.23, Hình 6.24 và Hình 6.25, chúng ta nhận thấy: với những khung hình sai khác nhau ít sẽ làm cho ảnh Iwb có màu đen (Ic, Ip có độ giống nhau lớn), còn với những khung hình có độ khác nhau đáng kể, thì ngoài những đối tượng chuyển động, còn có nhiễu (do sự tương phản, thời tiết...). Vì vậy, với kỹ thuật trừ khung hình liền kề có thể làm cho quá trình bám đối tượng chuyển động bị mất đi (vì có thể có 2 khung hình có độ khác nhau không đáng kể). Thực nghiệm cho thấy: Nếu chúng ta giảm giá trị ngưỡng thì sẽ tăng được độ phân biệt giữa các khung hình gần giống nhau, nhưng cũng sẽ làm tăng vùng khác nhau đối với khung hình có độ khác nhau nhiều, do đó sẽ làm cho quá trình xử lý phức tạp hơn. Như vậy, với kỹ thuật trừ khung hình liền kề chúng ta có được nhận xét sau:

- Chưa tối ưu được quá trình xử lý, vì phải xử lý tất cả khung hình có trong đoạn video mặc dù chúng không có sự thay đổi (các đối tượng không chuyển động).
- Thu được những thay đổi nhỏ đối với những đối tượng có chuyển động chậm và có thể là không phân biệt được vì thay đổi ít, vì vậy mà khó có thể lấy được toàn bộ đối tượng chuyển động.
- Chứa nhiều nhiễu, do thời tiết, độ tương phản, độ bóng làm thay đổi giá trị màu ở những vùng không chuyển động.

Vì vậy, chúng ta sẽ không thực hiện trừ khung hình liền kề để phát hiện đối tượng. Dưới đây chúng ta sẽ xem xét một hướng tiếp cận kết hợp: kỹ thuật trừ khung hình, đo độ thay đổi, xét vị trí, và kỹ thuật dò biên ảnh đa cấp xám để nâng cao hiệu quả phát hiện đối tượng chuyển động.

6.7.2. Phát hiện đối tượng chuyển động theo hướng tiếp cận kết hợp

Đây cũng là một hướng tiếp cận dựa trên kỹ thuật trừ ảnh, độ đo tương tự, xét vị trí tương đối giữa các vùng thay đổi giá trị màu ở các khung hình, phát hiện biên ảnh đa cấp xám. Hướng tiếp cận này được trình bày như sau:

Đầu tiên, phát hiện vùng thay đổi về giá trị màu giữa các khung hình theo kỹ thuật trừ ảnh điểm. Nếu 2 điểm được so sánh mà có giá trị khác nhau nhỏ hơn một ngưỡng cho trước thì chúng được coi là giống nhau, ngược lại chúng được coi là khác nhau, và được đánh dấu lại vào ảnh Iwb. Quá trình khử nhiễu bằng các cửa sổ 5x5, 3x3 trên ảnh đen trắng Iwb và xác định độ dịch chuyển sẽ khử nhiễu do thời tiết, độ từng phận. Tiếp theo, ảnh Igc được phát hiện biên cho kết quả là ảnh Iec (ảnh biên). Kết hợp Iec với Iwb, chúng ta sẽ loại bỏ được tất cả những điểm biên nằm ngoài vùng chuyển động (có màu trắng), lấy được biên của đối tượng chuyển động. Những điểm biên của Iec và lân cận vùng trắng sẽ được đánh dấu lại. Cuối cùng chúng ta xử lý điểm biên được đánh dấu để xác định được đối tượng chuyển động.

Dưới đây là bước thực hiện của thuật toán phát hiện đối tượng chuyển động theo hướng tiếp cận kết hợp.

Bước 1: Trừ ảnh và đánh dấu Iwb

Bước 2: Lọc nhiễu trên ảnh Iwb, phát hiện độ dịch chuyển

Bước 3: Phát hiện biên Igc

Bước 4: Kết hợp Igc với Iwb

Tiếp theo, chúng ta sẽ xem xét cụ thể hơn các bước thực hiện của hướng tiếp cận kết hợp dựa trên kỹ thuật trừ điểm ảnh và biên.

6.7.2.1. Trừ ảnh và đánh dấu Iwb

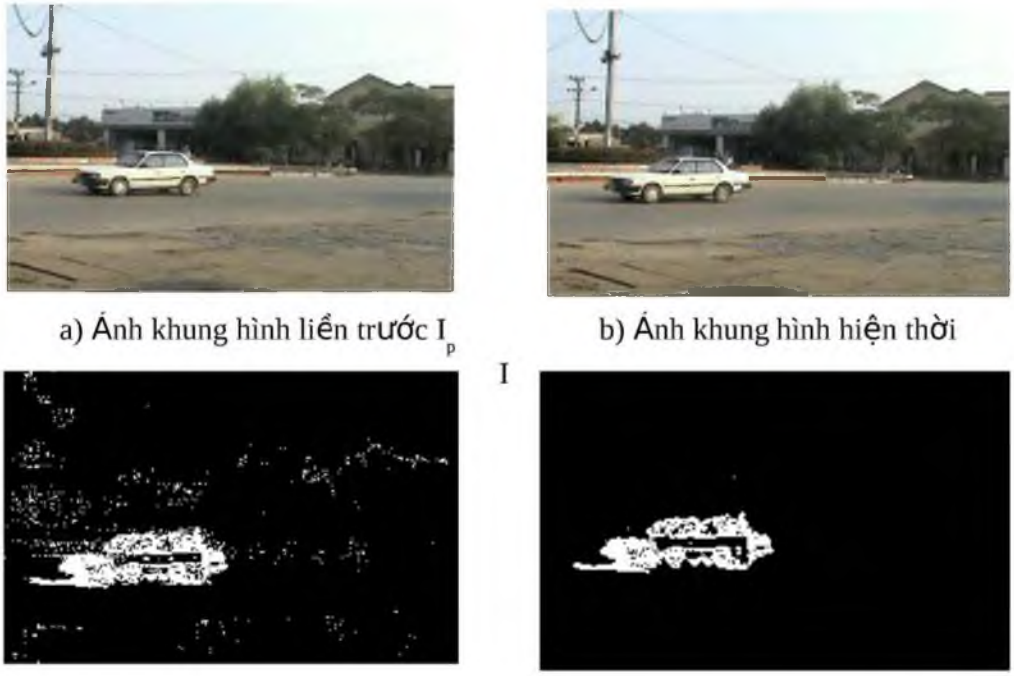
Đây là bước phát hiện vùng khác biệt dựa trên kỹ thuật trừ điểm ảnh, giá trị ngưỡng Threshold = 20. Tiến hành trừ từng điểm ảnh của Igc với Igp.

Khác với kỹ thuật trừ ảnh liên kề, trong quá trình trừ điểm ảnh, chương trình sẽ đếm số điểm Count giống nhau trên toàn vùng ảnh. Nếu giá trị tuyệt đối của hiệu số Count với Isize (kích thước ảnh) lớn hơn một giá trị ngưỡng thì được coi là khác nhau, ngược lại 2 khung hình được coi là giống nhau. Do đó, quá trình xử lý chỉ xem xét với những khung hình khác nhau đáng kể.

6.7.2.2. Lọc nhiễu và phát hiện độ dịch chuyển

Đây là bước để hạn chế được các vùng đốm dựa theo cửa sổ lọc nhiễu được chọn là 5x5, 3x3 và vị trí thay đổi của vùng ảnh. Một tập Iwb được xử lý để lấy được vị trí của vùng sáng. Nếu vị trí của vùng sáng thay đổi nhỏ hơn một giá trị ngưỡng thì được coi là nhiễu, và lập tức được dập tắt. Kết quả chúng ta có được ảnh đen trắng Iwb được lọc. Vùng sáng có độ lớn 1 điểm ảnh sẽ được dập tắt bởi cửa sổ 3x3.

Hình 6.26 dưới đây minh họa kết quả của ảnh Iwb chưa lọc nhiễu và đã được lọc nhiễu.



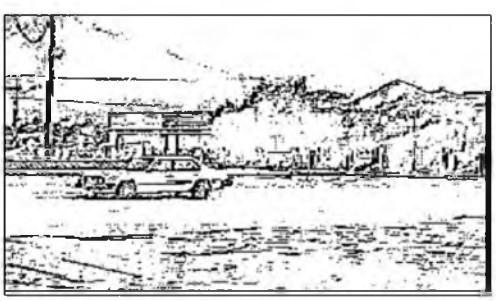
a) Ảnh khung hình liền trước I_p b) Ảnh khung hình hiện thời I

c) Ảnh Iwb chưa lọc nhiễu d) Ảnh Iwb đã lọc nhiễu

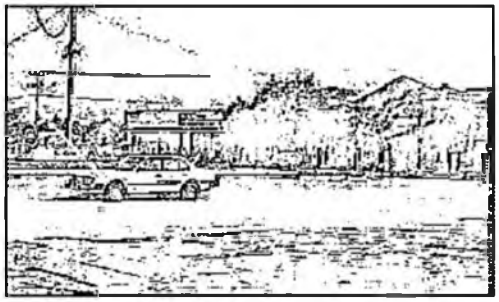
Hình 6.26. a), b) là 2 khung hình có độ sai khác thỏa mãn ngưỡng, c) Ảnh Iwb chưa lọc nhiễu, d) là ảnh Iwb sau khi lọc nhiễu

6.7.2.3. Phát hiện biên ảnh đa cấp xám Igc

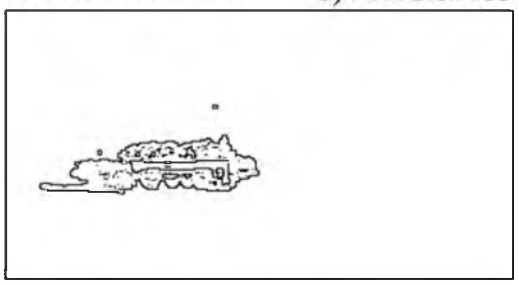
Trong quá trình này, ảnh xám hiện thời Igc được phát hiện biên dựa theo kỹ thuật phát hiện biên đã đề xuất được trình bày trong chương 2. Từ trên có thể xác định được ảnh biên của đối tượng chuyển động.



a) Ảnh biên Iep của ảnh xám liền



b) Ảnh biên Iec của ảnh xám hiện thời



c) Ảnh biên của đối tượng chuyển động

Hình 6.27. **a), b) là 2 ảnh biên có độ sai khác thỏa mãn ngưỡng, c) Ảnh biên của đối tượng chuyển động sau khi lọc nhiễu**

6.7.2.4. Kết hợp ảnh Igc với Iwb

Ảnh sau khi đã được phát hiện biên Iec được xếp chồng với ảnh đen trắng Iwb. Vùng đen trắng kết hợp với điểm biên sẽ cho ta được tập hợp các điểm bao quanh vùng chuyển động. Những điểm biên cúa Iec gần với vùng trắng của Iwb được đánh dấu lại trên Igc.

Hàm WB_Edge() trả về giá trị đúng nếu điểm p là biên, ngược lại trả về giá trị sai. Hàm GetPointEdge() sẽ thực hiện lấy điểm biên trên Iec sai lệch với điểm Iwb(p) là 2 đơn vị. Kết quả điểm biên trả về sẽ là điểm gần nhất. Nếu không có kết quả, trả về điểm trắng (giá trị trả về là 255) thì điểm Iwb(p) được coi là biên của ảnh Igc. Điểm này sẽ được lưu lại. Những điểm được lưu lại sẽ được xử lý để lấy được hình bao quanh đối tượng chuyển động đó. Và quá trình phát hiện được vùng biên của đối tượng chuyển động kết thúc.

Hệ thống phát hiện khung chuyển động của đối tượng liên tục đọc hai ảnh rồi lấy khung và in ra.

Dưới đây là một số hình ảnh kết quả đạt theo hướng tiếp cận kết hợp. Đoạn băng video có dạng avi, được quay tại ngã ba Bắc Nam của thành phố Thái Nguyên. Đoạn băng có độ lớn 14,01 MB. Kết quả cho thấy thuật toán phát hiện khá tốt các đối tượng chuyển động với các tốc độ chuyển động khác nhau: Ôtô, xe máy, xe đạp. Hình 6.28 là một số hình ảnh thu được và đã được phát hiện các đối tượng chuyển động.

Hình 6.28. **Ảnh kết quả phát hiện đối tượng chuyển động**

Phụ lục 1:

MỘT SỐ ĐỊNH DẠNG TRONG XỬ LÝ ẢNH

Hiện nay trên thế giới có trên 50 khuôn dạng ảnh thông dụng. Sau đây là một số định dạng ảnh hay dùng trong quá trình xử lý ảnh hiện nay.

1. Định dạng ảnh IMG

Ảnh IMG là ảnh đen trắng, phần đầu của ảnh IMG có 16 byte chứa các thông tin:

- 6 byte đầu: dùng để đánh dấu định dạng ảnh. Giá trị của 6 byte này viết dưới dạng Hexa: 0x0001 0x0008 0x0001
- 2 byte tiếp theo: chứa độ dài mẫu tin. Đó là độ dài của dãy các byte kề liền nhau mà dóy này sẽ được lặp lại một số lần nào đó. Số lần lặp này sẽ được lưu trong byte đếm. Nhiều dãy giống nhau được lưu trong một byte.
- 4 byte tiếp: mô tả kích cỡ pixel.
- 2 byte tiếp: số pixel trên một dòng ảnh.
- 2 byte cuối: số dòng ảnh trong ảnh.

Ảnh IMG được nén theo từng dòng, mỗi dòng bao gồm các gói (pack). Các dòng giống nhau cũng được nén thành một gói. Có 4 loại gói sau:

- Loại 1: Gói các dòng giống nhau.

Quy cách gói tin này như sau: 0x00 0x00 0xFF Count. Ba byte đầu tiên cho biết số các dãy giống nhau, byte cuối cho biết số các dòng giống nhau.

- Loại 2: Gói các dãy giống nhau.

Quy cách gói tin này như sau: 0x00 Count. Byte thứ hai cho biết số các dãy giống nhau được nén trong gói. Độ dài của dãy ghi ở đầu tệp.

- Loại 3: Dãy các Pixel không giống nhau, không lặp lại và không nén được.

Quy cách gói tin này như sau: 0x80 Count. Byte thứ hai cho biết độ dài dãy các pixel không giống nhau không nén được.

- Loại 4: Dãy các Pixel giống nhau.

Tuỳ theo các bít cao của byte đầu tiên được bật hay tắt. Nếu bít cao được bật (giá trị 1) thỡ đây là gói nén các byte chỉ gồm bít 0, số các byte được nén được tính bởi 7 bít thấp còn lại. Nếu bớt cao tắt (giỏ trị 0) thì đây là gói nén các byte gồm toán bít 1. Số các byte được nén được tính bởi 7 bít còn lại.

Các gói tin của file IMG rất đa dạng do ảnh IMG là ảnh đen trắng, do vậy chỉ cần 1 bít cho 1 pixel thay vì 4 hoặc 8 như đã nói ở trên. Toàn bộ ảnh chỉ có những điểm sáng và tối tương ứng với giá trị 1 hoặc 0. Tỷ lệ nén của kiểu định dạng này là khá cao.

2. Định dạng ảnh PCX

Định dạng ảnh PCX là một trong những định dạng ảnh cổ điển. Nó sử dụng phương pháp mã hoá loạt dài RLE (Run – Length – Encoded) để nén dữ liệu ảnh. Quá trình nén và giải nén được thực hiện trên từng dạng ảnh. Thực tế, phương pháp giải nén PCX kém hiệu quả hơn so với kiểu IMG. Tệp PCX gồm 3 phần: đầu tệp (header), dữ liệu ảnh (Image data) và bảng màu mở rộng.

Header của tệp PCX có kích thước cố định gồm 128 byte và được phân bố như sau:

- 1 byte: chỉ ra kiểu định dạng.Nếu là PCX/PCC thì nó luôn có giá trị là 0Ah.
- 1 byte: chỉ ra version sử dụng để nén ảnh, có thể có các giá trị sau:
  - 0: version 2.5.
  - 2: version 2.8 với bảng màu.
  - 3: version 2.8 hay 3.0 không có bảng màu.
  - 5: version 3.0 cổ bảng màu.
- 1 byte: chỉ ra phương pháp mã hoá. Nếu là 0 thì mã hoá theo phương pháp BYTE PACKED, ngược lại là phương pháp RLE.
- 1 byte: Số bít cho một điểm ảnh plane.
- 1 word: toạ dộ góc trái của ảnh. Với kiểu PCX nó có giá trị là (0,0), cũn PCC thì khác (0,0).
- 1 word: toạ độ góc phải dưới.
- 1 word: kích thước bề rộng và bề cao của ảnh.
- 1 word: số điểm ảnh.
- 1 word: độ phân giải màn hình.
- 1 word.

- 48 byte: chia nó thành 16 nhóm, mỗi nhóm 3 byte. Mỗi nhóm này chứa thông tin về một thanh ghi màu. Như vậy ta có 16 thanh ghi màu.
- 1 byte: không dùng đến và luôn đặt là 0.
- 1 byte: số bớt plane mà ảnh sử dụng. Với ảnh 16 màu, giá trị này là 4, với ảnh 256 mầu (1pixel/8bits) thì số bít plane lại là 1.
- 1 byte: số bytes cho một dòng quét ảnh.
- 1 word: kiểu bảng màu.
- 58 byte: không dùng.

Định dạng ảnh PCX thường được dùng để lưu trữ ảnh và thao tác đơn giản, cho phép nén và giải nén nhanh. Tuy nhiên, vì cấu trúc của nó cố định, nên trong một số trường hợp làm tăng kích thước lưu trữ. Cũng vì nhược điểm này mà một số ứng dụng sử dụng một kiểu định dạng khác mềm dẻo hơn: định dạng TIFF (Targed Image File Format) sẽ mô tả dưới đây.

3. Định dạng ảnh TIFF

Kiểu định dạng TIFF được thiết kế để làm nhẹ bớt các vấn đề liên quan đến việc mở rộng tệp ảnh cố định. Về cấu trúc, nó cũng gồm 3 phần chính:

- Phần Header(IFH): cú trong tất cả cỏc tệp TIFF và gồm 8 byte:
  - 1 word: chỉ ra kiểu tạo tệp trên máy tính PC hay máy Macintosh. Hai loại này khác nhau rất lớn ở thứ tự các byte lưu trữ trong các số dài 2 hay 4 byte. Nếu trường này có giá trị là 4D4Dh thì đó là ảnh cho máy Macintosh, nếu là 4949h là của máy PC.
  - 1 word: version. từ này luôn có giá trị là 42. đây là đặc trưng của file TIFF và không thay đổi.
  - 2 word: giá trị Offset theo byte tính từ đầu tới cấu trúc IFD là cấu trúc thứ hai của file. Thứ tự các byte này phụ thuộc vào dấu hiệu trường đầu tiên.
- Phần thứ 2(IFD): Không ở ngay sau cấu trúc IFH mà vị trí được xác định bởi trường Offset trong đầu tệp. Có thể có một hay nhiều IFD cùng tồn tại trong một file.

  Một IFD bao gồm:
  - 2 byte: chứa các DE (Directory Entry).

- 12 byte là các DE xếp liên tiếp, mỗi DE chiếm 12 byte.
- 4 byte: chứa Offset trỏ tới IFD tiếp theo. Nếu đây là IFD cuối cùng thì trường này có giá trị 0.

- Phần thứ 3: các DE: các DE có độ dài cố định gồm 12 byte và chia làm 4 phần:
  - 2 byte: chỉ ra dấu hiệu mà tệp ảnh đó được xây dựng.
  - 2 byte: kiểu dữ liệu của tham số ảnh. Có 5 kiểu tham số cơ bản:

    1: BYTE (1 byte)

    2: ASCII (1 byte)

    3: SHORT (2 byte).

    4: LONG (4 byte)

    5: RATIONAL (8 byte)
  - 4 byte: trường độ dài chưa số lượng chỉ mục của kiểu dữ liệu đó chỉ ra. Nó không phải là tổng số byte cần thiết để lưu trữ. Để có số liệu này ta cần nhân số chỉ mục với kiểu dữ liệu đã dùng.
  - 4 byte: đó là Offset tới điểm bắt đầu dữ liệu liên quan tới dấu hiệu, tức là liên quan với DE không phải lưu trữ vật lý cùng với nó nằm ở một vị trí nào đó trong file.

Dữ liệu chứa trong tệp thường được tổ chức thành các nhóm dòng (cột) quét của dữ liệu ảnh. Cách tổ chức này làm giảm bộ nhớ cần thiết cho việc đọc tệp. Việc giải nén được thực hiện theo 4 kiểu khác nhau được lưu trữ trong byte dấu hiệu nén.

4. Định dạng file ảnh BITMAP

Mỗi file BITMAP gồm đầu file chứa các thông tin chung về file, đầu thông tin chứa các thông tin về ảnh, một bảng màu và một mảng dữ liệu ảnh. Khuôn dạng được cho như sau:

```
BITMAPFILEHEADER  bmfh;
BITMAPINFOHEADER bmih;
RGBQUAD                aColors[];
BYTE                      aBitmapBits[];
```

Trong đó, các cấu trúc được định nghĩa như sau:

```
typedef struct tagBITMAPFILEHEADER {    /* bmfh */
  UINT       bfType;
  DWORD   bfSize;
```

```
    UINT        bfReserved1;
    UINT        bfReserved2;
    DWORD       bfOffBits;
} BITMAPFILEHEADER;
typedef struct tagBITMAPINFOHEADER {    /* bmih */
    DWORD       biSize;
    LONG        biWidth;
    LONG        biHeight;
    WORD        biPlanes;
    WORD        biBitCount;
    DWORD       biCompression;
    DWORD       biSizeImage;
    LONG        biXPelsPerMeter;
    LONG        biYPelsPerMeter;
    DWORD       biClrUsed;
    DWORD       biClrImportant;
} BITMAPINFOHEADER, *LPBITMAPINFOHEADER;
```

với

biSize	kích thước của BITMAPINFOHEADER
biWidth	Chiều rộng của ảnh, tính bằng số điểm ảnh
biHeight	Chiều cao của ảnh, tính bằng số điểm ảnh
biPlanes	Số plane của thiết bị, phải bằng 1
biBitCount	Số bit cho một điểm ảnh
biCompression	Kiểu nén
biSizeImage	Kích thước của ảnh tính bằng byte
biXPelsPerMeter	độ phân giải ngang của thiết bị, tính bằng điểm ảnh trên met
biYPelsPerMeter	độ phân giải dọc của thiết bị, tính bằng điểm ảnh trên met
biClrUsed	Số lượng các màu thực sự được sử dụng
biClrImportant	Số lượng các màu cần thiết cho việc hiển thị, bằng 0 nếu tất cả các màu đều cần để hiển thị

Nếu bmih.biBitCount > 8 thì mảng màu rgbq[] trống, ngược lại thì mảng màu có 2<< bmih.biBitCount phần tử.

```
typedef struct tagRGBQUAD {     /* rgbq */
```

```
   BYTE        rgbBlue;
   BYTE        rgbGreen;
   BYTE        rgbRed;
   BYTE        rgbReserved;
} RGBQUAD;
```

Ta cũng có:

```
typedef struct tagBITMAPINFO {
   BITMAPINFOHEADER    bmiHeader;
   RGBQUAD             bmiColors[1];
} BITMAPINFO, *PBITMAPINFO;
```

Phụ lục 2:

CÁC BƯỚC THAO TÁC VỚI FILE AVI

AVI là chuẩn video thường được tích hợp trong các thư viện của các môi trường lập trình. Để xử lý video, cần có các thao tác cơ bản để chuyển về xử lý ảnh các khung hình (các frames).

1. Bước 1: Mở và đóng thư viện

Trước mọi thao tác với file AVI, chúng ta phải mở thư viện:

AVIFileInit( )

Hàm này không cần tham số, có nhiệm vụ khởi động thư viện cung cấp các hàm thao tác với file AVI. (Đó là thư viện vfw32.lib, được khai báo trong file vfw.h).

Sau tất cả các thao tác bạn phải nhớ đóng thư viện đã mở lúc đầu, chỉ bằng lệnh:

AVIFileExit( )

Nếu thiếu bất cứ hàm nào, dù là mở hay đóng thư viện thì trình biên dịch đều sẽ thông báo lỗi.

2. Bước 2: Mở và đóng file AVI để thao tác:

Sau khi mở thư viện, bạn phải mở file AVI bạn định thao tác:

AVIFileOpen(PAVIFILE ppfile, LPCSTR fname, UINT mode,*

CLSID pclsidHandler)

Thực chất, hàm này tạo ra một vùng đệm chứa con trỏ trỏ đến file có tên là fname cần mở. Và ppfile là con trỏ trỏ đến vùng bộ đệm đó. Tham số mode quy định kiểu mở file; chẳng hạn OF_CREATE để tạo mới, OF_READ để đọc, OF_WRITE để ghi …. Tham số cuối dùng là NULL.

Trước khi đóng thư viện, bạn phải đóng file AVI đã mở, bằng cách dùng hàm:

AVIFileRelease(PAVIFILE pfile)

Trong đó, pfile là con trỏ trỏ đến file cần đóng.

3. Bước 3:

Mở dòng dữ liệu hình ảnh hay âm thanh trong file AVI đã mở ra để thao tác:

*AVIFileGetStream(PAVIFILE pfile, PAVISTREAM * ppavi, DWORD fccType, LONG lParam)*

Trong đó, pfile là con trỏ đến file đã mở; ppavi trỏ đến dòng dữ liệu kết quả; fccType là loại dòng dữ liệu chọn để mở, là streamtypeAUDIO nếu là tiếng và streamtypeVIDEO nếu là hình,... lParam đếm số loại dòng được mở, là 0 nếu chỉ thao tác với một loại dòng dữ liệu.

Sau các thao tác với dòng dữ liệu này, bạn nhớ phải đóng nó lại:

AVIStreamRelease(PAVITREAM pavi).

4. Bước 4: Trường hợp thao tác với dữ liệu hình của phim

Chuẩn bị cho thao tác với khung hình (frames):

AVIStreamGetFrameOpen(PAVISTREAM pavi,

LPBITMAPINFOHEADER lpbiWanted)

Trong đó pavi trỏ đến dòng dữ liệu đã mở, lpbiWanted là con trỏ trỏ đến cấu trúc mong muốn của hình ảnh, ta dùng NULL để sử dụng cấu trúc mặc định.

Hàm này trả về đối tượng có kiểu PGETFRAME để dùng cho bước 5.

Sau khi thao tác với các frame rồi, phải gọi hàm :

AVIStreamGetFrameClose(PGETFRAME pget)

5. Bước 5: Thao tác với frame

Dùng hàm

AVIStreamGetFrame(PGETFRAME pget, LONG lpos)

Hàm này trả về con trỏ trỏ đến dữ liệu của frame thứ lpos. Dữ liệu đó có kiểu là DIB đã định khối.

Thực hiện các thao tác mong muốn.

Phụ lục 3:

MỘT SỐ MODUL CHƯƠNG TRÌNH

1. Nhóm đọc, ghi và hiển thị ảnh

1.1. Nhóm đọc ảnh

```
HDIB WINAPI ReadPCXFile(HFILE hf)
{
   PCXHEADER pcx;
   if (!::ReadPCXHeader(hf, &pcx)) return NULL;

   // make a new bitmap header
   BITMAPINFOHEADER bmi;
   ::InitBitmapInfoHeader(&bmi, (DWORD)(pcx.window.xmax-
      pcx.window.xmin+1), (DWORD)(pcx.window.ymax-
      pcx.window.ymin+1), pcx.bitsperpixel);
   // Locate the memory
   HDIB hDIB = ::GlobalAlloc(GMEM_MOVEABLE,
      (DWORD)sizeof(BITMAPINFOHEADER) +
      (DWORD)::PaletteSize((LPSTR)&bmi) + bmi.biSizeImage);
   if (!hDIB) return NULL; // Fail
   LPBITMAPINFOHEADER pDIB =
      (LPBITMAPINFOHEADER)::GlobalLock(hDIB);
   *pDIB = bmi;
   DWORD wBytes = (WORD)WIDTHBYTES(pDIB→biWidth*pDIB→
      biBitCount);

   /*---------------- DU LIEU CHUNG ------------------*/
   W = wBytes;
   nH = (int)pDIB→biHeight;
   nW = pDIB→biWidth;
   lpRGB = (LPRGBQUAD)(pDIB + 1);
   pImage = ((HBYTE)::FindDIBBits((LPSTR)pDIB));
```

```
/*-------------------------------------------------*/
HBYTE pLine = ((HBYTE)::FindDIBBits((LPSTR)pDIB)) +
   wBytes*(pDIB→biHeight-1);
WORD sizeBuff = 10240,      // 10 KB
index = 10, cr = 0, tmp = 0;
HGLOBAL hBuffers = ::GlobalAlloc(GMEM_MOVEABLE,
   sizeBuff+64);
HBYTE   pBuffers = (HBYTE)::GlobalLock(hBuffers);

for(int i=0;i<256;++i)
{
   HistogramC[i] = 0;
   MauNen[i] = 0;
}
int L;
BYTE Color;
DWORD d=0;
for (i = 0; i < (int)pDIB→biHeight; i++)
{
   DWORD total = 0;
   while (total < pcx.bytesperline)
   {
      if (index >= cr) // Buffers
      {
         if ((tmp > 0)&&(index == cr)) pBuffers[0] = pBuffers[index];
         else tmp = 0;
            index = 0;
         #ifdef _WIN32
         cr = _lread(hf, (LPVOID)(pBuffers+tmp), sizeBuff);
         d+=cr;
         #else
```

```
            #endif// _WIN32
            if (!tmp) {tmp = 1; cr--;}
        }
        static BYTE b;
        if ((b = pBuffers[index++]) >= 0xC0) // Get first byte
        {
            b &= 0x3F;
            if (total < wBytes)
            #ifdef _WIN32
            {
                L = min((int)b, (int)(wBytes-total));
                memset((void*)(pLine+total),(Color=pBuffers[index++]),L);
                HistogramC[Color] += L;
            }
            #else
            #endif// _WIN32
            total += (WORD)b;
        }
        else if (total < wBytes)
        {
            pLine[total++] = b;
            ++HistogramC[b];
        }
        else total++;
    }
    pLine -= (LONG)wBytes;
}
LPRGBQUAD lpRGB = (LPRGBQUAD)(pDIB + 1);
if (pDIB→biBitCount == 1) // Create the Look Up Table
{
    lpRGB[0].rgbRed = lpRGB[0].rgbGreen = lpRGB[0].rgbBlue = 0; //
        Black
```

```
    lpRGB[1].rgbRed = lpRGB[1].rgbGreen = lpRGB[1].rgbBlue = 255; //
      White
    lpRGB[0].rgbReserved = lpRGB[1].rgbReserved = 0;
} else // 8 bit image, read LUT from file
{
    #ifdef _WIN32
    _llseek(hf, -768, FILE_END);
    _lread(hf, (LPVOID)pBuffers, 768); // Read
    #else
    #endif// _WIN32
    CString s;
    for (i = 0; i < 256; i++) // Convert to RGBQUAD
    {
        RGBRoad[i].rgbRed  =  lpRGB[i].rgbRed = pBuffers[i*3];
        RGBRoad[i].rgbGreen  =  lpRGB[i].rgbGreen = pBuffers[i*3+1];
        RGBRoad[i].rgbBlue  =  lpRGB[i].rgbBlue = pBuffers[i*3+2];
        RGBRoad[i].rgbReserved  =  lpRGB[i].rgbReserved = 0;
    }
}
double TotalPixel = 1.0*nH*nW;
double Color_i;
int Count=0;
CString s;
for(i=1;i<255;++i)
{
    if(HistogramC[i])
        ++Count;
    Color_i = 1.0*HistogramC[i]/TotalPixel;
    if(Color_i > NGUONG_NEN)
        MauNen[i] = 1;
}
if(Count>=1)
```

```
    {
        int LENGHT = 150;
        DWORD j=LENGHT;
        DWORD i;
        while(j<=nH)
        {
            for(i=1;i<=nW;++i)
                if(i%LENGHT == 0)
                    PhatHienNen(i-LENGHT,j-LENGHT,i,j);
            j += LENGHT;
        }
    }
    MauNen[255]=1; // COI MAU TRANG LA 1 MAU NEN.
    MauNen[0] = 0; // COI MAU DEN NHU LA 1 MAU CUA DUONG.
    ::GlobalUnlock(hDIB);
    ::GlobalUnlock(hBuffers);
    ::GlobalFree(hBuffers);
    sNgaba  = 0;
    for(i=0;i<S;++i)
    for(int j=0;j<S;++j)
        dd[i][j] = 0;

    return hDIB;
}

double m_TotalPoints;
VOID WINAPI PhatHienNen(int x1,int y1,int x2,int y2)
{
    ZeroMemory(HistogramC,256*4);
    int i,j;
    BYTE c;
    for( j=y1;j<=y2;++j)
```

```
    for( i=x1;i<=x2;++i)
    {
       c = getPoint(i,j);
       ++HistogramC[c];
     }
    m_TotalPoints = (x2-x1)*(y2-y1);
    for(i=1;i<256;++i)
       if(!MauNen[i] && 1.0*HistogramC[i]/m_TotalPoints >
          NGUONG_NEN)
    MauNen[i] = TRUE;
    PhatHienNen(x1,y1,(x1+x2)/2,(y1+y2)/2);
    PhatHienNen((x1+x2)/2,y1,x2,(y1+y2)/2);
    PhatHienNen(x1,(y1+y2)/2,(x1+x2)/2,y2);
    PhatHienNen((x1+x2)/2,(y1+y2)/2,x2,y2);
}

/*---------------------------------------------------------------------*/
HDIB WINAPI ReadPCXFile(LPCSTR fName)
{
    #ifdef _WIN32
       HFILE hf = _lopen(fName, OF_READ);
    #else
    #endif// _WIN32
    if (!hf) return NULL;
    ::SetCursor(::LoadCursor(NULL, IDC_WAIT));
    HDIB hDIB = ::ReadPCXFile(hf);
    _lclose(hf);
    ::SetCursor(::LoadCursor(NULL, IDC_ARROW));
    return hDIB;
}

/*---------------------------------------------------------------------*/
```

```
BOOL WINAPI ReadPCXHeader(HFILE hf, LPPCXHEADER pcxh)
{
   // Read the file's header
   #ifdef _WIN32
      if (_lread(hf, (LPVOID)pcxh, 128) != 128) return FALSE;
   #else
   #endif// _WIN32
   if ( pcxh→manufacture != 0x0A ) // Check manufacture of the PCX file
      return FALSE;
   // Only work with B/W and 8 bit image
   if ((pcxh→bitsperpixel*pcxh→nplanes != 1) &&
      (pcxh→bitsperpixel*pcxh→nplanes != 8))
      return FALSE;
   if (pcxh→encoding != 1) // Unknow how to decode
      return FALSE;
   return TRUE;
}
/*-------------------------------------------------------------------*/
VOID WINAPI CreatePCXHeader(LPPCXHEADER pcxh,
   LPBITMAPINFOHEADER lpDIB)
{
   pcxh→manufacture= 0x0A;// Signature
   pcxh→version= (lpDIB→biBitCount == 1) ? 2 : 5;// PCX version
   pcxh→encoding= 0x01;// Run length
   pcxh→bitsperpixel = (char)lpDIB→biBitCount;
   pcxh→window.xmin  = 0;
   pcxh→window.ymin= 0;
   pcxh→window.xmax= lpDIB→biWidth -1;
   pcxh→window.ymax= lpDIB→biHeight-1;
   pcxh→hres= (WORD)lpDIB→biXPelsPerMeter;
   pcxh→vres= (WORD)lpDIB→biYPelsPerMeter;
```

```
    pcxh→reserved= 0x00;
    pcxh→nplanes= 1;
    pcxh→bytesperline=
      (WORD)WIDTHBYTES(lpDIB→biBitCount*lpDIB→biWidth);
    pcxh→palette_info= 1;
    for (int i = 0; i < 58; i++) pcxh→filler[i] = 0;
      if (lpDIB→biBitCount == 1)
      {// create LUT
        pcxh→colormap[0] = pcxh→colormap[1] = pcxh→colormap[2] = 0;
        pcxh→colormap[3] = pcxh→colormap[4] = pcxh→colormap[5] = 0;
      }
}
/*---------------------------------------------------------------------*/
DWORD WINAPI CompressLine(HBYTE pDes, HBYTE pSource,
    DWORD Bytes)
{
    DWORD j = 0, iw = 0;
    while ( j < Bytes )
    {
      BYTE Count = 1;
      BYTE item  = pSource[j];
      while ((j < Bytes-1) && (item == pSource[j+1]) && (Count < 0xFF-
        0xC0-1))
      {
        j++;
        Count++;
      }
      if ((Count > 1)||(item >= 0xC0))
      {
        pDes[iw++] = Count + 0xC0;
        pDes[iw++] = item;
      }
```

```
        else pDes[iw++] = item;
            j++;
        }
    return iw;
}
```

1.2. Nhóm ghi ảnh

```
BOOL WINAPI SavePCX(HDIB hDIB, HFILE hf)
{
    if (!hDIB || !hf) return FALSE;
    LPBITMAPINFOHEADER lpDIB =
        (LPBITMAPINFOHEADER)::GlobalLock(hDIB);
    PCXHEADER pcxh;
    ::CreatePCXHeader(&pcxh, lpDIB);
    #ifdef _WIN32
        _lwrite(hf, (LPCSTR)&pcxh, 128);
    #else
    #endif// _WIN32
    DWORD wBytes =
        (WORD)WIDTHBYTES(lpDIB→biWidth*lpDIB→biBitCount);
    // Get DIB line 0
    HBYTE pLine = ((HBYTE)::FindDIBBits((LPSTR)lpDIB)) +
        wBytes*(lpDIB→biHeight-1);
    ///// Buffer
    WORDsizeBuff = 10240; // 10 KB
    HGLOBAL hBuffers = ::GlobalAlloc(GMEM_MOVEABLE,
        sizeBuff+64);
    HBYTEpBuffers = (HBYTE)::GlobalLock(hBuffers);
    ///
    for (DWORD i = 0; i < (DWORD)lpDIB→biHeight; i++)
    {
        // Write line
        #ifdef _WIN32
```

```
            _lwrite(hf, (LPCSTR)pBuffers, (UINT)::CompressLine(pBuffers,
               pLine, wBytes));
         #else
         #endif// _WIN32
         pLine -= (LONG)wBytes;// Next line
      }
      if (lpDIB→biBitCount == 8)
      {
         LPRGBQUAD lpRGB = (LPRGBQUAD)(lpDIB + 1);
         for (i = 0; i < 256; i++) // Convert to RGBQUAD
         {
            pBuffers[i*3]  = lpRGB[i].rgbRed;
            pBuffers[i*3+1]= lpRGB[i].rgbGreen;
            pBuffers[i*3+2]= lpRGB[i].rgbBlue;
         }
         #ifdef _WIN32
            BYTE b = 0x0C;
            _lwrite(hf, (LPCSTR)&b, 1); // Write signature of palette
            _lwrite(hf, (LPCSTR)pBuffers, 768); // Write palette
         #else
         #endif// _WIN32
      }
      ::GlobalUnlock(hDIB);
      ::GlobalUnlock(hBuffers);
      ::GlobalFree(hBuffers);
      return TRUE;
   }
   /*------------------------------------------------------------------*/
   BOOL WINAPI SavePCX(HDIB hDIB, LPCSTR fName)
   {
      HFILE hf = _lcreat(fName, 0); // Check the file could be created
      if (!hf) return FALSE;
```

```
    ::SetCursor(::LoadCursor(NULL, IDC_WAIT));
  #ifdef _WIN32
    hf = _lopen(fName, OF_READWRITE);
  #else
  #endif// _WIN32
  BOOL rs = ::SavePCX(hDIB, hf);
  _lclose(hf);
  ::SetCursor(::LoadCursor(NULL, IDC_ARROW));
  return rs;
}
```

1.3. Nhóm hiển thị ảnh

```
BOOL WINAPI PaintDIB(HDC hDC, LPRECT lpDCRect, HDIB hDIB,
  LPRECT lpDIBRect, CPalette* pPal)
{
  /* Check for valid DIB handle */
  if (!hDIB) return FALSE;
  /* Lock down the DIB, and get a pointer to the beginning of the bit
    buffer
  */
  LPSTR lpDIBHdr  = (LPSTR) ::GlobalLock((HGLOBAL)hDIB);
  LPSTR lpDIBBits = ::FindDIBBits(lpDIBHdr);
  // Get the DIB's palette, then select it into DC
  // Select as background since we have
  // already realized in forground if needed
  HPALETTE hOldPal = ::SelectPalette(hDC,
    (HPALETTE)pPal→m_hObject, TRUE);
  ::RealizePalette(hDC);
  /* Make sure to use the stretching mode best for color pictures */
  ::SetStretchBltMode(hDC, COLORONCOLOR);
  /* Determine whether to call StretchDIBits() or SetDIBitsToDevice() */
  BOOL bSuccess;
  if ((RECTWIDTH(lpDCRect)  == RECTWIDTH(lpDIBRect)) &&
```

```
   (RECTHEIGHT(lpDCRect) == RECTHEIGHT(lpDIBRect)))
   bSuccess = ::SetDIBitsToDevice(hDC,    // hDC
      lpDCRect→left,                      // DestX
      lpDCRect→top,                       // DestY
      RECTWIDTH(lpDCRect),                // nDestWidth
      RECTHEIGHT(lpDCRect),               // nDestHeight
      lpDIBRect→left,                     // SrcX
      (int)DIBHeight(lpDIBHdr) - lpDIBRect→top -
      RECTHEIGHT(lpDIBRect),              // SrcY
      0,                                  // nStartScan
      (WORD)DIBHeight(lpDIBHdr),          // nNumScans
      lpDIBBits,                          // lpBits
      (LPBITMAPINFO)lpDIBHdr,             // lpBitsInfo
      DIB_RGB_COLORS);                    // wUsage
else
   bSuccess = ::StretchDIBits(hDC,        // hDC
      lpDCRect→left,                      // DestX
      lpDCRect→top,                       // DestY
      RECTWIDTH(lpDCRect),                // nDestWidth
      RECTHEIGHT(lpDCRect),               // nDestHeight
      lpDIBRect→left,                     // SrcX
      lpDIBRect→top,                      // SrcY
      RECTWIDTH(lpDIBRect),               // wSrcWidth
      RECTHEIGHT(lpDIBRect),              // wSrcHeight
      lpDIBBits,                          // lpBits
      (LPBITMAPINFO)lpDIBHdr,             // lpBitsInfo
      DIB_RGB_COLORS,                     // wUsage
      SRCCOPY);                           // dwROP
///
::GlobalUnlock((HGLOBAL) hDIB);
/* Reselect old palette */
```

```
    if (hOldPal) ::SelectPalette(hDC, hOldPal, TRUE);
      return bSuccess;
    }
/*---------------------------------------------------------------------*/
BOOL WINAPI CreateDIBPalette(HDIB hDIB, CPalette* pPal)
{
    /* if handle to DIB is invalid, return FALSE */
    if (!hDIB) return FALSE;
    LPSTR lpbi = (LPSTR)::GlobalLock((HGLOBAL)hDIB);
    /* get pointer to BITMAPINFO */
    LPBITMAPINFO lpbmi = (LPBITMAPINFO)lpbi;
    /* get the number of colors in the DIB */
    WORD wNumColors = ::DIBNumColors(lpbi);
    if (!wNumColors)
    {
      ::GlobalUnlock((HGLOBAL)hDIB);
      return FALSE;
    }
    /* allocate memory block for logical palette */
    HANDLE hLogPal = ::GlobalAlloc(GHND, sizeof(LOGPALETTE) +
      sizeof(PALETTEENTRY) * wNumColors);
    /* if not enough memory, clean up and return NULL */
    if (!hLogPal)
    {
      ::GlobalUnlock((HGLOBAL)hDIB);
      return FALSE;
    }
    LPLOGPALETTE lpPal =
      (LPLOGPALETTE)::GlobalLock((HGLOBAL)hLogPal);
    /* set version and number of palette entries */
    lpPal→palVersion = PALVERSION;
    lpPal→palNumEntries = (WORD)wNumColors;
```

```
    for (int i = 0; i < (int)wNumColors; i++)
    {
       lpPal→palPalEntry[i].peRed   = lpbmi→bmiColors[i].rgbRed;
       lpPal→palPalEntry[i].peGreen = lpbmi→bmiColors[i].rgbGreen;
       lpPal→palPalEntry[i].peBlue  = lpbmi→bmiColors[i].rgbBlue;
       lpPal→palPalEntry[i].peFlags = 0;
    }
    /* create the palette and get handle to it */
    BOOL bResult = pPal→CreatePalette(lpPal);
    ::GlobalUnlock((HGLOBAL) hLogPal);
    ::GlobalFree((HGLOBAL) hLogPal);
    ::GlobalUnlock((HGLOBAL) hDIB);
    return bResult;
    }
/*-------------------------------------------------------------------------*/
LPSTR WINAPI FindDIBBits(LPSTR lpbi)
{
    return (lpbi + *(LPDWORD)lpbi + ::PaletteSize(lpbi));
}
```

```
/*-------------------------------------------------------------------*/
DWORD WINAPI DIBWidth(LPSTR lpDIB)
{
   return ((LPBITMAPINFOHEADER)lpDIB) →biWidth;
}
/*-------------------------------------------------------------------*/
DWORD WINAPI DIBHeight(LPSTR lpDIB)
{
   return ((LPBITMAPINFOHEADER)lpDIB) →biHeight;
}
/*-------------------------------------------------------------------*/
WORD WINAPI PaletteSize(LPSTR lpbi)
{
   return (WORD)(::DIBNumColors(lpbi) * sizeof(RGBQUAD));
}
/*-------------------------------------------------------------------*/
WORD WINAPI DIBNumColors(LPSTR lpbi)
{
   if (((LPBITMAPINFOHEADER)lpbi) →biClrUsed)
      return (WORD)((LPBITMAPINFOHEADER)lpbi) →biClrUsed;
   /*  Calculate the number of colors in the color table based on the number of
      bits per pixel for the DIB.
   */
   WORD wBitCount = ((LPBITMAPINFOHEADER)lpbi) →biBitCount;
   if (wBitCount <= 8) return (2 << wBitCount);
   return 0;
}
/*-------------------------------------------------------------------*/
void WINAPI InitBitmapInfoHeader(LPBITMAPINFOHEADER lpbi,
DWORD dwWidth, DWORD dwHeight, int nBPP)
{
   #ifdef _WIN32
```

```
    memset((void*)lpbi, 0, sizeof (BITMAPINFOHEADER));
  #else
  #endif// _WIN32
  lpbi→biSize      = sizeof (BITMAPINFOHEADER);
  lpbi→biWidth     = dwWidth;
  lpbi→biHeight    = dwHeight;
  lpbi→biPlanes    = 1;
  if (nBPP <= 1) lpbi→biBitCount = 1;
  else if(nBPP <= 4)lpbi→biBitCount = 4;
  else if(nBPP <= 8)lpbi→biBitCount = 8;
  else lpbi→biBitCount = 24;
  lpbi→biSizeImage =
    ((DWORD)WIDTHBYTES(dwWidth*lpbi→biBitCount))*dwHeight;
  if(lpbi→biBitCount == 1)
    M("Anh 1 bit khong xet !");
  CString s;
  s.Format("W =  %ld H = %ld bit =
    %d",dwWidth,dwHeight,lpbi→biBitCount);
}
```

2. Nhóm phát hiện góc nghiêng văn bản

```
/*=============================================
 * Hàm: Next(HBYTE Image,DWORD x,DWORD y,int dir ,BYTE c0)
* Mục đích:Tìm điểm đen kế tiếp ngược chiều đồng hồ trong dò biên
*/
int WINAPI Next(HBYTE Image,DWORD x,DWORD y,int dir ,BYTE c0)
{
  int c,dem=8;
  DWORD newx,newy;
  while(dem>0)
  {
    newx=x+ROW[(dem+dir)%8];
```

```
        newy=y+COL[(dem+dir)%8];
        if(newx>=0 && newx<biHeight && newy>=0 && newy<biWidth)
        if(GetPoint(Image,newx,newy)==c0) return((dir+dem)%8);
        --dem;
    }
    return(-1);
}

/*=============================================
 * Hàm: Inverse(HBYTE Image,DWORD x,DWORD y,int dir ,BYTE c0)
* Mục đích:Tìm điểm đen kế tiếp ngược chiều đồng hồ trong dò biên
*/
int WINAPI Inverse(HBYTE Image, DWORD x, DWORD y, int dir,
    BYTE c0)
{
    DWORD newx,newy,dem=0;
    while(dem<8)
    {
        newx=x+ROW[(dem+dir)%8];
        newy=y+COL[(dem+dir)%8];
        if(newx>=0 && newx<biHeight && newy>=0 && newy<biWidth)
            if(GetPoint(Image,newx,newy)==c0) return((dem+dir-1)%8);
        ++dem;
    }
    return(8);
}

/*=============================================
 * Hàm: PreProcessing(HDIB hDIB,int numclr,int dilation)
* Mục đích: Xử lý sơ bộ
*/
void WINAPI PreProcessing(HDIB hDIB,int numclr,int dilation)
```

```
{
  LPBITMAPINFOHEADER lpDIB
    =(LPBITMAPINFOHEADER)::GlobalLock(hDIB);
  HBYTE Image=((HBYTE)::FindDIBBits((LPSTR)lpDIB));
  if(lpDIB→biBitCount<8)
{
    AfxMessageBox("Khong xu ly anh den trang ");
    return;
}
int c,id=0,Id[256];
WORD clr,min;
DWORD i,j,temp,maxhis=0,His[256];
biHeight=lpDIB→biHeight; biWidth =lpDIB→biWidth;
for(i=-1;++i<256;){ His[i]=0;Id[i]=i;}
for(i=-1;++i<360;)for(j=-1;++j<4500;)Hough[i][j]=0;
for(i=-1;++i<biHeight;) for(j=-1;++j<biWidth;)
{
    Mark[i][j]=0;
    c=GetPoint(Image,i,j);
    if(His[c]==0)++id;
      ++His[c];
}
if(id>numclr)
{
    for(i=-1;++i<255;) for(j=i;++j<256;)
    if(His[i]<His[j])
    {
       id=Id[i];Id[i]=Id[j];Id[j]=id;
       temp=His[i];His[i]=His[j];His[j]=temp;
    }
    for(i=-1;++i<256;)His[i]=0;
    for(i=-1;++i<biHeight;)  for(j=-1;++j<biWidth;)
```

```
        {
            min=256; c=GetPoint(Image,i,j);
            for(int l=numclr;--l>=0;)
                if(min>abs(Id[l]-c))
                {
                    min=abs(Id[l]-c);clr=Id[l];
                }
            SetPoint(Image,i,j,clr);++His[clr];
        }
    }
    for(i=-1;++i<256;)
        if(His[i]>maxhis)
        {
            maxhis=His[i];Maunen=int(i);
        }
    if(dilation==1)
        Dilation(hDIB);
    ::GlobalUnlock(hDIB);
}
/*============================================
 * Hàm: Dilation(HDIB hDIB)
* Mục đích: Giãn nở phục vụ cho việc xử lý
*/
void WINAPI Dilation(HDIB hDIB)
{
    int i,j;
    BYTE c0,c1,c2,c3,c4;

    LPBITMAPINFOHEADER
        lpDIB=(LPBITMAPINFOHEADER)::GlobalLock(hDIB);
    HBYTE Image= ((HBYTE)::FindDIBBits((LPSTR)lpDIB));
    HBYTE Image1=((HBYTE)::FindDIBBits((LPSTR)lpDIB));
```

```
   for(i=0;++i<lpDIB→biHeight;)
   for(j=0;++j<lpDIB→biWidth;)
   {
      c0=GetPoint(Image,i,j);
      c1=GetPoint(Image,i,j+1);
      c2=GetPoint(Image,i,j-1);
      c3=GetPoint(Image,i-1,j);
      c4=GetPoint(Image,i+1,j);
      if(c0!=c1&&c1!=Maunen&&c1==c2)
      {
         SetPoint(Image,i,j,c1);j++;
      }
      else if(c0!=c3&&c3!=Maunen&&c4==c3){ SetPoint(Image,i,j,c3);j++;}
   }
   ::GlobalUnlock(hDIB);
}
/*=============================================
 * Hàm: HoughTransform(RectAngle Rec)
* Mục đích: Biến đổi Hough
*/
void WINAPI HoughTransform(RectAngle Rec)
{
   double phi,x,y;
   long i,j;
   x=double(Rec.bot);
   y=double(Rec.rig+Rec.lef)/2;

   for(i=-1;++i<=360;)
   {
      phi=double(i)/2-90;
      phi=double(phi*pi)/180;
      j=int(x*cos(phi)+y*sin(phi)+0.5);
```

```
      if(j>0)
         ++Hough[i][j];
   }
}

/*=============================================
 * Hàm: DetectAnOject(HBYTE Image,DWORD x,DWORD y,BYTE
clr,DWORD lab,RectAngle   &rec,DWORD &ny)
* Mục đích: Phát hiện các đối tượng phục vụ cho biêns đổi Hough
*/
int DetectAnOject(HBYTE Image,DWORD x,DWORD y,BYTE
clr,DWORD lab,RectAngle   &rec,DWORD &ny)
{
   DWORD y0=y;ny=y;
   int   i,dir,newdir,savedir,prm=0,ch=0;
   tagPoint moi,P[20001];
   while(GetPoint(Image,x,y-1)==clr)
      y=y-1;
   if(Mark[x][y]>0)
   {
      Mark[x][y0]=Mark[x][y]; return(-1);
   }
   Mark[x][y]=lab;
   moi.x=x;moi.y=y;
   rec.top=x;rec.bot=x; rec.lef=y;rec.rig=y;
   savedir=dir=Inverse(Image,x,y,4,clr);
   if(dir==8)  return(-1);
   do
   {
      newdir=Next(Image,moi.x,moi.y,dir,clr);
      moi.x=moi.x+ROW[newdir]; moi.y=moi.y+COL[newdir];
      if(moi.x<rec.top) rec.top=moi.x;
      if(moi.x>rec.bot) rec.bot=moi.x;
```

```
    if(moi.y<rec.lef) rec.lef=moi.y;
    if(moi.y>rec.rig) rec.rig=moi.y;
    if(Mark[moi.x][moi.y]==0)
    {
      P[++prm]=moi; Mark[moi.x][moi.y]=lab;
      if(moi.x==x&&moi.y>ny) ny=moi.y;
    }
    dir = ((((newdir+1)/2)*2)%8+2)%8;
  } while(moi.x!=x||moi.y!=y||Inverse(Image,moi.x,moi.y,dir,clr)!=savedir);
  rec.prm=prm;
  if(prm>=12&&rec.bot-rec.top<=50&&rec.rig-rec.lef<=50)
  {
    if(prm<80&&rec.rig-rec.lef<40&&rec.bot-rec.top<40)
      for( i=0;++i<=prm;)
        SetPoint(TempImage,P[i].x,P[i].y,10);
    return(1);
  }
  if(prm>=600&&(rec.bot-rec.top)>=150&&(rec.rig-rec.lef)>=150)
  {
    for(DWORD j=y-1;++j<biWidth;)
      if(GetPoint(Image,x,j)!=clr)
      {
        ny=j-1;break;
      }
    for(i=0;++i<=prm;)
    {
      SetPoint(Image,P[i].x,P[i].y,Maunen);
      Mark[P[i].x][P[i].y]=0;
    }
  }
  return(0);
}
```

```
/*=============================================
 * Hàm: DetectAndCorrect(HDIB hDIB,int numclr,int dilation)
* Mục đích: Phát hiện và hiệu chỉnh góc nghiêng
*/
HDIB WINAPI DetectAndCorrect(HDIB hDIB,int numclr,int dilation)
{
   HBYTE  Image;
   DWORD  val,i,j,l,prm,wid,hei,nexty,id=0,label=0;
   double goc; tagPoint al;  RectAngle rec;
   HDIB hDIB1=CopyImage(hDIB);
   PreProcessing(hDIB,numclr,dilation);TempDIB=::Taoanh(hDIB);
   LPBITMAPINFOHEADER lpDIB
      =(LPBITMAPINFOHEADER)::GlobalLock(hDIB);
   LPBITMAPINFOHEADER lpDIB1
      =(LPBITMAPINFOHEADER)::GlobalLock(TempDIB);
   Image =((HBYTE)::FindDIBBits((LPSTR)lpDIB));
   TempImage =Image;
   for(i=-1;++i<biHeight;) for(j=-1;++j<biWidth;)
      if(Inverse(Image,i,j,4,GetPoint(Image,i,j))==8)
         SetPoint(Image,i,j,Maunen);
```

```
for(i=-1;++i<biHeight;) for(j=-1;++j<biWidth;)
{
   if(Mark[i][j]==0&&(c=GetPoint(Image,i,j))!=Maunen)
   {
      val=DetectAnArea(Image,i,j,c,++label,rec,nexty);
      if(val>=0) j=nexty;
      if(val==1) Rec[++id]=rec;
   }
   else if(Mark[i][j]>0)
   {
      for(l=biWidth;--l>=j;)
         if(Mark[i][l]==Mark[i][j])
         {
            j=l;break;
         }
   }
}
AverageComponents(Rec,id);al.x=biHeight;al.y=biWidth;
for(i=0;++i<=id;)
{
   prm=Rec[i].prm;wid=Rec[i].rig-Rec[i].lef; hei=Rec[i].bot-Rec[i].top;
   if(prm<10*Cvavr&&wid<5*Widthavr&&hei<5*Heightavr)
      HoughTransform(Rec[i]);
}
SkewEstimation(goc);
if(goc!=0)
{
   if(goc>0&&goc<180)TempDIB=Rote(hDIB1,goc);
   else if(goc<180)  TempDIB=NegativeRote(hDIB1,goc);
   biHeight=lpDIB1→biHeight;biWidth =lpDIB1→biWidth;

   for(i=-1;++i<biHeight;) for(j=-1;++j<biWidth;)
```

```
            if(GetPoint(TempImage,i,j)!=Maunen)
            {
               al.x=i;i=biHeight;break;
            }
         for(j=-1;++j<biWidth;) for(i=-1;++i<biHeight;)
            if(GetPoint(TempImage,i,j)!=Maunen)
            {
               al.y=j;j=biWidth;break;
            }
         if(goc!=0)
            TempDIB=CorrectAlignment(TempDIB,al);
         ::GlobalUnlock(hDIB);::GlobalUnlock(TempDIB);
         return(TempDIB);
      }
      ::GlobalUnlock(hDIB);
      ::GlobalUnlock(TempDIB);
      return(hDIB);
}

/*=============================================
 * Hàm: SkewEstimation(double &goc)
* Mục đích: Ước lượng góc nghiêng
*/
void WINAPI SkewEstimation(double &goc)
{
   double angle;
   DWORD max=0,max1=0,total=0,i,j;
   DWORD Dis=long(sqrt(biWidth*biWidth+biHeight*biHeight));
   if(Dis>=65535)
      Msg(" Vuot qua chi so mang Hough");

   for(i=-1;++i<360;)
```

```
    for(j=-1;++j<=Dis;)
      if(Hough[i][j]>max)
        max=Hough[i][j];
    for(i=-1;++i<360;)
    {
      total=0;
      for(j=-1;++j<=Dis;)
        if(Hough[i][j]>max/2)
          total+=Hough[i][j];
      if(total>max1)
      {
        max1=total;angle=double(i)/2-90;
      }
    }
    goc=double(angle*pi/180);
}

/*================================================
 * Hàm: AverageComponents(RectAngle Rec[15000],int id)
* Mục đích: Tính toán các đối tượng có kích thước trung bình
*/
void WINAPI AverageComponents(RectAngle Rec[15000],int id)
{
    DWORD i,label=0, Prm[3401],Hei[3401],Wid[3401];
    for(i=-1;++i<=3000;)
    {
      Prm[i]=0;Hei[i]=0;Wid[i]=0;
    }
    for(i=-1;++i<=id;)
    {
      if(Rec[i].prm<=400)
        ++Prm[Rec[i].prm];
```

```
        if(Rec[i].rig-Rec[i].lef<=400)
            ++Wid[Rec[i].rig-Rec[i].lef];
        if(Rec[i].bot-Rec[i].top<=400)
            ++Hei[Rec[i].bot-Rec[i].top];
    }
    Cvavr=Averaging(Prm,16,400);
    Widthavr=Averaging(Wid,8,100);
    Heightavr=Averaging(Hei,8,100);
}

/*=============================================
 * Hàm: Averaging(DWORD Av[400],WORD bn,WORD id)
* Mục đích: Phục vụ cho việc tính toán các đối tượng có kích thước trung bình
*/
WORD WINAPI Averaging(DWORD Av[400],WORD bn,WORD id)
{
    if(bn%2!=0)
    {
        Msg("ib%2!=0");
        return(0);
    }
    WORD i=0,j=0,a,b,dx;
    DWORD max=0,max1=0,max2=0,His[201];
    for(i=-1;++i<=200;)
        His[i]=0;
    for(i=-1;++i<=id;)
        His[i/bn]+=Av[i];
    for(i=-1;++i<=id/bn;)
        if(His[i]>max){ max=His[i];dx=i; }
    a=dx*bn;b=a+bn;
    while(1)
```

```
    {
        max1=0;max2=0;
        for(i=a;i<a+bn/2;++i)
            max1+=Av[i];
        for(i=a+bn/2;i<b;++i)
            max2+=Av[i];
        if(max1>max2)
            b=a+bn/2;
        else
            a=a+bn/2;
        bn=bn/2;
        if(bn==2)
            return(int(b));
    }
}
```

TÀI LIỆU THAM KHẢO

[1]. Phạm Việt Bình (2007), *Phát triển kỹ thuật dò biên, phát hiện biên và ứng dụng*, Luận án Tiến sỹ.

[2]. Phạm Việt Bình (2006), "Một số tính chất của phép toán hình thái và ứng dụng trong phát hiện biên", *Tạp chí Tin học và Điều khiển học*, Tập 22, số 2, 2006, 155-163.

[3]. Phạm Việt Bình, Cao Lê Mạnh Hà, Đỗ Năng Toàn (2005), "Một cách tiếp cận mới trong phát hiện biên của ảnh đa cấp xám", *Kỷ yếu Hội thảo Quốc gia lần thứ 8 - Một số vấn đề chọn lọc của Công nghệ Thông tin và Truyền thông*, Hải Phòng 25-27/08 /2005, Nxb KH&KT, Hà Nội 2006, 92-102.

[4]. Phạm Việt Bình, Ngô Mạnh Hùng, Đỗ Năng Toàn (2005), "Một cải tiến thuật toán dò biên và ứng dụng trong làm mảnh đối tượng", *Kỷ yếu Hội thảo Khoa học Quốc gia lần thứ 2 - nghiên cứu cơ bản và ứng dụng Công nghệ Thông tin-FAIR'05*, TP Hồ Chí Minh 23-24/09/2005, Nxb KH&KT, Hà Nội 2006, 477-485.

[5]. Đỗ Năng Toàn, Phạm Văn Dũng, Phạm Việt Bình (2005), "Ứng dụng chu tuyến trong phát hiện góc nghiêng văn bản", *Kỷ yếu Hội thảo Quốc gia lần thứ 7 - Một số vấn đề chọn lọc của Công nghệ Thông tin và Truyền thông*, Đà Nẵng 18-20/08 /2004, Nxb KH&KT, Hà Nội 2005, 432-441.

[6]. Đỗ Năng Toàn, Phạm Việt Bình (2004), "Một thuật toán cắt chữ in dính ở mức từ dựa vào chu tuyến", chuyên san Các công trình nghiên cứu và triển khai Công nghệ Thông tin và Viễn thông, *Tạp chí Bưu chính Viễn thông & Công nghệ Thông tin*, số 12 tháng 8/2004, 50-56.

[7]. Lương Mạnh Bá, Nguyễn Thanh Thủy (2002), *Nhập Môn Xử lý ảnh số*, Nxb Khoa học và Kỹ thuật, 2002.

[8]. Ngô Quốc Tạo, Đỗ Năng Toàn (2001), "Tách bảng dựa trên tập các hình chữ nhật rời rạc", chuyên san Các công trình nghiên cứu và triển khai Công nghệ thông tin và viễn thông, *Tạp chí Bưu chính Viễn thông*, số 5 năm 2001, 73-79.

[9]. Đỗ Năng Toàn (2001), *Nghiên cứu một số phương pháp biểu diễn hình dạng và ứng dụng trong nhận dạng ảnh*, Luận án Tiến sỹ.

[10]. J.R.Paker (1997), *Algorithms for Image processing and Computer Vision*. John Wiley & Sons, Inc.

[11]. Randy Crane (1997), *A simplified approach to image processing,* Prentice-Hall, Inc.

[12]. John C.Russ (1995), *The Image Procesing Handbook.* CRC Press, Inc.

[13]. Adrian Low (1991), *Introductory Computer Vision and Image Processing,* Copyright (c) 1991 by McGrow Hill Book Company (UK) Limited.

[14]. Anil K.Jain (1989), *Fundamental of Digital Image Processing. Prentice Hall,* Engwood cliffs.

[15]. T. Pavlidis (1982), *Algorithms for Graphics and Image Processing,* Computer Science Press.