



C BÁCH KHOA HÀ NỘI

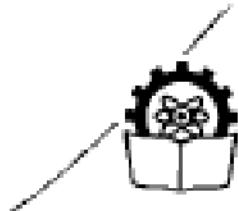
\* S 2 0 1 0 0 0 0 0 0 6 \*

# LƯƠNG MẠNH BÁ NGUYỄN THANH THUÝ

Ket-noi.com chia sẻ miễn phí



XỬ LÝ ẢNH HÌNH SỐ



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

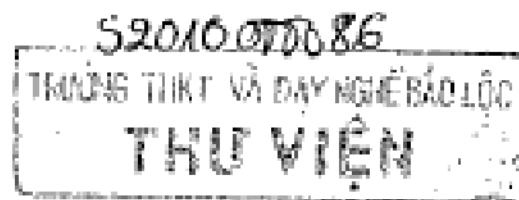
006.6

LU-B

LUONG MẠNH BÁ (chủ biên)  
NGUYỄN THANH THỦY

# NHẬP MÔN XỬ LÝ ẢNH SỐ

(Xuất bản lần thứ tư có chỉnh lý bổ sung)



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT  
HÀ NỘI - 2006

Chủ trách nhiệm xuất bản: **PGS, TS. Tô Đăng Hải**  
Biên tập: **Đặng Đình Thạch, Ngọc Khuê**  
Sửa bản in: **Đình Thach**

---

In 800 cuốn khổ 16 x 24 cm tại Công ty cổ phần In Hàng Không  
Quyết định xuất bản số: 136-2006/CXB/275-06 KHKT  
In xong và nộp lưu chiểu tháng 1 năm 2007

## *LỜI TƯA*

(Cho lần xuất bản thứ ba)

Trong lần xuất bản thứ hai, cuốn sách đã được chỉnh lý về nội dung cũng như hình thức. Chúng tôi đã nhận được nhiều e-mail và thư góp ý của các thầy cô, bạn bè đồng nghiệp, và nhiều bạn sinh viên. Để phục vụ bạn đọc được tốt hơn, trong lần xuất bản này chúng tôi đã sửa chữa một số lỗi lẩn loát, biên tập và đưa thêm một số bài tập để cung cấp các kỹ thuật tính nhằm giúp cho quá trình tự đọc, cũng như giúp cho sinh viên hiểu rõ hơn qui trình và các kỹ thuật trước khi bắt tay viết các chương trình thử nghiệm hay các ứng dụng.

Chúng tôi xin chân thành cảm ơn các thầy cô trong Khoa Công Nghệ thông tin, các Giáo sư, các đồng nghiệp đã cho những ý kiến quan trọng góp phần nâng cao chất lượng cuốn sách. Chúng tôi cũng chân thành cảm ơn các bạn sinh viên các trường đại học đã sử dụng tài liệu này trong quá trình học tập, nghiên cứu và cho những góp ý chân tình. Chúng tôi hy vọng cuốn sách sẽ có ích cho các bạn và mong nhận được các ý kiến đóng góp để hoàn thiện hơn nữa cho các lần tái bản sau này. Địa chỉ liên lạc:

Email: [balm@it-hut.edu.vn](mailto:balm@it-hut.edu.vn)

[thuynt@it-hut.edu.vn](mailto:thuynt@it-hut.edu.vn)

*Hà Nội, tháng 11 năm 2002*

*Các tác giả*

# LỜI MỞ ĐẦU

## <Lần xuất bản thứ nhất>

Xử lý ảnh là một lĩnh vực đang được quan tâm và đã trở thành một môn học chuyên ngành của sinh viên hệ kỹ sư, hệ cử nhân ngành Công Nghệ Thông Tin, cũng như một số ngành kỹ thuật khác trong các trường Đại học kỹ thuật. Tuy nhiên, tài liệu và giáo trình còn là một điều nan giải đối với sinh viên. Hiện tại chỉ có ít tài liệu bằng tiếng Anh, tiếng Pháp hay tiếng Nga. Tài liệu bằng tiếng Việt còn rất ít. Với mong muốn đóng góp vào sự nghiệp đào tạo và nghiên cứu trong lĩnh vực này, chúng tôi biên soạn cuốn sách "Nhập môn xử lý ảnh số" để làm tài liệu tham khảo trong việc giảng dạy và triển khai ứng dụng.

Xử lý ảnh có liên quan đến nhiều ngành khác như: hệ thống tin học, lý thuyết thông tin, lý thuyết thống kê, trí tuệ nhân tạo, nhận dạng, v.v. Do đó để có thể tiếp thu kiến thức một cách thuận lợi, tài liệu được phân bổ một cách hợp lý giữa các vấn đề, không đi sâu nhiều vào các phát biểu toán học gây khó hiểu, khó theo dõi. Ngoài phần lý thuyết, chúng tôi cũng đề cập tới các kỹ thuật được sử dụng trong xử lý ảnh hiện nay. Các lời giải được thể hiện dưới dạng giải thuật bằng ngôn ngữ thuật giải. Ngoài ra, có một số modul chương trình viết bằng Turbo C được giới thiệu trong phần phụ lục.

Với ý tưởng trên, để trình bày một cách logic các vấn đề và các kỹ thuật trong xử lý ảnh, tài liệu được phân bổ thành 8 chương và hai phụ lục. Chương 1 là phần nhập môn của xử lý ảnh. Mục đích của chương này nhằm giới thiệu các giai đoạn cơ bản trong xử lý ảnh, các thành phần của một hệ xử lý ảnh số bằng máy tính. Chương 2 giới thiệu về quá trình thu nhận ảnh, cách lấy mẫu và lượng tử hoá ảnh và các kiểu tệp được dùng để lưu trữ ảnh. Chương 3 trình bày các kỹ thuật và các công cụ biến đổi ảnh như biến đổi Fourier, Karhunen Loeve. Chương 4 trình bày các kỹ thuật nâng cao chất lượng ảnh (tiến xử lý ảnh). Ảnh thu nhận được, do nhiều nguyên nhân có thể bị suy biến hoặc là do mục đích của giai đoạn phân tích - xử lý ảnh, chúng cần được tăng cường để có chất lượng cao hơn. Chương 5 và chương 6 là nội dung của giai đoạn phân tích ảnh. Trong phần này, các kỹ thuật xác định biên (một vấn đề chủ yếu trong phân tích ảnh) được đề cập một cách chi tiết bao gồm: kỹ thuật lọc vi phân, kỹ thuật Gradient, Laplace. Tiếp sau là các kỹ thuật phân đoạn ảnh thông dụng và hiệu quả như: Quad-Tree, Merge, lõm mảnh biên, nhị phân hoá biên. Chương 7 đề cập tới vấn đề nhận dạng và các ứng dụng của nó trong nhận dạng chữ viết, nhận dạng vân tay. Chương 8 giới thiệu về nén ảnh và các phương pháp nén ảnh hiện

được dùng.

Ngoài ra, trong xử lý ảnh còn nhiều vấn đề rất đáng quan tâm nhưng chưa được trình bày trong tài liệu này như: mô tả đầy đủ về đối tượng, trích chọn đặc trưng, phân lớp ảnh, hoặc khôi phục ảnh từ hình chiếu rất có ích trong các ứng dụng y học.

Trong quá trình biên soạn, chúng tôi đã nhận được các ý kiến rất quý báu, sự giúp đỡ nhiệt tình của các giáo sư và bạn bè đồng nghiệp trong và ngoài khoa.

Chúng tôi xin cảm ơn PGS, TS Nguyễn Văn Ba, ThS Đỗ Văn Uy đã dành nhiều thời gian đọc kỹ bản thảo và cho nhiều ý kiến quý báu.

Chúng tôi cũng bày tỏ lòng biết ơn đối với Ban Chủ nhiệm khoa Công Nghệ Thông Tin - Đại Học Bách Khoa Hà Nội, Hội đồng Khoa học - Đào tạo Khoa đã tạo mọi điều kiện để tài liệu này sớm ra mắt bạn đọc.

*Các tác giả*

## 1

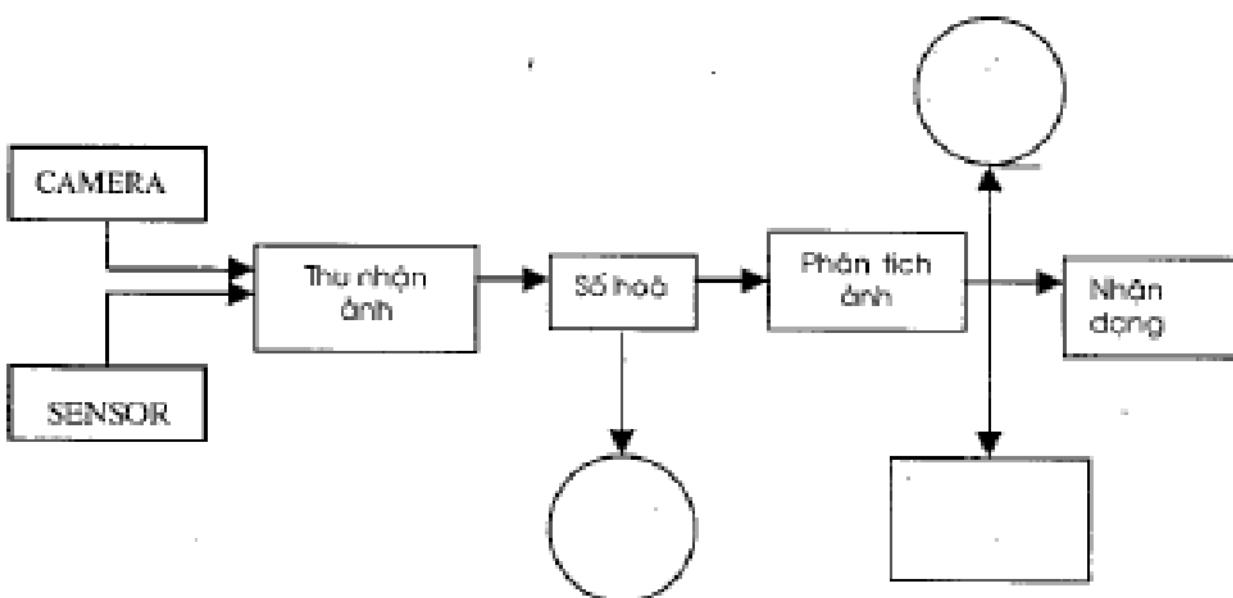
## NHẬP MÔN XỬ LÝ ẢNH

## INTRODUCTION TO DIGITAL IMAGE PROCESSING

## 1.1. TỔNG QUAN VỀ MỘT HỆ THỐNG XỬ LÝ ẢNH

Xử lý ảnh là một khoa học còn tương đối mới mẻ so với nhiều ngành khoa học khác, nhất là trên qui mô công nghiệp, song trong xử lý ảnh đã bắt đầu xuất hiện những máy tính chuyên dụng. Để có thể hình dung cấu hình một hệ thống xử lý ảnh chuyên dụng hay một hệ thống xử lý ảnh dùng trong nghiên cứu, đào tạo, trước hết chúng ta sẽ xem xét các bước cần thiết trong xử lý ảnh.

Trước hết là quá trình **thu nhận ảnh**. Ảnh có thể thu nhận qua camera. Thường ảnh thu nhận qua camera là tín hiệu tương tự (loại camera ống kính CCIR), nhưng cũng có thể là tín hiệu số hoá (loại CCD - Charge Coupled Device).



Hình 1.1.a. Các giai đoạn chính trong xử lý ảnh

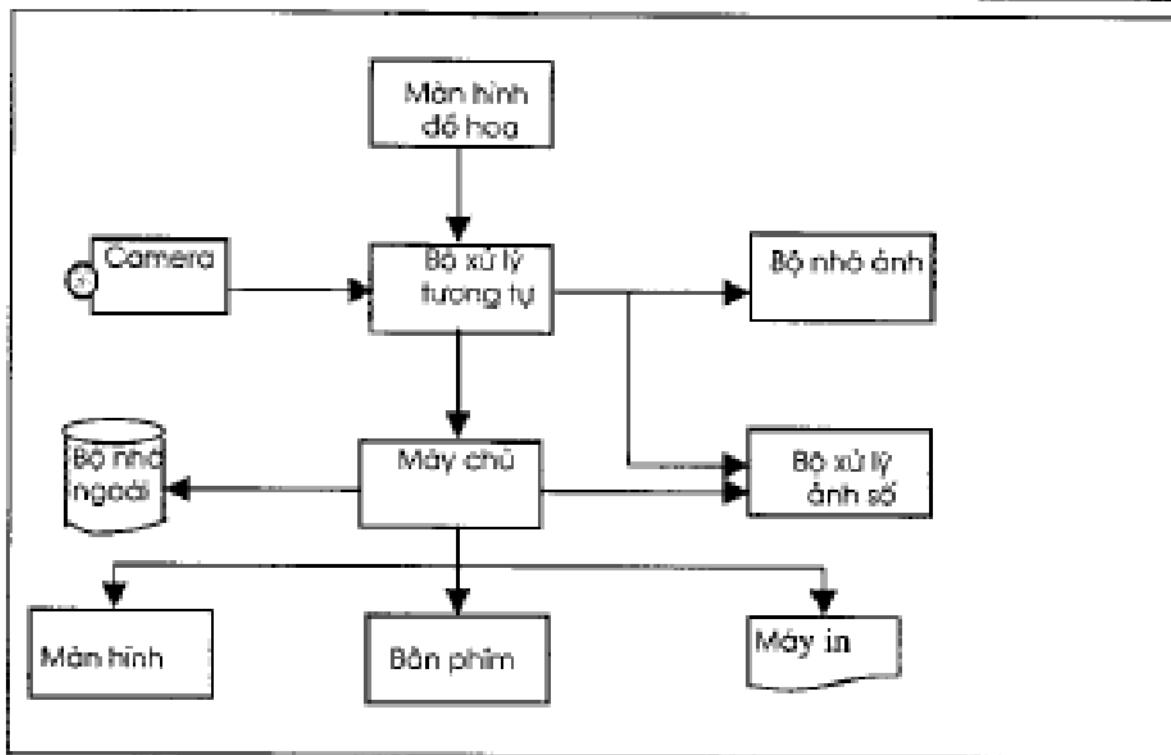
Ảnh cũng có thể thu nhận từ vệ tinh qua các bộ cảm ứng (sensor), hay ảnh, tranh được quét trên scanner. Chi tiết về quá trình thu nhận ảnh sẽ được mô tả trong chương 2. Tiếp theo là quá trình *số hoá* (Digitalizer) để biến đổi tín hiệu tương tự sang tín hiệu rời rạc (lấy mẫu) và số hoá bằng lượng hoá, trước khi chuyển sang giai đoạn xử lý, phân tích hay lưu trữ lại.

Quá trình **phân tích ảnh** thực chất bao gồm nhiều công đoạn nhỏ. Trước hết là công việc tăng cường ảnh (Image Enhancement) để nâng cao chất lượng ảnh. Do những nguyên nhân khác nhau: có thể do chất lượng thiết bị thu nhận ảnh, do nguồn sáng hay do nhiễu, ảnh có thể bị suy biến. Do vậy cần phải tăng cường và khôi phục (Image Restoration) lại ảnh để làm nổi bật một số đặc tính chính của ảnh, hay làm cho ảnh gần giống nhất với trạng thái gốc- trạng thái trước khi ảnh bị biến dạng. Giai đoạn tiếp theo là phát hiện các đặc tính như biên (Edge Detection), phân vùng ảnh (Image Segmentation), trích chọn các đặc tính (Feature Extraction), v.v...

Cuối cùng, tùy theo mục đích của ứng dụng, sẽ là giai đoạn nhận dạng, phân lớp hay các quyết định khác. Các giai đoạn chính của quá trình xử lý ảnh có thể mô tả ở hình 1.1.a.

Với các giai đoạn trên, một hệ thống xử lý ảnh (cấu trúc phần cứng theo chức năng) gồm các thành phần tối thiểu như hình 1.1.b.

- Đối với một hệ thống xử lý ảnh thu nhận qua camera - camera như là con mắt của hệ thống. Có 2 loại camera: camera ống loại CCIR và camera CCD. Loại camera ứng với chuẩn CCIR quét ảnh với tần số 1/25 và mỗi ảnh gồm 625 dòng. Loại CCD gồm các photo diode và làm tương ứng một cường độ sáng tại một điểm ảnh với một phần tử ảnh (pixel). Như vậy, ảnh là tập hợp các điểm ảnh. Số pixel tạo nên một ảnh gọi là độ phân giải (resolution).
- Bộ xử lý tương tự (analog processor). Bộ phận này thực hiện các chức năng sau:
  - Chọn camera thích hợp nếu hệ thống có nhiều camera.
  - Chọn màn hình hiển thị tín hiệu.
  - Thu nhận tín hiệu video thu nhận bởi bộ số hoá(digitalizer). Thực hiện lấy mẫu và mã hoá.
  - Tiến xử lý ảnh khi thu nhận: dùng kỹ thuật bảng tra (Look Up Table - LUT).



Hình 1.1b. Các thành phần chính của hệ thống xử lý ảnh.

- **Bộ xử lý ảnh số.** Gồm nhiều bộ xử lý chuyên dụng: xử lý lọc, trích chọn đường bao, nhị phân hóa ảnh. Các bộ xử lý này làm việc với tốc độ 1/25 giây.
- **Máy chủ.** Đóng vai trò điều khiển các thành phần miêu tả ở trên.
- **Bộ nhớ ngoài:** Dữ liệu ảnh cũng như các kiểu dữ liệu khác, để có thể chuyển giao cho các quá trình khác, nó cần được lưu trữ. Để có một ước lượng, xét thí dụ sau: một ảnh đen trắng cỡ 512 x 512 với 256 mức xám chiếm 256K bytes ( $2^9 * 2^9 = 2^8 * 2^{10}$ ). Với một ảnh màu cùng kích thước dung lượng sẽ tăng gấp 3 lần.

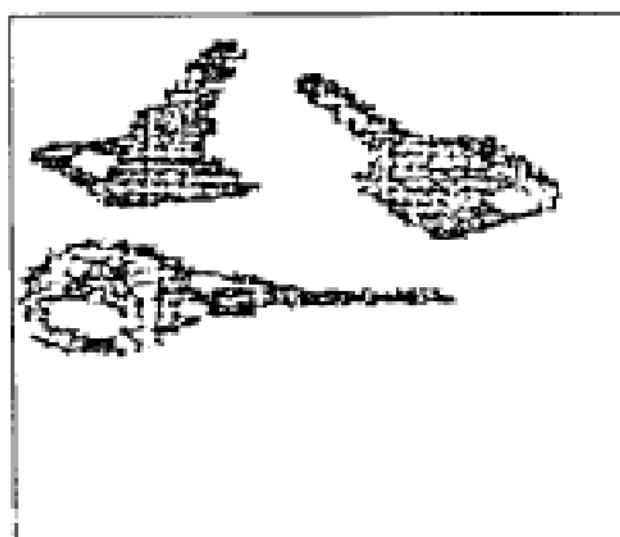
## 1.2 CÁC VẤN ĐỀ CƠ BẢN TRONG XỬ LÝ ẢNH

Như đã đề cập trong phần giới thiệu, chúng ta đã thấy được một cách khái quát các vấn đề chính trong xử lý ảnh. Để hiểu chi tiết hơn, trước tiên ta xem xét hai khái niệm (thuật ngữ) thường dùng trong xử lý ảnh đó là pixel (phản tử ảnh) và grey level (mức xám), tiếp theo là tóm tắt các vấn đề chính.

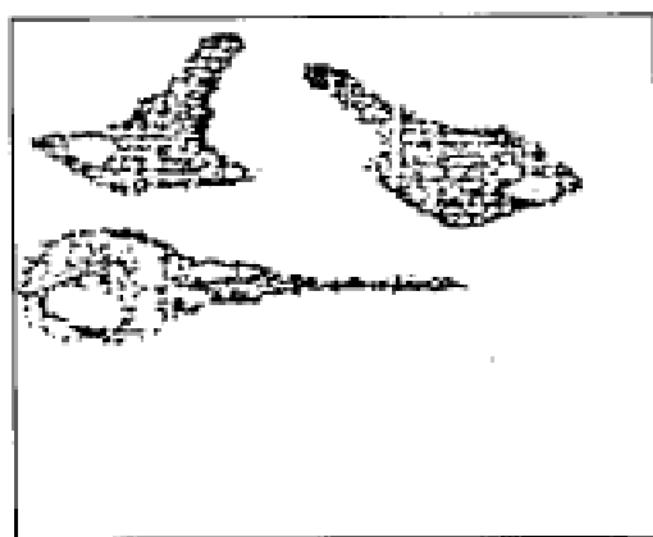
### 1.2.1. Một số khái niệm

- **Pixel (Picture Element):** phản tử ảnh

Ảnh trong thực tế là một ảnh liên tục về không gian và về giá trị độ sáng. Để có thể xử lý ảnh bằng máy tính cần thiết phải tiến hành số hoá ảnh. Trong quá trình số hoá, người ta biến đổi tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng hóa thành phần giá trị (rời rạc hóa biến độ giá trị) mà về nguyên tắc bằng mắt thường không phân biệt được hai mức kẽ nhau. Trong quá trình này, người ta sử dụng khái niệm *Picture element* mà ta quen gọi hay viết là *Pixel* - phần tử ảnh. Ở đây cũng cần phân biệt khái niệm pixel hay đố cộp đến trong các hệ thống đồ họa máy tính. Để tránh nhầm lẫn ta tạm gọi khái niệm pixel này là pixel thiết bị. Khái niệm pixel thiết bị có thể xem xét như sau: khi ta quan sát màn hình (trong chế độ đồ họa), màn hình không liên tục mà gồm nhiều điểm nhỏ, gọi là pixel. Mỗi pixel gồm một cặp tọa độ x, y và màu.



a) Ảnh với độ phân giải 128 x 128



b) Ảnh với độ phân giải 64 x 64

Hình 1.2. Biểu diễn ảnh với độ phân giải khác nhau.

Cặp tọa độ x, y tạo nên *độ phân giải* (resolution). Như màn hình máy tính có nhiều loại với độ phân giải khác nhau: màn hình CGA có độ phân giải là 320 x 200; màn hình VGA là 640 x 350....

Như vậy, một ảnh là một tập hợp các điểm ảnh. Khi được số hoá, nó thường được biểu diễn bởi bảng hai chiều  $I(n,p)$ : n dòng và p cột. Ta nói ảnh gồm  $n \times p$  pixels. Người ta thường ký hiệu  $I(x,y)$  để chỉ một pixel. Thường giá trị của n chọn bằng p và bằng 256. Hình 1.2 cho ta thấy việc biểu diễn một ảnh với độ phân giải khác nhau. Một pixel có thể lưu trữ trên 1, 4, 8 hay 24 bit.

- Gray level: Mức xám

Mức xám là kết quả sự mã hóa tương ứng một cường độ sáng của mỗi điểm ảnh với một giá trị số - kết quả của quá trình lượng hóa. Cách mã hóa kinh điển thường dùng 16, 32 hay 64 mức. Mã hóa 256 mức là phổ dụng nhất do lý do kỹ thuật. Vì  $2^8 = 256$  (0, 1, ..., 255), nên với 256 mức, mỗi pixel sẽ được mã hóa bởi 8 bit.

### 1.2.2. Biểu diễn ảnh

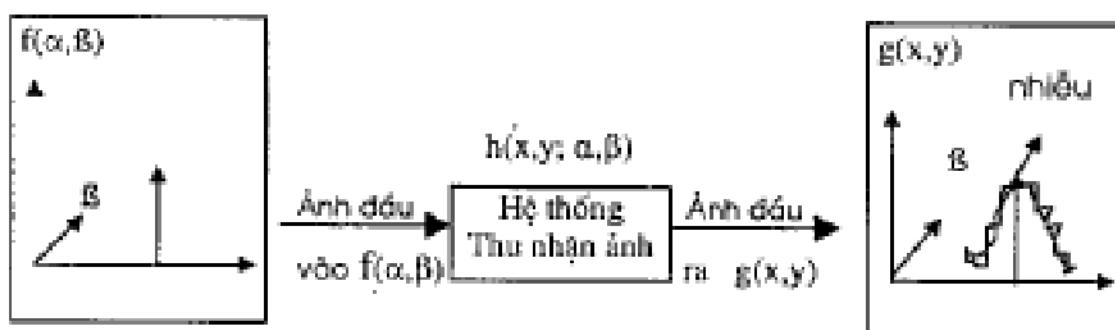
Trong biểu diễn ảnh, người ta thường dùng các phần tử đặc trưng của ảnh là pixel. Nhìn chung có thể xem một hàm hai biến chứa các thông tin như biểu diễn của một ảnh. Các mô hình biểu diễn ảnh cho ta một mô típ logic hay định lượng các tính chất của hàm này. Trong biểu diễn ảnh cần chú ý đến tính trung thực của ảnh hoặc các tiêu chuẩn "thông minh" để đo chất lượng ảnh hoặc tính hiệu quả của các kỹ thuật xử lý.

Việc xử lý ảnh số yêu cầu ảnh phải được mẫu hóa và lượng tử hóa. Thí dụ một ảnh ma trận 512 dòng gồm khoảng 512 x 512 pixel. Việc lượng tử hóa ảnh là chuyển đổi tín hiệu tương tự sang tín hiệu số (Analog Digital Convert) của một ảnh đã lấy mẫu sang một số hữu hạn mức xám. Vấn đề này sẽ trình bày chi tiết trong chương 2.

Một số mô hình thường được dùng trong biểu diễn ảnh: Mô hình toán, mô hình thống kê. Trong mô hình toán, ảnh hai chiều được biểu diễn nhờ các hàm hai biến trực giao gọi là các *hàm cơ sở*. Các biến đổi này sẽ trình bày kỹ trong chương 3. Với mô hình thống kê, một ảnh được coi như một phần tử của một tập hợp đặc trưng bởi các đại lượng như: kỳ vọng toán học, hiệp biến, phương sai, moment.

### 1.2.3. Tăng cường ảnh - khôi phục ảnh

Tăng cường ảnh là bước quan trọng, tạo tiền đề cho xử lý ảnh. Nó gồm một loạt các kỹ thuật như: tăng cường độ tương phản, khử nhiễu, nổi màu, v...v.



Hình 1.3 Ảnh biến dạng do nhiễu.

Hình 1.3 ở trên cho ta thí dụ về sự biến dạng của ảnh do nhiễu.

- Khôi phục ảnh là nhằm loại bỏ các suy giảm (degradation) trong ảnh. Với một hệ thống tuyến tính, ảnh của một đối tượng có thể biểu diễn bởi:

$$g(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x,y; \alpha, \beta) f(\alpha, \beta) d\alpha d\beta + \eta(x,y)$$

Trong đó:

- $\eta(x,y)$  là hàm biểu diễn nhiễu cộng.
- $f(\alpha, \beta)$  là hàm biểu diễn đối tượng.
- $g(x,y)$  là ảnh thu nhận.
- $h(x,y; \alpha, \beta)$  là hàm tán xạ điểm (Point Spread Function - PSF).

Một vấn đề khôi phục ảnh tiêu biểu là tìm một xấp xỉ của  $f(\alpha, \beta)$  khi PSF của nó có thể do lưỡng hay quan sát được, ảnh mờ và các tính chất sác xuất của quá trình nhiễu.

#### 1.2.4. Biến đổi ảnh

Thuật ngữ biến đổi ảnh (Image Transform) thường dùng để nói tới một lớp các ma trận đơn vị và các kỹ thuật dùng để biến đổi ảnh. Cũng như các tín hiệu một chiều được biểu diễn bởi một chuỗi các hàm cơ sở, ảnh cũng có thể được biểu diễn bởi một chuỗi rời rạc các ma trận cơ sở gọi là *ảnh cơ sở*. Phương trình ảnh cơ sở có dạng:

$A^*_{k,l} = a_k a_l^{-T}$ , với  $a_k$  là cột thứ k của ma trận A. A là ma trận đơn vị. Có nghĩa là A  $A^{*T} = I$ . Các  $A^*_{k,l}$  định nghĩa ở trên với  $k, l = 0, 1, \dots, N-1$  là ảnh cơ sở. Có nhiều loại biến đổi được dùng như :

- Biến đổi Fourier, Sin, Cosin, Hadamard, ...
- Tích Kronecker ( ${}^*$ )
- Biến đổi KL (Karhunen Loeve); biến đổi này có nguồn gốc từ khai triển của các quá trình ngẫu nhiên gọi là phương pháp trích chọn các thành phần chính.

Do phải xử lý nhiều thông tin, các phép toán nhân và cộng trong khai triển là khá lớn. Do vậy, các biến đổi trên nhằm làm giảm thử nguyên của ảnh để việc xử lý ảnh được hiệu quả hơn. Các biến đổi này sẽ được mô tả chi tiết trong chương 3.

<sup>1</sup> Trong xử lý ảnh, việc phân tích có thể được đơn giản hơn khá nhiều do làm việc với ma trận khối gọi là tích Kronecker

- *Má trận khối là má trận mà các phần tử của nó lại là một má trận.*

$$\begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ \dots & \dots & \dots & \dots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

**Má trận  $A$**

với  $A_{ij}$  là má trận  $m \times n$ ;  $i = 1, 2, \dots, m$  và  $j = 1, 2, \dots, n$ .

Tích Kronecker

Cho  $A$  là má trận kích thước  $M_1 \times M_2$  và  $B$  má trận kích thước  $N_1 \times N_2$ .

Tích Kronecker của  $A$  và  $B$  ký hiệu là  $A \otimes B$  là má trận khối được định nghĩa:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & A_{1,M_2}B \\ \dots & \dots & \dots & \dots \\ a_{M_1,1}B & a_{M_1,2}B & \dots & A_{M_1,M_2}B \end{bmatrix}$$

với  $a_{ij}$  là các phần tử của má trận  $A$ .

Thí dụ      má trận  $A$       má trận  $B$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\text{thì } A \otimes B = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \\ 1 & 2 & -1 & -2 \\ 3 & 4 & -3 & -4 \end{bmatrix}$$

### 1.2.5. Phân tích ảnh

Phân tích ảnh liên quan đến việc xác định các độ đo định lượng của một ảnh để đưa ra một mô tả đầy đủ về ảnh. Các kỹ thuật được sử dụng ở đây nhằm mục đích xác định biến của ảnh. Có nhiều kỹ thuật khác nhau như lọc vi phân hay đồ họa theo quy hoạch động. Vấn đề xác định biến cùng các kỹ thuật liên quan sẽ được trình bày trong chương 5.

Người ta cũng dùng các kỹ thuật để phân vùng ảnh. Từ ảnh thu được, người ta tiến hành kỹ thuật tách (split) hay hợp (fusion) dựa theo các tiêu chuẩn đánh giá như: màu sắc,

cường độ, v...v. Các phương pháp được biết đến như Quad-Tree, mành hoá biến, nhị phân hoá đường biến. Cuối cùng, phải kể đến các kỹ thuật phân lớp dựa theo cấu trúc. Vấn đề này được trình bày trong chương 6.

### 1.2.6. Nhận dạng ảnh

Nhận dạng ảnh là quá trình liên quan đến các mô tả đối tượng mà người ta muốn đặc tả nó. Quá trình nhận dạng thường đi sau quá trình trích chọn các đặc tính chủ yếu của đối tượng. Có hai kiểu mô tả đối tượng:

- Mô tả tham số (nhận dạng theo tham số).
- Mô tả theo cấu trúc (nhận dạng theo cấu trúc).

Trên thực tế, người ta đã áp dụng kỹ thuật nhận dạng khá thành công với nhiều đối tượng khác nhau như: nhận dạng ảnh vân tay, nhận dạng chữ (chữ cái, chữ số, chữ có dấu).

Nhận dạng chữ in hoặc đánh máy phục vụ cho việc tự động hoá quá trình đọc tài liệu, tăng nhanh tốc độ và chất lượng thu nhận thông tin từ máy tính.

Nhận dạng chữ viết tay (với mức độ ràng buộc khác nhau về cách viết, kiểu chữ, v.v...) phục vụ cho nhiều lĩnh vực.

Ngoài hai kỹ thuật nhận dạng trên, hiện nay một kỹ thuật nhận dạng mới dựa vào kỹ thuật mạng nơron đang được áp dụng và cho kết quả khả quan. Một số khái niệm về mạng nơron cũng như một ứng dụng mạng nơron trong nhận dạng ký tự sẽ được đề cập đến trong chương 7.

### 1.2.7. Nén ảnh

Dữ liệu ảnh cũng như các dữ liệu khác cần phải lưu trữ hay truyền đi trên mạng. Như đã nói ở trên, lượng thông tin để biểu diễn cho một ảnh là rất lớn. Trong phần 1.1 chúng ta đã thấy một ảnh đen trắng cỡ 512 x 512 với 256 mức xám chiếm 256K bytes. Do đó làm giảm lượng thông tin hay nén dữ liệu là một nhu cầu cần thiết. Nhiều phương pháp nén dữ liệu đã được nghiên cứu và áp dụng cho loại dữ liệu đặc biệt này. Các kỹ thuật nén ảnh sẽ được trình bày trong chương 8.

### Bài tập chương I

Để có ảnh cho các phần sau, hãy tạo một ảnh bằng một số phần mềm có sẵn (như Paint Brush) hoặc xây dựng một ảnh gồm *một số hình chữ nhật* theo thuật toán sau:

- Mỗi hình chữ nhật cho các tọa độ góc trên, góc dưới cùng như mức xám tương ứng (số mức xám hoặc nhập vào hay reo ngẫu nhiên một số từ 0 đến 255). Dữ liệu nhập vào từ bàn phím.

- Quá trình dừng khi cho mức xám là -1.

- Lưu ảnh lên tệp để cho các xử lý tiếp sau.

Chương trình viết trong bảng ngôn ngữ tự chọn.

# 2

## THU NHẬN ẢNH IMAGE REPRESENTATION AND MODELING

Chương này giới thiệu quá trình thu nhận ảnh cũng như các thiết bị dùng trong hệ thống xử lý ảnh. Tiếp theo là quá trình lấy mẫu và lượng tử hóa ảnh. Đồng thời cũng giới thiệu một số phương pháp biểu diễn ảnh, các kiểu tệp và cấu trúc của chúng dùng trong lưu trữ ảnh như .IMG, .PCX, TIFF, JPEG... Cuối cùng, trình bày nguyên tắc tái hiện ảnh gồm các kỹ thuật Bayer Dithering, Rylander Pattern Matrix...

### 2.1. CÁC THIẾT BỊ THU NHẬN ẢNH VÀ KỸ THUẬT PHÂN TÍCH MÀU

#### 2.1.1. Thiết bị thu nhận ảnh

Một hệ thống xử lý ảnh có thể trang bị kèm theo các hệ thống thông tin địa lý - GIS (Geographical Information System) hay hệ MORPHO(giá khoảng 7 đến 8 triệu USD) hoặc có thể là hệ thống máy tính cá nhân. Các thiết bị thu ảnh thông thường gồm máy quay (camera) cộng với bộ chuyển đổi tương tự số AD(Analog to Digital) hoặc máy quét (scanner) chuyên dụng.

Các thiết bị thu nhận ảnh này có thể cho ảnh trắng đen B/W (Black & White) với mật độ từ 400 đến 1600 dpi (dot per inch) hoặc ảnh màu 600 dpi. Với ảnh B/W mức màu z là 0 hoặc 1. Với ảnh đa cấp xám, mức xám biến thiên từ 0 đến 255. Ảnh màu, mỗi điểm ảnh lưu trữ trong 3 bytes và do đó ta có  $2^{8 \times 3} = 2^{24}$  màu (cứ 16,7 triệu màu).

Các thiết bị thu nhận ảnh này có thể cho ảnh trắng đen B/W (Black & White) với mật độ từ 400 đến 1600 dpi (dot per inch) hoặc ảnh màu 600 dpi. Với ảnh B/W mức màu z là 0 hoặc 1. Với ảnh đa cấp xám, mức xám biến thiên từ 0 đến 255. Ảnh màu, mỗi điểm ảnh lưu trữ trong 3 bytes và do đó ta có  $2^{8 \times 3} = 2^{24}$  màu (cứ 16,7 triệu màu).

Khi dùng scanner, một dòng photodiot sẽ quét ngang ảnh (quét theo hàng) và cho ảnh với độ phân giải ngang khá tốt. Đầu ra của scanner là ảnh ma trận số mà ta quen gọi là bản đồ ảnh (ảnh Bitmap). Bộ số hoá (digitizer) sẽ tạo ảnh vector có hướng.

TRƯỜNG THPT VÀ ĐIỂM HỌC SÁCH LỌC

THƯ VIỆN

ĐÀ NẴNG - VIỆT NAM

Trong xử lý ảnh bằng máy tính, ta không thể không nói đến thiết bị monitor (màn hình) để hiển thị ảnh. Monitor có nhiều loại khác nhau:

- CGA : 640 x 320 x 16 màu,
- EGA : 640 x 350 x 16 màu,
- VGA : 640 x 480 x 16 màu,
- SVGA: 1024 x 768 x 256 màu.

Với ảnh màu, có nhiều cách tổ hợp màu khác nhau. Theo lý thuyết màu do Thomas đưa ra từ năm 1802, mọi màu đều có thể tổ hợp từ 3 màu cơ bản: Red (đỏ), Green (lục) và Blue (tím).

Thiết bị ra ảnh có thể là máy in đen trắng, máy in màu hay máy vẽ (plotter). Máy vẽ cũng có nhiều loại: loại dùng bút, loại phun mực.

Nhìn chung, các hệ thống thu nhận ảnh thực hiện 2 quá trình:

- Cảm biến: biến đổi năng lượng quang học (ánh sáng) thành năng lượng điện.
- Tổng hợp năng lượng điện thành ảnh điện.

### 2.1.2. Biểu diễn màu

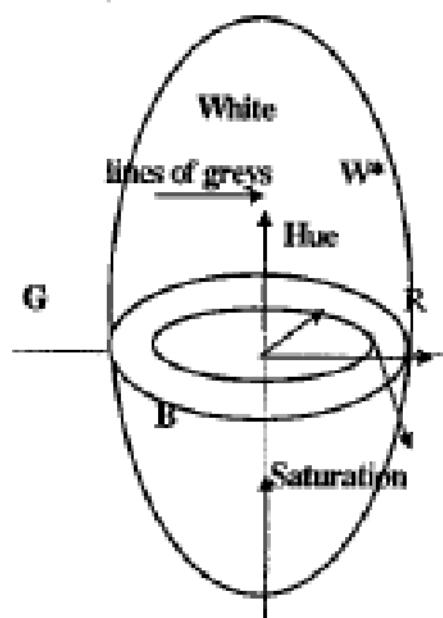
Ảnh sáng màu là tổ hợp của ánh sáng đơn sắc (monochrome). Mắt người chỉ có thể cảm nhận được vài chục màu, song lại có thể phân biệt được tới hàng ngàn màu. Có 3 thuộc tính chủ yếu trong cảm nhận màu:

- Brightness: sắc màu, còn gọi là độ chói.
- Hue : sắc lượng, còn gọi là sắc thái màu.
- Saturation: độ bão hòa.

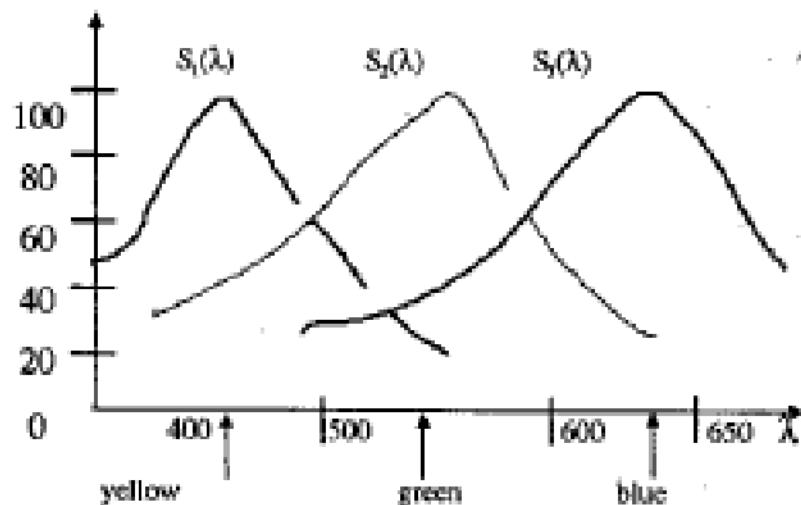
Với nguồn sáng đơn sắc, độ hue tương ứng với bước sóng  $\lambda$ . Độ bão hòa thay đổi nhanh nếu ta thêm lượng ánh sáng trắng. Hình 2.1 mô tả mối liên quan giữa các đại lượng trên và 3 màu chủ yếu R, G và B.

Với một điểm W\* cố định, các kí hiệu G, R, B chỉ vị trí tương đối của các phổ màu đỏ, lục và tím. Do sự tán sắc ánh sáng (ứng với khai triển Fourier) mà ta nhìn rõ màu. Theo Maxwell, trong võng mạc mắt có 3 loại tế bào hình nón cảm thụ 3 màu cơ bản ưng với 3 phổ hấp thụ  $S_1(\lambda)$ ,  $S_2(\lambda)$  và  $S_3(\lambda)$ .  $\lambda_{min} = 380$  nm;  $\lambda_{max} = 780$  nm.

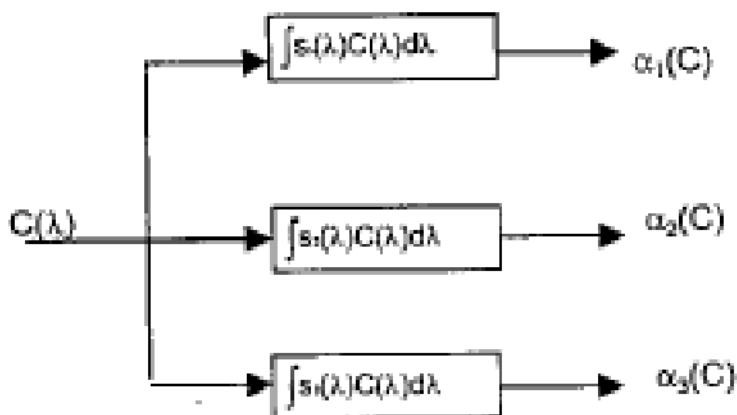
- Một màu bất kỳ sẽ là một điểm trên vòng tròn.
- Nếu màu đen và màu trắng là như nhau thì đường tròn là lớn nhất và R là điểm bao hoà.
- S thay đổi theo bán kính
- H thay đổi theo góc  $\theta$
- $W^*$  là sắc màu



Hình 2.1. Hệ toạ độ màu RGB.

Hình 2.2. Các đường cong cảm nhận  $S_1$ ,  $S_2$  và  $S_3$ .

Theo lý thuyết 3 màu, phân bố phổ năng lượng của một nguồn sáng màu ký hiệu là  $C(\lambda)$  và tổ hợp màu theo nguyên tắc 3 màu có thể mô tả bằng hình 2.3 dưới đây:



Hình 2.3. Nguyên tắc tổ hợp màu.

$$\text{Do đó, } \alpha_i(C) = \int_{\lambda_{min}}^{\lambda_{max}} S_i(\lambda) c(\lambda) d\lambda, \text{ với } i = 1, 2, 3. \quad (2.1)$$

$\alpha_i(C)$  gọi là đáp ứng phổ (spectral responses).

Phương trình 2.1 gọi là phương trình biểu diễn màu. Nếu  $C_1(\lambda)$  và  $C_2(\lambda)$  là hai phản bộ phổ năng lượng tạo nên các đáp ứng phổ  $\alpha_1(C_1)$  và  $\alpha_2(C_2)$  mà  $\alpha_1(C_1) = \alpha_2(C_2)$ , với  $i=1, 2, 3$  thì hai màu  $C_1$  và  $C_2$  là như nhau (sánh được).

### 2.1.3. Tổng hợp màu và sánh màu

Một trong các vấn đề cơ bản của lý thuyết biểu diễn màu là sử dụng một tập các nguồn sáng (màu) để biểu diễn màu. Theo lý thuyết 3 màu của Thomas, người ta hạn chế số màu còn 3 màu cơ bản: đỏ, lục và tím. Giả sử rằng ba nguồn sáng cơ bản có phản phổ năng lượng là  $p_k(\lambda)$  với  $k=1, 2, 3$  và:

$$\int_{\lambda_{min}}^{\lambda_{max}} p_k(\lambda) d\lambda = 1$$

Để sánh một màu  $C(\lambda)$ , giả sử rằng 3 màu cơ bản được tổ hợp theo tỉ lệ  $\beta_k(\lambda)$ ,  $k=1, 2, 3$ , như vậy:

$\sum_{k=1}^3 \beta_k(\lambda) p_k(\lambda)$  sẽ cho  $C(\lambda)$ . Thay giá trị này vào phương trình 2.1 ta có:

$$\begin{aligned} \alpha_i(C) &= \int_{\lambda_{min}}^{\lambda_{max}} S_i(\lambda) [\sum_{k=1}^3 \beta_k(\lambda) p_k(\lambda)] d\lambda = \sum_{k=1}^3 \beta_k(\lambda) \int_{\lambda_{min}}^{\lambda_{max}} p_k(\lambda) S_i(\lambda) d\lambda \\ &= \sum_{k=1}^3 \beta_k(\lambda) a_{ik} \quad \text{với } a_{ik} = \int_{\lambda_{min}}^{\lambda_{max}} p_k(\lambda) S_i(\lambda) d\lambda \end{aligned}$$

Như vậy, có thể tổng hợp màu theo phép cộng: màu  $X = \alpha_1$  đỏ +  $\alpha_2$  xanh +  $\alpha_3$  lơ với  $\alpha_1, \alpha_2$  và  $\alpha_3$  là các hệ số tổng hợp. Phương pháp này hay được dùng trong các ảnh dân dụng.

Lý thuyết tổng hợp màu trên cho phép đưa ra một số luật sánh màu sau:

i) mọi màu có thể sánh bởi nhiều nhất 3 màu.

ii) nguồn sáng của một màu tổng hợp bằng tổng nguồn sáng các màu thành phần.

iii) nếu màu  $C_1$  sánh được với màu  $C_1'$  và  $C_2$  sánh được với màu  $C_2'$  thì:

-  $\alpha_1 C_1 + \alpha_2 C_2 = \alpha_1 C_1' + \alpha_2 C_2'$  : luật cộng màu

- nếu  $C_1 + C_2 = C_1' + C_2'$  và  $C_2 = C_2'$  thì  $C_1 = C_1'$ .

iv) luật bắc cầu: nếu  $C_1 = C_2$  và  $C_2 = C_3$  thì  $C_1 = C_3$ , dấu = ở trên có nghĩa là sánh được.

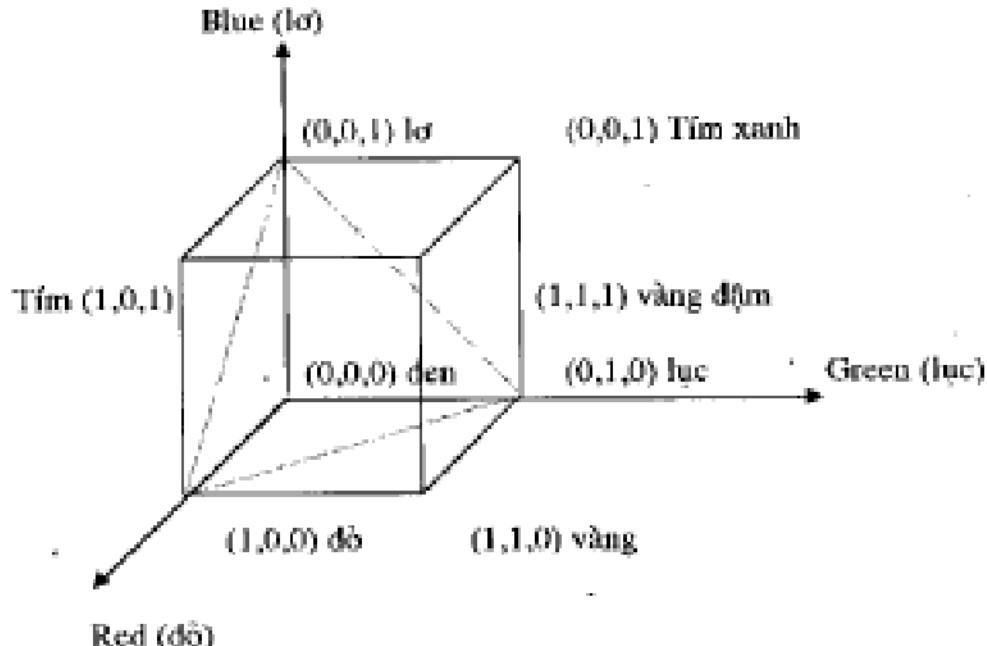
#### 2.1.4. Hệ toạ độ màu

##### a) Khái niệm

Tổ chức quốc tế về chuẩn hoá màu CIE (Commission Internationale d'Eclairage) đưa ra một số các chuẩn để biểu diễn màu. Các hệ này có các chuẩn riêng. Hệ chuẩn màu CIE-RGB dùng 3 màu cơ bản R, G, B và ký hiệu RGB<sub>CIE</sub> để phân biệt với các chuẩn khác. Như đã nêu trên, một màu là tổ hợp của các màu cơ bản theo một tỉ lệ nào đấy. Như vậy, một pixel ảnh màu kí hiệu P<sub>x</sub> được viết:

$$P_x = \begin{bmatrix} \text{red} \\ \text{green} \\ \text{blue} \end{bmatrix}$$

Người ta dùng hệ toạ độ ba màu R-G-B (tương ứng với hệ toạ độ x-y-z) để biểu diễn màu như sau:



Trong cách biểu diễn này ta có công thức: độ + lục + tía = 1. Công thức này gọi là công thức Maxwell. Trong hình vẽ trên, tam giác tạo bởi ba đường đứt đoạn gọi là tam giác Maxwell. Mùa trắng trong hệ toạ độ này được tính bởi: trắng<sub>CIE</sub> = (độ<sub>CIE</sub> = lục<sub>CIE</sub> = tía<sub>CIE</sub> = 1).

### b) Biến đổi hệ toạ độ màu

Hệ toạ độ màu do CIE đề xuất có tác dụng như một hệ qui chiếu và không biểu diễn hết các màu. Trên thực tế, phụ thuộc vào các ứng dụng khác nhau, người ta đưa ra các hệ biểu diễn màu khác nhau. Thí dụ:

- Hệ NTSC: dùng 3 màu R, G, B áp dụng cho màn hình màu, ký hiệu RGB<sub>NTSC</sub>
- Hệ CMY(Cyan Magenta Yellow): cho in ảnh màu.
- Hệ YIQ: cho truyền hình màu.

Việc chuyển đổi giữa các không gian biểu diễn màu được thực hiện theo nguyên tắc sau:

- nếu gọi  $\mathcal{X}$  là không gian biểu diễn màu ban đầu;
- $\mathcal{X}'$  là không gian biểu diễn màu mới;
- A: ma trận biểu diễn phép biến đổi.

Ta có:  $\mathcal{X}' = A\mathcal{X}$ . Ví dụ, biến đổi hệ toạ độ màu RGB<sub>CIE</sub> sang hệ toạ độ màu RGB<sub>NTSC</sub> ta có:

$$\mathbf{P}_x = \begin{bmatrix} R_{ce} \\ G_{ce} \\ B_{ce} \end{bmatrix} \quad \text{và} \quad \mathbf{P}_{x'} = \begin{bmatrix} R_{ntsc} \\ G_{ntsc} \\ B_{ntsc} \end{bmatrix}$$

Hay viết dưới dạng ma trận ta được:

$$\begin{bmatrix} R_{ce} \\ G_{ce} \\ B_{ce} \end{bmatrix} = \begin{bmatrix} 1.167 & -0.146 & -0.151 \\ 0.114 & 0.753 & 0.159 \\ -0.001 & 0.059 & 1.128 \end{bmatrix} \begin{bmatrix} R_{ntsc} \\ G_{ntsc} \\ B_{ntsc} \end{bmatrix}$$

Ma trận A cho các biến đổi khác được trình bày chi tiết trong [12].

## 2.2. LẤY MÀU VÀ LUÔNG TỬ HOÁ (IMAGE SAMPLING AND QUANTIZATION)

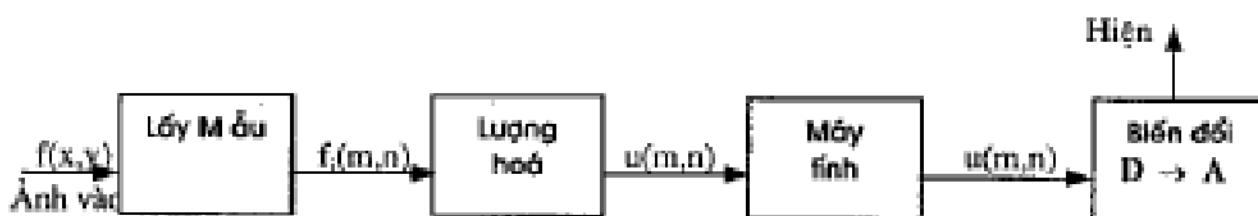
Yêu cầu cơ bản nhất trong xử lý ảnh bằng máy tính là đưa ảnh về dạng biểu diễn số

thích hợp, nghĩa là ảnh phải được biểu diễn bởi một ma trận hữu hạn tương ứng với việc lấy mẫu ảnh trên một lưới rời rạc và mỗi pixel được lượng hoá bởi một số hữu hạn bit. Ảnh số được lượng hoá có thể được xử lý hay chuyển qua bước biến đổi số tương tự — DA (Digital to Analog) để tái hiện trên thiết bị hiện ảnh hoặc được lưu trữ lại.

### 2.2.1. Lấy mẫu (Image sampling)

Phương pháp chung để lấy mẫu là quét ảnh theo hàng và mã hoá từng hàng. Về nguyên tắc, một đối tượng, phim hay giấy trong suốt sẽ được chiếu sáng liên tục để tạo nên một ảnh điện tử trên cảm quang. Tuỳ theo các loại camera mà cảm quang này là chất quang dẫn hay quang truyền. Hệ thống camera ống sử dụng phương pháp *scan-out-digitalizer*; còn hệ thống camera CCD (Charge Coupled Device) cho ảnh ma trận.

Camera CCD thực sự là thiết bị mẫu hoá tín hiệu 2 chiều và gọi là phương pháp *self-scanning matrix*. Nguyên tắc của 2 phương pháp được minh họa qua hình 2-6.



$$\begin{aligned} & -x, y \in (-\infty, +\infty) & -m \in [1, M], n \in [1, N] & -u(m,n) \in [0, L] \\ & -f(x,y) \in (-\infty, +\infty) & -f(m, n) \in (-\infty, +\infty) \end{aligned}$$

Hình 2.4. Lấy mẫu và lượng hoá

- Lý thuyết mẫu hoá 2 chiều

- Ảnh với dài giới hạn (Band limited Images)

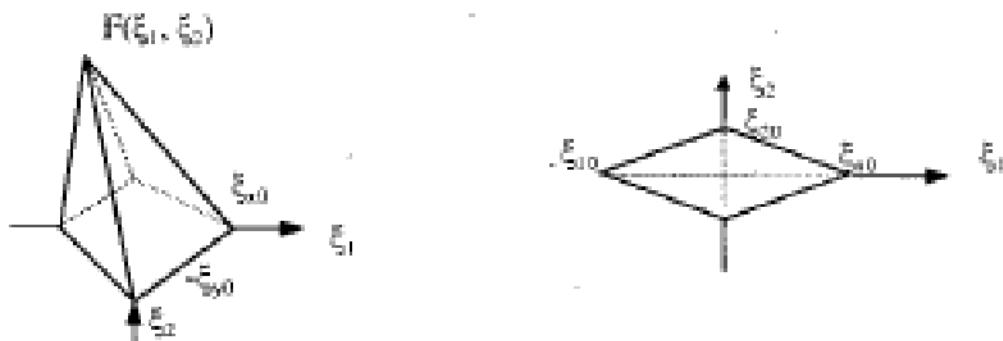
Một hàm  $f(x,y)$  gọi là dài giới hạn nếu khai triển Fourier  $F(\xi_1, \xi_2)$  của nó là 0 bên ngoài miền bao (hình 2.5).  $F(\xi_1, \xi_2) = 0$  với  $|\xi_1| > \xi_{x0}, |\xi_2| > \xi_{y0}$  (2.2)

Với  $\xi_{x0}$  và  $\xi_{y0}$  là dài giới hạn theo x và y của ảnh.

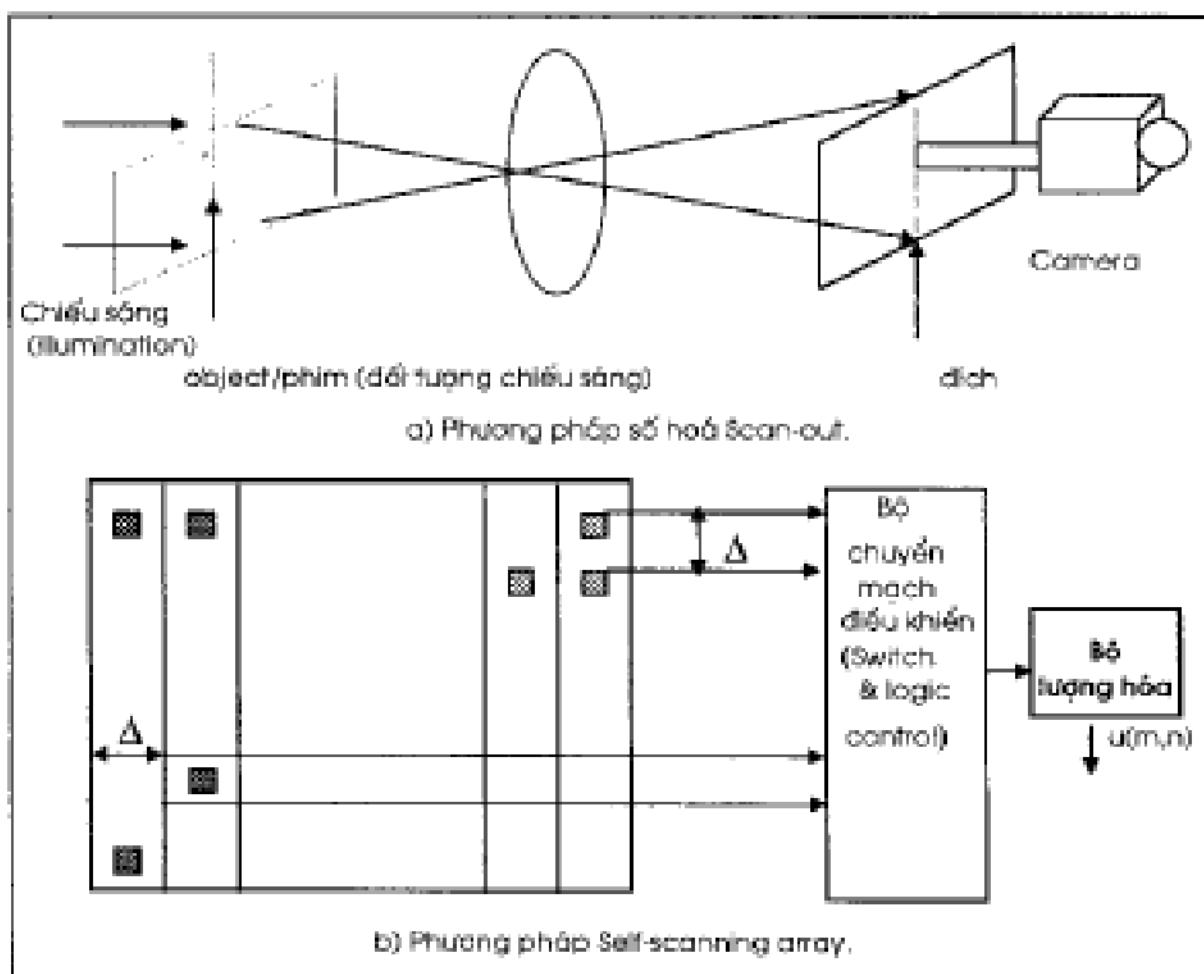
Quá trình số hoá ảnh có thể hiểu như mô hình tín hiệu dài giới hạn. Một ảnh dài giới hạn  $f(x,y)$  thoả mãn phương trình 2.2 và được lấy mẫu đều trên một lưới hình chữ nhật với bước nhảy  $\Delta x, \Delta y$  có thể khôi phục lại không có sai sót dựa trên các giá trị mẫu  $f(m\Delta x, n\Delta y)$ . Theo lý thuyết lấy mẫu trong xử lý tín hiệu, nếu tần số lấy mẫu theo x, y lớn hơn 2 lần dài giới hạn  $\xi_{x0}, \xi_{y0}$  hay tương đương với:

$$\frac{1}{\Delta x} = \xi_{x0} > 2 \xi_{x0}, \quad \frac{1}{\Delta y} = \xi_{y0} > 2 \xi_{y0}$$

thì có thể khôi phục được. Tỉ số này do Nyquist đề xuất và mang tên tỉ số Nyquist.



Hình 2.5. Khai triển Fourier của hàm dải giới hạn.



Hình 2.6. Phương pháp lấy mẫu & lượng hoá ảnh.

Trong xử lý ảnh, tần số lấy mẫu theo trục x và trục y thường được chọn là 8MHz (tần số max theo x và y nằm trong khoảng 4 ~ 6 MHz). Do vậy chu kỳ lấy mẫu theo cả 2 trục là:

$$\frac{1}{8\text{MHz}} = 0.125 \cdot 10^{-6} \text{ s}$$

Ngoài ra cần đảm bảo thời gian quét một mặt ảnh là 1/30 s. Việc khôi phục ảnh có thể nội suy theo công thức:

$$f(x,y) = \sum_{m,n=-\infty}^{\infty} f(mx, ny) \left( \frac{\sin(x\xi_m - x)\pi}{(x\xi_m - m)\pi} \right) \left( \frac{\sin(y\xi_n - n)\pi}{(y\xi_n - n)\pi} \right) \quad (2.3)$$

Ảnh sau khi lấy mẫu là ảnh rời rạc và thường được ký hiệu  $f(m,n)$ , với m, n là các giá trị nguyên. Những giá trị hay dùng của m và n là 256, 512, 1024. Tuy nhiên, giá trị của  $f(m,n)$  vẫn là liên tục và trong khoảng  $(-\infty, +\infty)$ .

Trong thực tế, nhiễu ngẫu nhiên luôn có mặt trong tín hiệu ảnh. Do đó, lý thuyết lấy mẫu ở trên phải được mở rộng với một số kỹ thuật khác như: lưới không vuông, lưới bất giác. Để đơn giản khi trình bày, những kỹ thuật này không nêu ở đây. Độc giả có quan tâm xin tham khảo tài liệu [1].

## 2.2.2. Lượng hoá ảnh (Image Quantization)

### 2.2.2.1. Khái niệm và nguyên tắc lượng hoá ảnh

Lượng hoá ảnh là bước kế tiếp của việc lấy mẫu, nhằm thực hiện một ảnh xạ từ một biến liên tục u (biểu diễn giá trị độ sáng) sang một biến rời rạc  $u^*$  với các giá trị thuộc tập hữu hạn  $\{t_1, t_2, \dots, t_L\}$ . Ảnh xạ này thường là một hàm bậc thang (hình 2.7) tuân theo nguyên tắc sau:

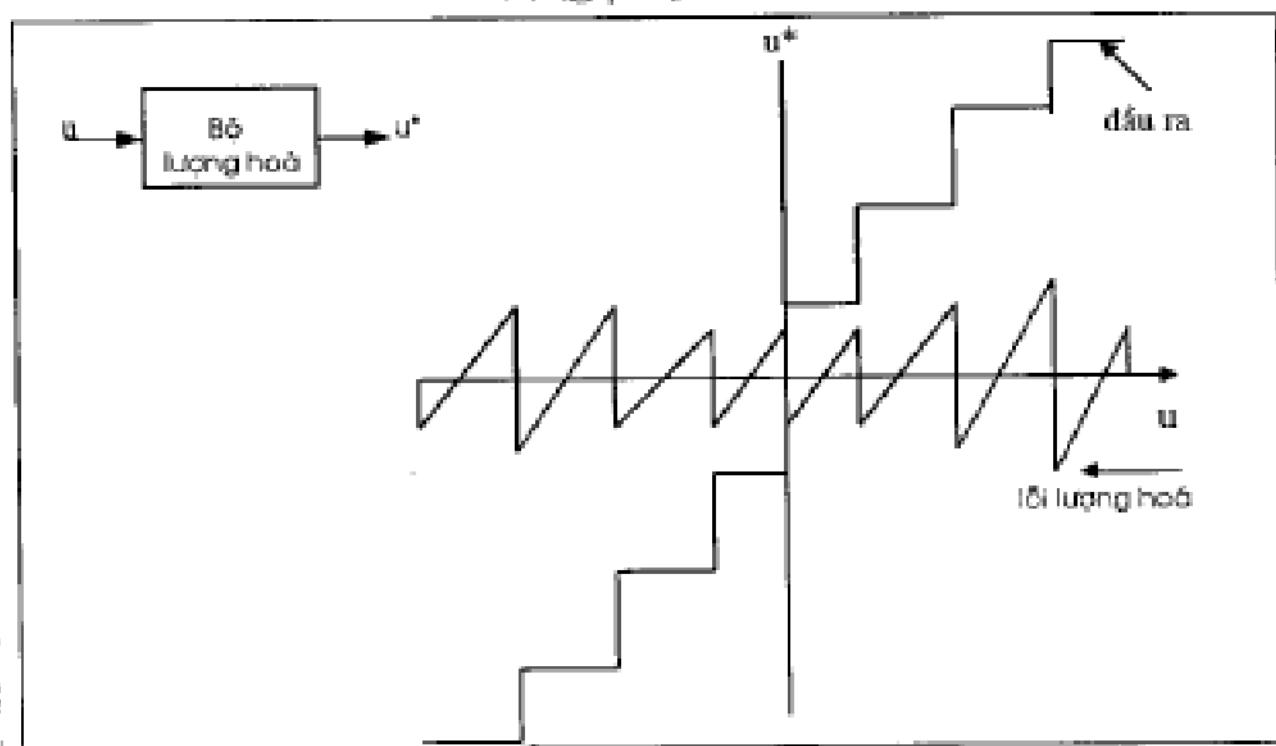
Cho  $\{t_k, k=1, 2, \dots, L+1\}$  là một tập các bước dịch chuyển hay mức độ quyết định;  $t_1$  là giá trị nhỏ nhất và  $t_{L+1}$  là giá trị lớn nhất của u.

Cơ sở lý thuyết của lượng hoá là dải độ sáng biến thiên hữu hạn từ  $L_{min}$  đến  $L_{max}$ . Người ta chia dải biến thiên đó thành một số mức (rời rạc và nguyên). Việc chia này phải thoả tiêu chí về độ nhạy của mắt: mắt người không thể phân biệt được các giá trị trong một mức; mắt người chỉ phân biệt được hai mức kề nhau.  $L_{min}$  thường là 0, còn  $L_{max}$  là một số nguyên dạng  $2^n$  và thường chọn bằng 256 (256 mức khác nhau).

Với các khoảng chia như trên  $(t_i, t_{i+1}, \dots, t_L)$ , nếu  $u \in (t_i, t_{i+1})$  thì gán cho u giá trị i, hay nói cách khác u đã được *lượng hoá bởi mức i* (giá trị i).

Cách đơn giản nhất để lượng hoá là dùng lượng hoá đều. Theo phương pháp này, giá

sử đầu ra của một bộ cảm biến ảnh nhận giá trị từ 0 đến 10.0. Nếu mẫu là lượng hoá đều trên 256 mức, thì bước dịch chuyển  $t_k$  và mức xây dựng lại  $r_k$  được tính bởi:



Hình 2.7. Mô hình bộ lượng hóa.

$$t_k = \frac{10(k-1)}{256} \quad \text{với } k=1, 2, \dots, 257; \quad r_k = t_k - \frac{5}{256} \quad \text{với } k=1, 2, \dots, 256$$

Đại lượng  $q = t_k - t_{k-1} = r_k - r_{k-1}$  là hằng số với các giá trị  $k$  và gọi là khoảng lượng hoá.

Trong phần này, ta chỉ xem xét các bộ lượng hoá không bộ nhớ (zero memory quantizer), có nghĩa là đầu ra chỉ phụ thuộc duy nhất là đầu vào. Các bộ lượng hoá kiểu này rất có ích trong kỹ thuật mã hoá ảnh như mã hoá điều xung PCM (Pulse Code Modulation), PCM vi phân, chuyển mã, v.v. Chú ý rằng, ánh xạ lượng hoá này không thuận nghịch, nghĩa là với một đầu ra đã cho, đầu vào là không duy nhất. Vì vậy, người ta đã nghiên cứu bổ sung nhiều kỹ thuật khác nhau để cực tiểu hoá biến dạng, tăng hiệu quả. Một kỹ thuật phổ dụng là *trung bình bình phương cực tiểu* (do Lloyd-max đề xuất) chúng ta sẽ mô tả dưới đây.

#### 2.2.2.2. Kỹ thuật lượng hoá trung bình bình phương cực tiểu

Kỹ thuật này nhằm cực tiểu hoá sai số trung bình bình phương đối với một số mức

lượng hoá đã cho. Cho  $u$  là một biến thực ngẫu nhiên với hàm mật độ liên tục  $P_u(u)$ . Mong muốn ở đây là tìm được mức độ quyết định  $t_k$  và mức khôi phục lại  $r_k$  với một bộ lượng hoá  $L$  mức sao cho sai số trung bình bình phương là nhỏ nhất.

$$\text{Gọi } \varepsilon = E[(u - u^*)^2] = \int_{t_0}^{t_{L+1}} (u - u^*)^2 P_u(u) du \quad (2.4)$$

Nhiệm vụ là tìm min của  $\varepsilon$ .

Viết lại (2.4) ta có:

$$\varepsilon = \sum_{i=1}^L \frac{t_{i+1} - t_i}{t_1} \int_{t_i}^{t_{i+1}} (u - r_k)^2 P_u(u) du \quad i=0, 1, \dots, L-1 \quad (2.5)$$

Để tính  $r_k$ , ta cần giải hệ phương trình (nhận được khi lấy vi phân 2.5):

$$\begin{cases} (t_k - r_{k-1})^2 P_u(t_k) - (t_k - r_k)^2 P_u(t_k) = 0 \\ 2 \int_{t_k}^{t_{k+1}} (u - r_k) P_u(u) du = 0 \end{cases}$$

Lưu ý rằng  $t_k \geq t_{k-1}$ , do đó giá trị của  $t_k$  và  $r_k$  cho bởi:

$$t_k = (r_k + r_{k-1})/2 \quad k = 1, 2, \dots, L \quad (2.6)$$

$$\text{và} \quad r_k = \frac{\int_{t_k}^{t_{k+1}} u p_u(u) du}{\int_{t_k}^{t_{k+1}} p_u(u) du} \quad k = 0, 1, \dots, L-1 \quad (2.7)$$

Thông thường hệ phương trình (2.6), (2.7) là không tuyến tính.

Kết quả trên chứng tỏ rằng mức dịch chuyển tối ưu nằm trên nửa đường của các mức khôi phục lại. Các mức khôi phục lại tối ưu nằm tại trọng tâm của phân bố mật độ giữa các mức dịch chuyển.

Giải hệ phương trình (2.6) và (2.7) ta thu được các cận  $t_0$  và  $t_{L+1}$ . Trong thực tế, người ta hay áp dụng phương pháp Newton để giải phương trình trên. Khi số mức lượng hoá lớn, người ta dùng phương pháp xấp xỉ mật độ xác suất như một hàm hàng *khôn ngoan* (piecewise)  $p_u(u) = p_u(v_i)$  với  $v_i = (t_i + t_{i+1})/2$ ;  $t_i \leq u < t_{i+1}$ . Thay giá trị mới của  $p_u(u)$  vào 2.5 và tính cực tiểu hoá, ta có lời giải xấp xỉ cho mức quyết định  $t_{i+1}[1]$ :

$$t_{k+1} = \frac{A \int_{t_k}^{t_{k+1}} |p_v(u)|^{1/3} du}{\int_{t_k}^{t_{L+1}} |p_v(u)|^{1/3} du} - t_k \quad (2.8)$$

với  $A = t_{L+1} - t_1$  và  $t_k = (k/L)A$ ,  $k=1,2,\dots,L$ . Từ đó ta dễ dàng tính được giá trị của sai số  $\epsilon$ . (\*)

Các hàm mật độ thường dùng là hàm Gauss và hàm Laplace.

Hàm Gauss có dạng:

$$P_v(u) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) \quad (2.9)$$

$$\text{Hàm Laplace có dạng: } P_v(u) = \alpha/2 * \exp(-\alpha|u-\mu|) \quad (2.10)$$

trong đó:

- $\mu$  là kỳ vọng toán học.
- $\sigma^2$  là hiệp biến với biến ngẫu nhiên  $u$  đối với hàm Gauss.

Hiệp biến Laplace được tính bởi  $\sigma^2 = 2/\alpha$ .

Trường hợp đặc biệt, nếu phân bố là đều thì hệ phương trình (2.6) và (2.7) là tuyến tính và sẽ cho ta các khoảng đều nhau giữa các mức dịch chuyển và mức khôi phục lại. Do vậy, phép lượng hoá này có tên là lượng hoá tuyến tính.

Giả sử hàm mật độ cho bởi công thức:

$$p_v(u) = \begin{cases} 1/(t_{L+1} - t_1) & \text{nếu } t_1 \leq u \leq t_{L+1} \\ 0 & \text{kết khác} \end{cases}$$

Từ phương trình (2.7) ta có:

$$r_k = \frac{t_{2k+1} - t_{2k}}{2(t_{k+1} - t_k)} = \frac{t_{k+1} - t_k}{2} \quad (2.11)$$

do đó  $t_k = (t_{k+1} - t_k)/2 \Rightarrow t_k = t_{k+1} = t_{k+1} - t_k = \text{const} = q$ .

$$\text{Cuối cùng ta có } q = (t_{L+1} - t_1)/L; t_k = t_{k+1} + q; r_k = t_k - q/2 \quad (2.12)$$

Như vậy, mọi mức dịch chuyển và mức khôi phục lại đều cách đều. Sai số của phép lượng hoá là  $u - u^*$  sẽ phân phối đều trên khoảng  $(-q/2, q/2)$ . Sai số trung bình bình phương sẽ là:

$$e = \frac{1}{q} \int_{-q/2}^{q/2} u^2 du = \frac{q^2}{12} \quad (2.13)^*$$

Lượng hoá đều như trên khá thuận tiện cho cài đặt. Tuy nhiên, trong thực tế ta còn gặp nhiều loại phân bố không đều của các biến ngẫu nhiên. Đặc biệt quan tâm đến các biến đổi này cũng như so sánh kết quả giữa một số phương pháp xin tham khảo tài liệu [1].

### 2.3. MỘT SỐ PHƯƠNG PHÁP BIỂU DIỄN ẢNH (IMAGE REPRESENTATION)

Sau bước số hoá, ảnh sẽ được lưu trữ hay chuyển sang giai đoạn phân tích. Trước khi đề cập đến vấn đề lưu trữ ảnh, ta cần xem xét ảnh sẽ được biểu diễn ra sao trong bộ nhớ máy tính. Phản trên cũng đã nói đến các mô hình toán học để biểu diễn ảnh. Nếu lưu trữ trực tiếp ảnh theo kiểu bản đồ ảnh, dung lượng sẽ khá lớn, tốn kém mà nhiều khi không hiệu quả theo quan điểm ứng dụng. Thường người ta không biểu diễn toàn bộ ảnh mà tập trung đặc tả các đặc trưng của ảnh như: biên ảnh (Boundary) hay các vùng ảnh (Region). Các kỹ thuật phát hiện biên hay phân vùng ảnh sẽ được giới thiệu kỹ trong chương 5 và 6. Dưới đây giới thiệu một số phương pháp biểu diễn. Thường người ta dùng:

- Biểu diễn mã loạt dài (Run - Length Code).
- Biểu diễn mã xích (Chain Code).
- Biểu diễn mã tứ phân (Quad Tree Code).

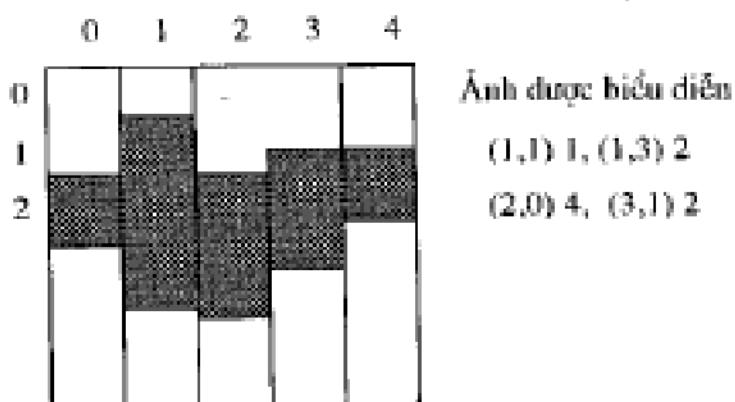
Ngoài ra cũng dùng mô hình thống kê.

#### 2.3.1. Mã loạt dài

Phương pháp này hay dùng biểu diễn cho vùng ảnh hay ảnh nhị phân. Một vùng ảnh có thể biểu diễn đơn giản nhờ một ma trận nhị phân:

$$u(m,n) = \begin{cases} 1 & \text{nếu } (m,n) \in \Omega \\ 0 & \text{nếu không} \end{cases}$$

Với cách biểu diễn trên, một vùng ảnh hay ảnh nhị phân được xem như gồm các chuỗi 0 hay 1 đan xen. Các chuỗi này gọi là một mạch (run). Theo phương pháp này, mỗi mạch sẽ được biểu diễn bởi địa chỉ bắt đầu của mạch và chiều dài mạch theo dạng: (<hàng,cột>, chiều dài).



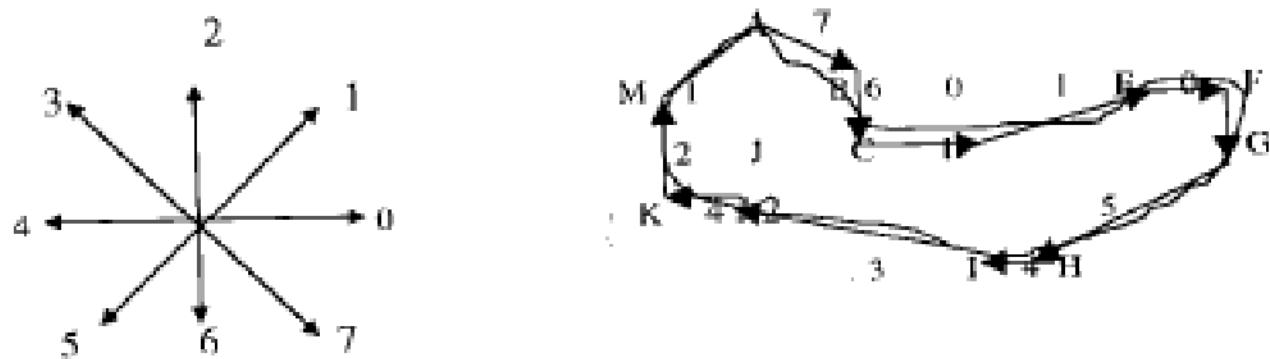
Hình 2.8. Ảnh nhị phân và các biểu diễn mã lot dài tương ứng.

Nhiều dạng biến thể khác nhau của phương pháp này sẽ được chi tiết trong chương 8 (8.2.1).

### 2.3.2. Mã xích

Mã xích thường được dùng để biểu diễn biên của ảnh. Thay vì lưu trữ toàn bộ ảnh, người ta lưu trữ dãy các điểm ảnh như A, B, ..., M (hình 2.9). Theo phương pháp này, 8 hướng của véc-tơ nối 2 điểm biên liên tục được mã hóa. Khi đó ảnh được biểu diễn qua điểm ảnh bắt đầu A cùng với chuỗi các từ mã. Điều này được minh họa trong hình 2.9 dưới đây.

Một biến thể của phương pháp này là tăng số hướng. Với biên cũng còn nhiều phương pháp khác (Chương 5).



Hình 2.9. Hướng các điểm biên và mã tương ứng.

A 111 110 000 001 000 110 101 110 101 010 100 010

### 2.3.3. Mã tử phân

Theo phương pháp mã tử phân, một vùng của ảnh coi như bao kín bởi một hình chữ nhật. Vùng này được chia làm 4 vùng con (quadrant). Nếu một vùng con gồm toàn điểm đen (1) hay toàn điểm trắng(0) thì không cần chia tiếp. Trong trường hợp ngược lại, vùng con gồm cả đen và trắng gọi là vùng xám lại tiếp tục được chia làm 4 vùng con tiếp. Quá trình chia dừng lại khi không thể chia tiếp được nữa, có nghĩa là vùng con chỉ chứa thuần nhất điểm đen hay trắng. Như vậy, cây biểu diễn gồm một chuỗi các ký hiệu b(black), w (white) và g(grey) kèm theo ký hiệu mã hoá 4 vùng con. Biểu diễn theo phương pháp này ưu việt hơn so với các phương pháp trên, nhất là so với mã loạt dài. Tuy nhiên, để tính toán số do các hình như chu vi, mô men là khá khó.

## 2.4. CÁC ĐỊNH DẠNG ẢNH CƠ BẢN TRONG XỬ LÝ ẢNH

### 2.4.1. Khái niệm chung

Ảnh thu được sau quá trình số hoá thường được lưu lại cho các quá trình xử lý tiếp theo hay truyền đi. Trong quá trình phát triển của kỹ thuật xử lý ảnh, tồn tại nhiều định dạng khác nhau từ ảnh đen trắng IMG, ảnh đa cấp xám/ảnh màu: BMP, GIF, JPEG, ... Tuy các định dạng này là khá khác nhau, song chúng cũng tuân theo một cấu trúc chung nhất. Để trong sáng cách trình bày, định dạng chi tiết của các file ảnh sẽ được mô tả trong phần phụ lục. Dưới đây chỉ trình bày cấu trúc chung và ý nghĩa của các phần cấu trúc đó.

Nhìn chung, một tệp ảnh gồm 3 phần:

- Đầu tệp (Header)
- Dữ liệu nén (Data Compression)
- Bảng màu (Palette Color)

a) *Đầu tệp*: Chứa các thông tin về kiểu ảnh, kích thước, độ phân giải, số bit dùng cho 1 pixel, cách mã hoá, vị trí bảng màu,... Kích thước phần header rất khác nhau, phụ thuộc kiểu định dạng ảnh.

b) *Dữ liệu nén*: Số liệu ảnh đã được mã hoá bởi kiểu mã hoá chỉ ra trong phần Header.

c) *Bảng màu*: Không nhất thiết phải có. Nếu có, nó chỉ ra số màu dùng trong ảnh và sử dụng để hiện ảnh.

<i>Đầu tệp (Header)</i>
<i>Dữ liệu nén (Data Compression)</i>
<i>Bảng màu (Palette Color)</i>

### 2.4.2. Qui trình đọc 1 tệp ảnh

Để xử lý được ảnh, ta phải tiến hành đọc tệp ảnh và chuyển vào bộ nhớ của máy tính dưới dạng ma trận số liệu ảnh. Khi lưu trữ dưới dạng tệp, ảnh là một khối các byte. Để đọc đúng, ta cần hiểu ý nghĩa các phần trong cấu trúc của tệp ảnh như đã nêu trên. Trước tiên, ta cần đọc Header để lấy các thông tin điều khiển. Việc đọc này sẽ dừng ngay khi ta không gặp được chữ ký mong muốn (thường là 6 bytes đầu). Dựa vào thông tin điều khiển này, ta xác định được vị trí bảng màu (nếu có) và đọc nó vào bộ nhớ. Cuối cùng, ta đọc phần dữ liệu nền.

Sau khi đã đọc xong các khối vào bộ nhớ, ta tiến hành giải nén số liệu ảnh. Cần có vào phương pháp nén chỉ ra trong phần Header (có thể có các bảng phụ) ta giải mã được ảnh. Cuối cùng là khâu hiện ảnh. Dựa vào số liệu ảnh đã giải nén, vị trí và kích thước ảnh, cùng sự trợ giúp của bảng màu ảnh được hiện lên màn hình.

## 2.5. CÁC KỸ THUẬT TÁI HIỆN ẢNH

### 2.5.1. Kỹ thuật chụp ảnh (photography hardcopy techniques)

Phương pháp sao chụp ảnh là phương pháp đơn giản giá thành thấp, chất lượng cao. Sau bước chụp là kỹ thuật phòng tối (darkroom) nhằm tăng cường ảnh như mong muốn. Thi dụ như : phóng đại ảnh, thu nhỏ ảnh,..., tùy theo ứng dụng. Kỹ thuật chụp ảnh màn hình màu là khá đơn giản. Nó gồm một số bước như sau:

- 1) Đặt camera trong phòng tối, cách màn hình khoảng 10 feet (1 feet = 0,3048 m).
- 2) Mở ống kính để làm phẳng mặt cong màn hình do vậy ảnh sẽ dàn đều hơn.
- 3) Tắt phím phản chiếu (brightness) và phím tương phản (contrast) của màn hình để tạo nên độ rõ cho ảnh. Các màu chói, cường độ cao trên ảnh sẽ giảm đi.
- 4) Đặt tốc độ ống kính từ 1/8 đến 1/2 giây.

Với ống kính tốc độ 1/4 giây, bắt đầu với f và dừng ở f/8.

### 2.5.2. Kỹ thuật in ảnh (Printer Hardcopy techniques)

Trước khi đi sâu vào kỹ thuật in ảnh, ta xem xét các ảnh được thể hiện trên sách, báo ảnh và tạp chí như thế nào. Nhìn chung, người ta dùng kỹ thuật nửa cường độ (*halftone*). Theo kỹ thuật này, một ảnh tạo nên bởi một chuỗi các điểm in trên giấy. Thực chất, mỗi pixel gồm một hình vuông trắng bao quanh một chấm đen (black dot). Do vậy, nếu chấm đen càng lớn ảnh sẽ càng xám màu. Màu xám có thể coi như chấm đen chiếm

nửa vùng trắng. Vùng trắng là vùng gồm một chùm các pixel gồm rất ít hoặc không có chấm đen.

Do sự cảm nhận của mắt người, sự thay đổi cường độ chấm đen trong các phần tử ảnh trắng tạo nên mô phỏng của một ảnh liên tục. Như vậy, mắt người cảm nhận là một ảnh mà màu biến đổi từ đen qua xám rồi đến trắng. Tổng số cường độ duy nhất hiện diện sẽ xác định các kích thước khác nhau của chấm đen. Thường báo ảnh tạo ảnh nửa cường độ với độ phân giải từ 60 đến 80 dpi, sách có thể in đến 150 dpi.

Tuy nhiên, các máy in của máy tính không có khả năng sắp xếp các chấm đen của các kích thước khác nhau của ảnh. Do đó, người ta phải dùng một số kỹ thuật biến đổi như: phân ngưỡng, chọn mẫu, dithering. Chúng ta lần lượt xem xét dưới đây.

#### • Phân ngưỡng (Thresholding)

Kỹ thuật này đặt ngưỡng để hiển thị các tông màu liên tục. Các điểm trong ảnh được so sánh với ngưỡng định trước. Giá trị của ngưỡng sẽ quyết định điểm có được hiển thị hay không. Do vậy ảnh kết quả sẽ mất đi một số chi tiết. Có nhiều kỹ thuật chọn ngưỡng áp dụng cho các đối tượng khác nhau:

- Hiển thị 2 màu: dùng cho ảnh đen trắng có 256 mức xám. Bản chất của phương pháp này là chọn ngưỡng dựa trên lược đồ mức xám của ảnh; để đơn giản có thể lấy ngưỡng với giá trị là 127. Và như vậy:

$$u(m,n) = \begin{cases} 1 \text{ (cho hiển thị)} & \text{nếu } u(m,n) < 127 \\ 0 \text{ (hay hiển thị trắng)} & \text{nếu ngược lại} \end{cases}$$

Nhìn chung kỹ thuật này khó chấp nhận vì ảnh mất khá nhiều chi tiết.

- Kỹ thuật hiện 4 màu: để khắc phục nhược điểm của cách hiện 2 màu, người ta qui định cách hiện 4 màu như sau:

Màu	Màn hình monochrome (đơn sắc)	Màn hình màu
0	đen	đen
1	xám đậm	đỏ
2	xám nhạt	xanh
3	trắng	vàng

Ta có thể hình dung cách phân ngưỡng 4 màu theo sơ đồ sau:



- Dùng ngưỡng ngẫu nhiên 4 màu: theo phương pháp này, ta chia không gian mức xám thành các miền hiển thị và đánh số là 0, 1, 2 và 3. Tiếp sau, định nghĩa các miền mà các cặp hiển thị (bật, tắt) tương ứng với: (0,1), (1,2) và (2,3). Khác với ngưỡng cố định, ở đây ngưỡng được reo ngẫu nhiên. Quá trình thực hiện được mô tả trong thuật toán :

for each pixel  $I(x,y)$  do

Begin

if  $I(x,y) < 85$  then

Begin

. Khởi tạo 1 số ngẫu nhiên  $r$  trong khoảng [0,84]

. if  $I(x,y) > r$  then màu =1 else màu =0

End

else if  $I(x,y) < 170$  then

Begin

. Khởi tạo 1 số ngẫu nhiên  $r$  trong khoảng [85,169]

. if  $I(x,y) > r$  then màu =1 else màu =0

End

Else

Begin

. Khởi tạo một số ngẫu nhiên  $r$  trong khoảng [170,255]

. if  $I(x,y) > r$  then màu =1 else màu =0

End

end

#### • Kỹ thuật chọn theo màu (patterning)

Kỹ thuật này sử dụng một nhóm các phần tử trên thiết bị ra (máy in chẳng hạn) để biểu diễn một pixel trên ảnh nguồn. Các phần tử của nhóm quyết định độ sáng tối của cả

nhóm. Các phần tử này mô phỏng các chấm đen trong kỹ thuật nửa cường độ. Nhóm thường được chọn có dạng ma trận vuông. Nhóm  $n \times n$  phần tử sẽ tạo nên  $n^2 + 1$  mức sáng. Ma trận mẫu thường được chọn là ma trận Rylander. Ma trận Rylander cấp 4 có dạng:

0	8	2	10
4	12	6	14
3	11	1	9
7	15	5	13

Việc chọn kích thước của nhóm như vậy sẽ làm giảm đi độ mịn của ảnh. Vì vậy kỹ thuật này chỉ áp dụng trong trường hợp mà độ phân giải của thiết bị ra lớn hơn độ phân giải của ảnh nguồn. Thí dụ: thiết bị ra có độ phân giải 640 x 480 khi sử dụng nhóm có kích thước 4 x 4 sẽ chỉ còn 160 x 120.

#### \* Kỹ thuật Dithering

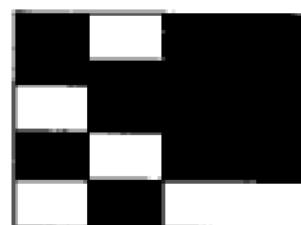
Kỹ thuật Dithering được áp dụng để tạo ra ảnh đa cấp sáng khi độ phân giải nguồn và đích là như nhau. Kỹ thuật này sử dụng một ma trận mẫu gọi là ma trận Dither. Ma trận này gần giống như ma trận Rylander.

Để tạo ảnh, mỗi phần tử của ảnh gốc sẽ được so sánh với phần tử tương ứng của ma trận Dither. Nếu lớn hơn, phần tử ở đầu ra sẽ sáng và ngược lại. Ma trận Dither cấp  $2n$  sẽ được tính như sau:

8	7	8	15
6	7	15	13
7	5	13	12
15	13	12	12

a) ảnh gốc

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

b) ma trận Dither -  $D^4$ 

c) ảnh kết quả

Hình 2.11. Tạo ảnh theo phương pháp Dithering

$$D^{2n} = \begin{bmatrix} 4D^n + D_{1n}^2U^n & 4D^n + D_{1n}^2U^n \\ 4D^n + D_{2n}^2U^n & 4D^n + D_{2n}^2U^n \end{bmatrix} \text{ với } D^n = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} D_{1n}^2 & D_{1n}^2 \\ D_{2n}^2 & D_{2n}^2 \end{bmatrix}$$

$D^n$  là ma trận Dither cấp  $n$ ;  $U^n$  là ma trận cấp  $n$  (các phần tử đều là 1) dạng:

$$\begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}$$

Thí dụ, với  $D^2$  như trên, ta tính  $D^4$  như sau:

$$D^4 = \begin{bmatrix} 4D^2 + D_{00}^2 U^2 & 4D^2 + D_{01}^2 U^2 \\ 4D^2 + D_{10}^2 & 4D^2 + D_{11}^2 U^2 \end{bmatrix} \text{ và } D^4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

Một cách tương tự, ta tính được  $D^{16}$ . Với  $D^{16}$ , ta thấy tất cả các giá trị từ 0 đến 255 đều có mặt. Khác với phương pháp ngẫu nhiên chỉ dựa vào một ngưỡng biến thiên, ở đây ngưỡng được xác định một cách rõ ràng. Cách dùng ma trận ngưỡng có thể hình dung như sau:

0	128	32	160	8	136	40	168	2	130	34	162	10	138	42	170
192	64	224	96	200	72	232	104	194	66	226	98	202	74	234	106
48	176	16	144	56	184	24	152	50	178	18	146	58	186	26	154
240	112	208	80	248	120	216	88	242	114	210	82	250	122	218	90
12	140	44	172	4	132	36	164	14	142	46	174	6	134	38	166
204	76	236	108	196	68	228	100	206	78	238	110	198	70	230	102
60	188	28	156	52	180	20	148	62	192	30	158	54	182	22	150
252	124	220	92	244	116	212	84	254	126	222	94	246	118	214	86
3	131	35	163	11	139	43	171	1	129	33	161	9	137	41	169
195	67	227	99	203	75	235	107	193	65	225	97	201	73	233	105
51	179	19	147	59	187	27	155	49	177	17	145	57	185	25	153
243	115	211	83	251	123	219	91	241	113	209	81	249	121	217	89
15	143	47	175	7	135	39	167	13	141	45	173	5	133	37	165
207	79	239	111	199	71	231	103	205	77	237	109	197	69	229	101
63	191	31	159	55	183	23	151	61	189	29	157	53	181	21	149
255	127	223	95	247	119	215	87	253	125	221	93	245	117	213	85

Ma trận Dither cấp 16 -  $D^{16}$

Giả sử  $I(x,y)$  là một điểm ảnh. Đặt  $x_0 = x \bmod 16$  và  $y_0 = y \bmod 16$  thì  $x_0, y_0$  sẽ chỉ nhận một trong các giá trị từ 0 đến 15. Như vậy, nó sẽ xác định một phần tử của  $D^{16}$ . Gọi  $S = D^{16}(x_0, y_0)$ .  $S$  sẽ có giá trị trong khoảng từ 0 đến 255 và có thể dùng làm ngưỡng để hiện ảnh. Hơn nữa, nếu chỉ dịch chuyển theo một chiều ( $x$  chẳng hạn), ta thấy rằng cả 16 điểm ảnh sẽ rơi vào ngưỡng  $S$  thu được từ 16 điểm trước. Do vậy, ta có một cách thức chuẩn để bật hay tắt các điểm ảnh với trạng thái lưới.

Để sử dụng được 4 màu, ta cũng sử dụng theo kỹ thuật nguồng 4 màu nhưng chỉ cần 3 vùng, mỗi vùng 85 mức. Ở đây cần có sự lựa chọn giữa  $D^{16}$  và  $D^8$ . Nếu chọn  $D^{16}$  thì sẽ thừa, nên ta chọn  $D^8$  với chuẩn hoá theo cách thức chỉ dùng 63 mức giá trị.

$$\text{vùng 1} \quad p = \frac{I(x,y) * 63}{84} = \frac{3 * I(x,y)}{4}$$

$$\text{vùng 2} \quad p = \frac{(I(x,y) - 85) * 3}{4}$$

$$\text{vùng 3} \quad p = \frac{(I(x,y) - 170) * 3}{4}$$

Thuật toán phân nguồng dùng ma trận nguồng được mô tả như sau:

for each  $I[i,j]$  do

if  $I[i,j] < 84$  then {vùng 1}

Begin  $I[i,j] := I[i,j] * 3/4$

$x_0 := (i \bmod 8) + 1; y_0 := (j \bmod 8) + 1;$

nguong :=  $D[x_0, y_0]$ ;

If  $I[i,j] < \text{nguong}$  then Hien(j+x+1, i+y+1, 0)

Else Hien(j+x+1, i+y+1, 0)

end

Else If  $I[i,j] < 169$  then {vùng 2}

Begin  $I[i,j] := (I[i,j] * 3-85)/4$

$x_0 := (i \bmod 8) + 1; y_0 := (j \bmod 8) + 1;$

nguong :=  $D[x_0, y_0]$ ;

If  $I[i,j] < \text{nguong}$  then Hien(j+x+1, i+y+1, 1)

Else Hien(j+x+1, i+y+1, 2)

end

Else {vùng 3}

Begin  $I[i,j] := (I[i,j] * 3-170)/4$

$x_0 := (i \bmod 8) + 1; y_0 := (j \bmod 8) + 1;$

nguong :=  $D[x_0, y_0]$ ;

```

If I[i,j] < nguong then Hien(j+x+1, i+y+1,2)
Else Hien(j+x+1, i+y+1,3)
end
End

```

## 2.6. KHÁI NIỆM ẢNH ĐEN TRẮNG, ĐA CẤP XÁM, ẢNH MÀU

Có xu hướng hay nhầm lẫn giữa các khái niệm ảnh nhị phân, ảnh đen trắng cũng như ảnh đơn sắc (một màu — monochrome) với ảnh màu. Thực tế, ảnh đen trắng bao gồm ảnh nhị phân và ảnh đa cấp xám. Khái niệm cấp xám hay mức xám (grey level) đã đề cập trong phần 1.2.1 chương 1. Dưới đây sẽ làm rõ thêm khái niệm này cũng như cách biểu diễn và lưu trữ các loại ảnh này.

### 2.6.1. Ảnh đen trắng

Ảnh đen trắng chỉ bao gồm 2 màu: màu đen và màu trắng. Người ta phân sự biến đổi đó thành L mức. Nếu L bằng 2, nghĩa là chỉ có 2 mức: mức 0 và mức 1 và còn gọi là ảnh nhị phân. Mức 1 ứng với màu sáng, còn mức 0 ứng với màu tối. Nếu L lớn hơn 2 ta có ảnh đa cấp xám. Việc xác định số mức là phụ thuộc vào tiêu chí lượng hoá. L thường chọn 32, 64, 128 và 256. Ảnh 256 mức là ảnh có chất lượng cao và thường được sử dụng.

Với ảnh nhị phân, mỗi pixel mã hoá trên 1 bit; còn với ảnh 256 mức, mỗi pixel mã hoá trên 8 bit. Ví dụ với một ảnh 256 mức xám, kích thước 512 x 512 cần không gian lưu trữ là 512 x 512 bytes hay 256 Kbytes.

Ảnh nhị phân khá đơn giản, các phần tử ảnh có thể coi như các phần tử logic. Ứng dụng chính của nó được dùng theo tính logic, để phân biệt đối tượng ảnh với nền hay để phân biệt điểm biến với điểm khác.

### 2.6.2. Ảnh màu

Ảnh màu là ảnh tổ hợp từ 3 màu cơ bản: đỏ (R), lục (Green) và tím (Blue) và thường thu nhận trên các dải băng tần khác nhau. Mỗi pixel ảnh màu gồm 3 thành phần như nêu trong phần 2.1.4. Mỗi màu cũng phân thành L cấp khác nhau (L thường là 256). Do vậy, để lưu trữ ảnh màu, người ta có thể lưu trữ từng mặt màu riêng biệt, mỗi màu lưu trữ như một ảnh đa cấp xám. Do đó, không gian nhớ dành cho một ảnh màu lớn gấp 3 lần một ảnh đa cấp xám cùng kích thước.

**Bài tập chương 2**

1. Viết thủ tục hiện ảnh theo kỹ thuật 4 màu với nguồng ngẫu nhiên.
2. Với ma trận Dither cấp 4 -  $D^4$  đã cho trong giáo trình, hãy viết chương trình tính ma trận  $D^8$  và  $D^{16}$ .
3. Với ma trận  $D^4$  vừa tính được, hãy áp dụng kỹ thuật Dithering để biến đổi ảnh ra và so sánh kết quả. Ảnh vào có thể tự tạo hay sử dụng ảnh đã tạo ra ở chương I.
4. Viết thủ tục hiển thị hiện chúc năng in ảnh từ màn hình ra máy in:
  - a) đọc một ảnh PCX hay BMP và hiển thị lên màn hình;
  - b) dùng kỹ thuật Dither để in ảnh.
5. Viết thủ tục đọc một ảnh PCX (giải nén) và lưu vào bảng 2 chiều.
6. Viết thủ tục lưu ảnh số biểu diễn bởi bảng lên tệp PCX.
7. Xây dựng giải thuật chuyển đổi định dạng PCX sang BMP và ngược lại.
8. Đọc và hiển thị ảnh với định dạng TIFF.
9. Đọc và hiển thị ảnh với định dạng GIF.
10. Một ảnh da cấp xám kích thước 1024 x 1024, mỗi pixel mã hóa trên 12 bit.
  - a) Số mức lượng hóa tối đa của ảnh là bao nhiêu?
  - b) Không gian nhớ dùng lưu trữ cho ảnh ?
  - c) Nếu ảnh trên là ảnh màu thì ảnh đó có tối đa bao nhiêu màu?

**3**

## CÔNG CỤ TRỢ GIÚP XỬ LÝ ẢNH SỐ

### TOOLS FOR IMAGE PROCESSING

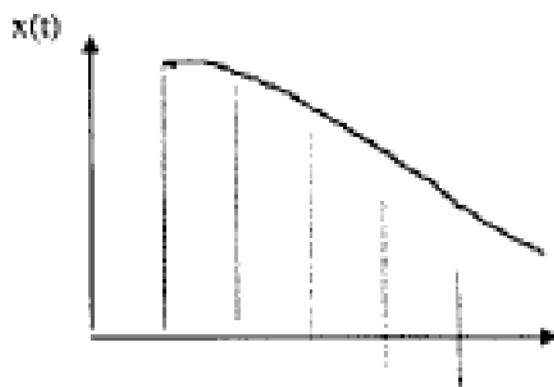
Thuật ngữ "xử lý ảnh số" thường dùng để chỉ các quá trình xử lý ảnh 2 chiều bằng máy tính. Ảnh số thường được biểu diễn bởi ma trận 2 chiều các số thực hay số phức gồm một số hữu hạn các bit. Để có thể xử lý được trên máy tính, ảnh đã cho (ảnh, giấy phim hay đồ thị) đầu tiên phải được số hóa (digitalized) và lưu dưới dạng ma trận 2 chiều các bit. Trong chương này chúng ta sẽ đề cập tới các công cụ và các kỹ thuật sử dụng trong xử lý ảnh số. Trước tiên là giới thiệu tổng quan về xử lý ảnh số (tín hiệu trong không gian). Tiếp theo, giới thiệu một số khái niệm như : toán tử tuyến tính, tích chập (convolution product) và lọc số (filtering) - các công cụ cơ bản và ứng dụng của chúng trong xử lý ảnh. Kế đó trình bày về một số biến đổi hay dùng như biến đổi Fourier, biến đổi Karhunen Loeve. Các công cụ xử lý điểm ảnh được trình bày chi tiết về nguyên tắc cũng như công cụ lược đồ xám (histogram) và các phép biến đổi lược đồ. Cuối cùng là một số kỹ thuật khác trong mô hình thống kê.

### 3.1. TỔNG QUAN VỀ XỬ LÝ ẢNH TRONG KHÔNG GIAN

#### 3.1.1. Tín hiệu số và biểu diễn ảnh số

Như đã nêu trong chương 1, một hàm hai biến thực hoặc phức có thể coi như một ảnh. Một ảnh trong không gian 2 chiều có thể biểu diễn bởi một tập hợp các ma trận cơ sở gọi là ảnh cơ sở. Như vậy một tín hiệu 2 chiều liên tục trong không gian, theo khái niệm trên, gọi là ảnh liên tục trong không gian số thực và ký hiệu là  $f(x,y)$ : giá trị của  $f(x,y)$  là liên tục trong khoảng  $(-\infty, \infty)$ .

Các tín hiệu liên tục theo thời gian qua quá trình số hóa ta thu được tín hiệu rời rạc (tín hiệu số).



Hình 3.1. Tín hiệu số rời rạc.

Ảnh số chính là ảnh xử lý bằng máy tính thu được từ ảnh liên tục bởi quá trình số hoá (lấy mẫu và lượng hoá), thường được ký hiệu là  $I[m,n]$ . Giá trị  $I[x,y]$  biểu diễn cường độ sáng được mã hoá của mỗi điểm ảnh  $(x,y)$ . Giá trị đó còn gọi là mức xám (grey level). Vậy  $I[x,y]$  có giá trị rời rạc và để tiện xử lý, ta coi giá trị của  $I[x,y]$  là nguyên:  $I[x,y] \in \{0, 1, \dots, L-1\}$  với  $L$  là mức xám tối đa dùng để biểu diễn.

Để giảm độ phức tạp tính toán, các giá trị của  $(m,n)$  thường chọn là hữu hạn và thường chọn là 512; còn  $L$  chọn là 256. Ảnh có nhiều mức xám gọi là ảnh đa cấp xám. Ảnh chỉ có 2 mức xám 0 và 1 gọi là ảnh nhị phân.

Với cách biểu diễn trên, ảnh số chính là một phần của tín hiệu số trong không gian 2 chiều. Và cách biểu diễn ảnh số thông dụng nhất là dùng bằng 2 chiều mà thuật ngữ thường gọi là ma trận ảnh hay bản đồ ảnh.

### 3.1.2. Khái quát về hệ thống xử lý tín hiệu số

Hệ thống số là một hệ thống tiếp nhận tín hiệu số ở đầu vào, xử lý tín hiệu theo một qui trình nào đấy và đưa ra cũng là một tín hiệu số. Vì ảnh số là một phần của tín hiệu số, nên hệ thống xử lý ảnh số có đặc thù như hệ thống số cộng thêm một số tính chất riêng.

Nếu gọi tín hiệu số đầu vào là  $X(m,n)$ , tín hiệu số đầu ra là  $Y(m,n)$ , đặc trưng của hệ thống là  $H$ , ta có thể biểu diễn hệ thống số một cách hình thức như sau:

$$Y(m,n) = H [X(m,n)]$$

Phản lớn các hệ thống số là tuyến tính và bất biến. Khái niệm tuyến tính và bất biến sẽ trình bày trong phần 3.2. Trong xử lý tín hiệu số, thường có 2 cách tiếp cận khác nhau:

- Biên độ của tín hiệu được lấy mẫu, lượng hoá theo một qui chuẩn và có thể biểu diễn bởi một hàm liên tục theo thời gian. Đây là cách tiếp cận theo không gian thực.

- Cách tiếp cận thứ hai là tiếp cận theo miền tần số của tín hiệu. Trong cách tiếp cận này, trước tiên tín hiệu được biến đổi chẵng hạn như phép biến đổi Fourier, sau đó, tiến hành xử lý trên miền tần số. Cuối cùng dùng biến đổi ngược để đưa tín hiệu đã xử lý về miền số thực.

Thí dụ như tín hiệu thu nhận là tiếng còi ô tô. Ta có thể tiếp cận theo 2 cách khác nhau:

- Lấy mẫu biến độ tín hiệu nhiều lần trong một chu kỳ và được một xấp xỉ của tín hiệu là một hàm liên tục theo thời gian.

- Phân tích tín hiệu theo độ cao của âm thanh hay tần số của âm thanh và lưu trữ biến độ của mỗi tần số.

Hai cách tiếp cận trên cho ta 2 kỹ thuật cơ bản được dùng trong xử lý ảnh (đề cập trong các phần sau):

- Tác động trực tiếp lên điểm ảnh: tích chập, lọc số và các toán tử điểm.

- Biểu diễn ảnh sang một không gian khác bằng các biến đổi, xử lý và biến đổi ngược lại.

### 3.2. CÁC TOÁN TỬ KHÔNG GIAN (SPATIAL OPERATORS)

Các toán tử không gian (KG) thường dùng là các toán tử tuyến tính, tích chập và lọc. Mục đích chính của các toán tử này là làm cho ảnh "tốt hơn" và thuận tiện cho việc biến đổi và xử lý ảnh về sau như: tăng cường và nâng cao chất lượng ảnh, dò biến, trích chọn đặc tính v.v.

#### a) Toán tử tuyến tính

Phần lớn các hệ thống xử lý ảnh có thể mô hình hóa như một hệ thống tuyến tính hai chiều. Giả sử  $x(m,n)$  và  $y(m,n)$  biểu diễn các tín hiệu vào và ra tương ứng của hệ thống. Hệ thống hai chiều được biểu diễn bởi:

$$y(m,n) = H[x(m,n)] \quad (3.1)$$

Hệ thống này gọi là tuyến tính *khi và chỉ khi*: tổ hợp tuyến tính của 2 tín hiệu vào  $x_1(m,n)$ ,  $x_2(m,n)$  cũng tạo nên chính tổ hợp tuyến tính tương ứng của đầu ra  $y_1(m,n)$ ,  $y_2(m,n)$ , nghĩa là: với 2 hằng số bất kì  $\alpha$  và  $\beta$ , ta có:

$$\begin{aligned} H[\alpha x_1(m,n) + \beta x_2(m,n)] &= \alpha H[x_1(m,n)] + \beta H[x_2(m,n)] \\ &= [\alpha y_1(m,n)] + [\beta y_2(m,n)] \end{aligned} \quad (3.2)$$

Phương trình 3.2 gọi là *chỗng tuyến tính* của 2 tín hiệu. Ý nghĩa quan trọng của hệ tuyến tính là: khi có nhiều tín hiệu vào, hệ thống có thể xử lý độc lập từng tín hiệu sau đó tổng hợp kết quả lại.

Khi tín hiệu vào là hàm delta Kronecker 2 chiều  $\delta$  (xung đơn vị) tại vị trí  $(m',n')$ , tín hiệu ra ở vị trí  $(m,n)$  được định nghĩa:

$$h(m,n ; m',n') = H[\delta(m-m'; n-n')] \quad (3.3)$$

Dấu ";" trong các công thức trên để phân biệt tọa độ vào và tọa độ ra.

Hàm delta  $\delta(m,n)$  có dạng:

$$\delta(m,n) = \begin{cases} 1 & \text{nếu } m = n \\ 0 & \text{nếu } m \neq n \end{cases}$$

### b) Tích chập

Trước khi đề cập đến khái niệm này, ta xét một khái niệm có liên quan, đó là khái niệm bất biến trượt (shift invariance). Một hệ thống gọi là bất biến trượt nếu dịch chuyển đầu vào thì cũng tạo nên một dịch chuyển tương ứng của đầu ra. Theo phương trình 3.3, nếu xung xảy ra ở gốc tọa độ, ta có:

$$H[\delta(m-n)] = h[m,n ; 0,0] \quad (3.4)$$

$$\Rightarrow h(m,n ; m',n') = h(m-m' ; n-n') \quad (3.5)$$

Theo định nghĩa này, tín hiệu ra có dạng:

$$y(m,n) = \sum_{m',n'=-\infty}^{\infty} h(m-m' ; n-n')x(m',n') \quad (3.6)$$

Phương trình 3.6 gọi là chập của đầu vào  $x(m',n')$  với đáp ứng xung (impulse response)  $h(m,n)$ .

Hình 3.2 minh họa toán tử chập. Ma trận đáp ứng xung quay quanh gốc  $180^\circ$  và trượt một khoảng  $(m,n)$  rồi chồng lên ma trận tín hiệu vào  $x(m',n')$ .

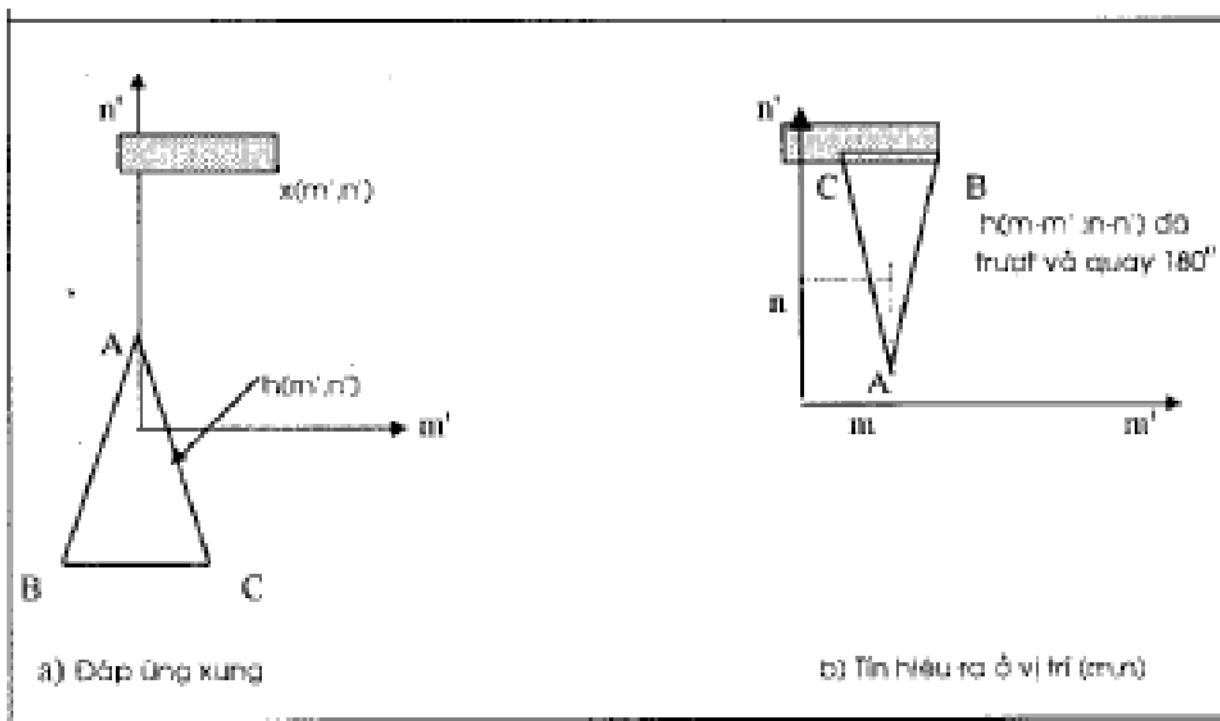
Toán tử tích chập được định nghĩa như sau:

+ trường hợp liên tục

$$g(x,y) = h(x,y) \otimes f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x-x',y-y')f(x',y')dx'dy' \quad (3.7)$$

+ trường hợp rời rạc

$$y(m,n) = h(m,n) \otimes x(m,n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m-m', n-n') x(m', n') \quad (3.8)$$



Hình 3.2. Một biểu diễn của toán tử chập.

Để tiện theo dõi, ta xét ví dụ sau:

- ma trận tín hiệu  $x_{2 \times 3}$

$$\begin{bmatrix} 1 & 4 & 1 \\ 2 & 5 & 3 \end{bmatrix}$$

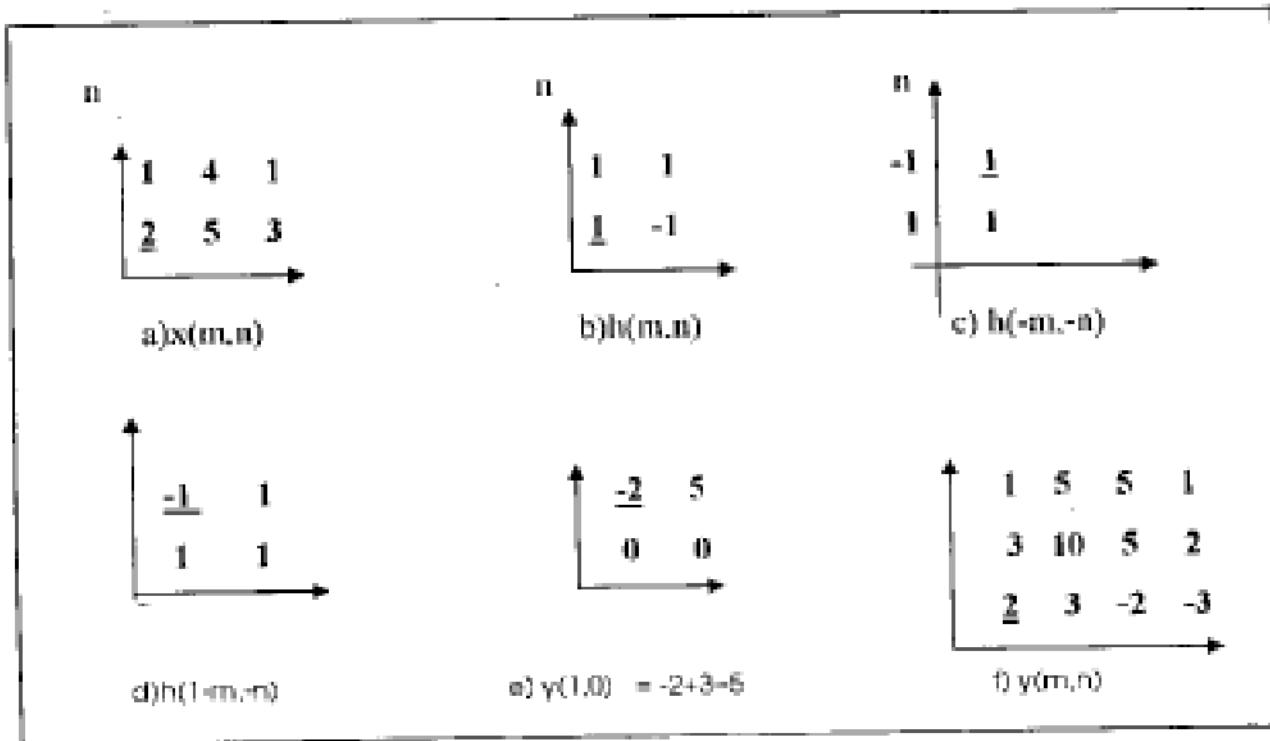
- ma trận đáp ứng xung  $h_{2 \times 2}$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Ma trận thu được bởi tích chập của 2 ma trận  $h$  và  $x$  là một ma trận  $4 \times 3$ . Nói chung, chập của 2 ma trận số  $(M_1 \times N_1)$  và  $(M_2 \times N_2)$  là một ma trận có  $(M_1 + M_2 - 1, N_1 + N_2 - 1)$ . Hình 3.3 dưới đây mô tả các bước thực hiện chập của 2 ma trận  $h$  và  $x$  ở trên. Các số gạch dưới là điểm bắt đầu thực hiện qua mỗi bước.

Theo công thức 3.8, tích chập  $H \otimes X$  có độ phức tạp tính toán rất cao. Để giảm độ phức tạp tính toán người ta thường dùng nhân chập  $H_{K,L}$  có kích thước hữu hạn và nhỏ. Nhân chập này thường chọn có kích thước lẻ và các giá trị hay dùng là:  $K = L = 3, 5, 7$ . Trong các phần sau, ta thấy đa số các nhân chập được sử dụng trong tích chập, lọc số là

nhân chập vuông, đôi khi là nhân chập chữ thập. Thực ra nhân chập chữ thập là nhân chập vuông, song một số phần tử của nó có giá trị 0 nên ta coi như không có.



Hình 3.3. Ví dụ về toán tử chập cuộn.

Với cách chọn nhân chập như trên, hai công thức tính nhân chập sau đây thường được sử dụng:

- *Xếp chồng tại biên*

$$Y(m,n) = \sum_{k=0}^{L-1} \sum_{l=0}^{M-1} H(k,l)^* X(m-k, n-l) \quad (3.9)$$

Theo công thức này, nếu  $K=L=3$ , nhân chập  $H$  có thể viết:

$$H(k,l) = \begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix}$$

Với nhân chập  $H$  như trên, công thức (3.9) được viết lại một cách tường minh như sau:

$$Y(m,n) = H_{00}X_{m,n} + H_{01}X_{m,n-1} + H_{02}X_{m,n-2} + H_{10}X_{m-1,n} + H_{11}X_{m-1,n-1} + H_{12}X_{m-1,n-2} + H_{20}X_{m-2,n} + H_{21}X_{m-2,n-1} + H_{22}X_{m-2,n-2}.$$

với  $m \in [1, M]$  và  $n \in [1, N]$ . Các chỉ số được viết theo ký pháp toán cho dễ quan sát.

Như vậy, việc tính  $Y(m,n)$  như được thực hiện bởi xoay nhân chập  $H$   $90^\circ$  rồi xếp chồng với lân cận của điểm  $(m,n)$  sao cho  $H_{22}$  trùng với điểm lấy ra. Điều này lý giải cho tên gọi của phương pháp “*xếp chồng tại biên*”.

- *Xếp chồng tại trung tâm*

$$Y(m,n) = \sum_{k=1}^L \sum_{l=1}^L H(k,l)^* X(m-k+L_c, n-l+L_c) \text{ với } L_c = \frac{L+1}{2} \quad (3.10)$$

Theo công thức này, nếu  $K=L=3$ , nhân chập  $H$  có thể viết:

$$H(k,l) = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

Và công thức (3.10) được viết lại một cách tương minh như sau:

$$Y(m,n) = H_{11}X_{m+1,n+1} + H_{12}X_{m+1,n} + H_{13}X_{m+1,n-1} + H_{21}X_{m,n+1} + H_{22}X_{m,n} + H_{23}X_{m,n-1} + H_{31}X_{m-1,n+1} + H_{32}X_{m-1,n} + H_{33}X_{m-1,n-1}$$

với  $m \in [1, M]$  và  $n \in [1, N]$ . Các chỉ số được viết theo ký pháp toán cho dễ quan sát.

Như vậy, việc tính  $Y(m,n)$  như được thực hiện bởi xoay nhân chập  $H$   $90^\circ$  rồi xếp chồng với lân cận của điểm  $(m,n)$  sao cho  $H_{22}$  trùng với điểm lấy ra (điểm lấy ra trùng với tâm nhân chập). Điều này cũng lý giải cho tên gọi của phương pháp “*xếp chồng tại trung tâm*”.

Thực tế, công thức này có thể áp dụng cho cả 2 trường hợp. Nếu áp dụng để tính cho điểm ở biên, ta coi các điểm ngoài biên có giá trị 0. Thí dụ, cho ảnh số I sau:

$$I = \begin{bmatrix} 4 & 7 & 2 & 7 & 1 \\ 5 & 7 & 1 & 7 & 1 \\ 6 & 6 & 1 & 8 & 3 \\ 5 & 7 & 5 & 7 & 1 \\ 5 & 7 & 6 & 1 & 2 \end{bmatrix}$$

và nhân chập  $H$ :

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

tích chập  $H \otimes I$  tính theo công thức 3.10 được:

$$H \otimes I = \frac{1}{9} \begin{bmatrix} 23 & 26 & 31 & 19 & 16 \\ 35 & 39 & 46 & 31 & 27 \\ 36 & 43 & 49 & 34 & 27 \\ 36 & 48 & 48 & 34 & 22 \\ 24 & 35 & 33 & 22 & 11 \end{bmatrix}$$

Để hiểu rõ cách tính nhân chập, ta tính một vài phần tử. Các phần tử còn lại coi như một bài tập nhỏ tự làm.

$$Y(1,1) = \frac{1}{9} (1^*7 + 1^*5 + 1^*7 + 1^*4) = \frac{1}{9} *23$$

$$Y(4,2) = \frac{1}{9} (1^*6 + 1^*7 + 1^*5 + 1^*5 + 1^*7 + 1^*5 + 1^*1 + 1^*6 + 1^*6) = \frac{1}{9} *48$$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 7 \\ 0 & 5 & 2 \end{bmatrix}$$

nhân chập

các lân cận

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 6 & 6 & 1 \\ 5 & 1 & 2 \\ 5 & 7 & 6 \end{bmatrix}$$

nhân chập      các lân cận

Vị trí điểm ảnh đầu ra là điểm được in đậm

a) Tính  $Y(1,1)$

b) Tính  $Y(4,2)$

Tích chập là một khái niệm rất quan trọng trong xử lý ảnh, đặc biệt là tính chất của nó có liên quan đến biến đổi Fourier: biến đổi Fourier của một tích chập bằng tích đơn giản các biến đổi Fourier của các tín hiệu đó:

$$F[H(x,y) \otimes I(x,y)] = F[H(x,y)] \cdot F[I(x,y)] \quad (3.11)$$

Trong kỹ thuật, người ta gọi  $H$  là nhân chập hay nhân cuộn và cũng còn gọi là mặt nạ (mask);  $I[x,y]$  trong công thức trên là ảnh đối tượng.

Dưới đây, đưa ra một thuật toán tổng quát để tính nhân chập dùng cho mọi trường hợp. Để sử dụng thuật toán này chỉ cần thay đổi 2 thông số: ma trận biểu diễn ảnh số cần xử lý và ma trận biểu diễn nhân chập. Thuật toán được mô phỏng dưới dạng Pascal:

**NhanChap(ImgIn,ImgOut: ảnh;H: Nhân chập;N:kích thước ảnh;w:kích thước nhân chập)**

/\*      Vào: ImgIn

```

Nhân chập H
Ra: ImagOut    */
Begin
    For i:=1 to N do
        For j:=1 to N do
            Begin   Sum :=0; Lc:=(w+1) div 2;
            For k:=1 to w do
                For l:=1 to w do
                    Begin   Col:=i-k+Lc;Row:=j+l-Lc
                    If (Col<>0)and (Col <=N) then
                        , If (Row<>0)and (Row <=N) then
                            Sum:= Sum + ImagIn[Col,Row] * H[k,l];
                    End;
                    ImagOut[i,j]:=Sum
                End;
            End;
        End;
    End;

```

### c) Kỹ thuật lọc số

Trong nhiều lĩnh vực kỹ thuật, nhiều đóng vai trò chủ yếu gây nên những khó khăn khi ta cần phân tích một tín hiệu nào đó, cũng không loại trừ tín hiệu ảnh. Giữa một ảnh thực và ảnh số hoá thu nhận được khác nhau khá nhiều vì có nhiều quá trình can thiệp vào. Nguyên nhân là do nhiều điện tử của máy thu hay chất lượng kém của bộ số hoá. Ta xem xét để biết nhiều thể hiện trên ảnh thế nào. Giả sử ảnh là một miền có mức xám đồng nhất. Như vậy các phần tử của ma trận biểu diễn ảnh sau quá trình số hoá phải có cùng giá trị. Nhưng thực tế quan sát, ta thấy: gần giá trị trung bình của mức xám có những phần tử trội lên khá nhiều. Đó chính là hiện tượng nhiễu. Như vậy, nhiều trong ảnh số được xem như sự dịch chuyển nhanh của tín hiệu thu nhận (tín hiệu ảnh  $I[m,n]$ ) trên một khoảng cách ngắn. Xem xét một cách tương đương trong không gian tần số, nhiễu ứng với các thành phần tần số cao trong ảnh. Do vậy, người ta nghĩ đến việc biến đổi có tính đến ảnh hưởng của các phần tử lân cận bằng cách lấy "tổ hợp" các điểm lân cận này (trong không gian thực) hay lọc các thành phần tần số cao (trong không gian tần số). Đây chính là *kỹ thuật lọc*

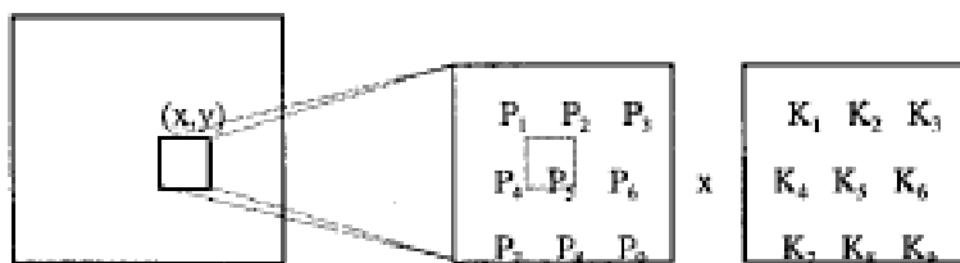
(filtering). Cơ sở lý thuyết của kỹ thuật lọc số là dựa trên tính dữ thừa thông tin không gian: các pixel lân cận có thể có cùng hoặc gần cùng một số đặc tính. Hơn nữa, nhiều có thể coi như sự đổi biến của một điểm ảnh so với các điểm lân cận.

Trong kỹ thuật này, người ta sử dụng một mặt nạ và di chuyển khắp ảnh gốc. Tuỳ theo cách tổ hợp điểm đang xét với các điểm lân cận mà ta có kỹ thuật lọc tuyến tính hay phi tuyến. Điểm ảnh chịu tác động của biến đổi là điểm ở tâm mặt nạ.

#### • Lọc tuyến tính

Trong kỹ thuật lọc tuyến tính, ảnh thu được sẽ là tổng trọng số hay là trung bình trọng số các điểm lân cận với nhân cuộn hay mặt nạ. Nguyên tắc lọc theo tổng trọng số được minh họa qua hình 3.4. Thí dụ tâm mặt nạ là điểm  $P_5$ , thì điểm  $P_5$  mới sẽ được tính theo công thức sau:

$$P_5 = P_1K_1 + P_2K_2 + P_3K_3 + P_4K_4 + P_5K_5 + P_6K_6 + P_7K_7 + P_8K_8 + P_9K_9$$



8 lân cận của  $P_5$       Nhân cuộn 3 \* 3

Hình 3.4. Lấy tổ hợp các điểm ảnh lân cận.

Nói chung, người ta sử dụng nhiều kiểu mặt nạ khác nhau:

$$H_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad H_2 = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad H_3 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Mặt nạ  $H_1$  là mặt nạ dùng để tính trung bình không trọng số (không ưu tiên theo hướng nào). Mặt nạ  $H_2$  cho trọng số lớn nhất với điểm ở tâm. Còn mặt nạ  $H_3$  ưu tiên cho 2 hướng x, y.

Giả sử  $I_i$  là ảnh đang xét,  $I_f$  là ảnh thu được và cả 2 ảnh đều có cùng kích thước  $p \times p$ . Với mặt nạ trên, mỗi điểm ảnh thu được  $I_f(x,y)$  sẽ được tính bởi:

$$\begin{aligned}
 I_f &= \frac{1}{9} (I(x-1,y-1) + I(x-1,y) + I(x-1,y+1) + I(x,y-1) + I(x,y) + I(x,y+1) \\
 &\quad + I(x+1,y-1) + I(x+1,y) + I(x+1,y+1)) \\
 &= \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 H_i(i+1,j+1) I(x+i,y+j)
 \end{aligned} \tag{3.12}$$

Nếu  $H$  là bộ lọc kích thước  $(n+1) \times (n+1)$ ,  $n$  chẵn và tổng các hệ số là  $K$ ,  $I_f$  sẽ được tính bởi:

$$I_f = \frac{1}{K} \sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} H_i(i+n/2, j+n/2) I(x+i, y+j) \tag{3.13}$$

Công thức trên chính là tích chập giữa mặt nạ  $H$  và ảnh gốc  $I$ :  $I_f = H \otimes I$ . Trên thực tế, kỹ thuật lọc tuyến tính được xây dựng trên nền phép nhân chập. Phụ thuộc vào dấu ra mà chọn các nhân chập có ý nghĩa thích hợp (xem chương 4 và chương 5).

Chú ý rằng vừa rồi ta chưa xét đến biên của ảnh khi sử dụng kỹ thuật lọc. Giả sử ta áp mặt nạ  $H$  vào điểm tại gốc tọa độ  $(0,0)$ , rõ ràng là điều này không thể được. Do vậy, chỉ có thể lọc phần trong của ảnh từ  $n/2$  đến  $p-n/2$  và trong trường hợp này ta thu được ảnh cỡ  $(p+1-n) \times (p+1-n)$  hoặc là tạo thêm một nửa cỡ  $n/2$  bằng cách sao.

Ngoài các bộ lọc trên, người ta cũng hay dùng bộ lọc Gauss. Bộ lọc này có ưu điểm là dễ cài đặt và cho chất lượng cao. Bộ lọc Gauss gồm tích chập của một ảnh  $I_f$  với mặt nạ Gauss  $G(x,y,\sigma)$ :  $I_f = G \otimes I$ , với

$$G(x,y,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$G$  là mặt nạ hình vuông mà các hệ số của nó là các phần tử rời rạc của phân bố Gauss. Vì mặt nạ có kích thước  $(n+1) \times (n+1)$  hữu hạn, còn đường cong  $G$  định nghĩa trên toàn miền thực, do vậy ta cần chọn một khoảng hữu hạn. Thường người ta chọn khoảng là  $4\sigma$  (95%) hay  $6\sigma$  (99.9%).

Người ta cũng chứng minh được rằng với mặt nạ  $N \times N$  cần  $N^2$  phép nhân và  $N^2-1$  phép cộng. Các phương pháp lọc nói trên, nhìn chung làm giảm mức nhiễu trắng đi  $N_v$  lần, với  $N_v$  là số phần tử của mặt nạ và hạn chế nhòe sau khi lọc.

#### • Lọc phi tuyến

Khác với lọc tuyến tính, kỹ thuật lọc phi tuyến coi một điểm ảnh kết quả không phải

Là tổ hợp tuyến tính của các điểm lân cận. Bộ lọc phi tuyến thường dùng là lọc trung vị (median filtering) mang tên Tukey.

Trong trường hợp một chiều, trung vị  $x_m$  của một chuỗi  $n$  phần tử  $\{x_i\}$  được định nghĩa:

- Nếu  $n$  lẻ: có  $(n-1)/2$  phần tử lớn hơn  $x_m$  và  $(n-1)/2$  nhỏ hơn hay bằng  $x_m$ .

- Nếu  $n$  chẵn:  $x_m$  là trung bình cộng của 2 phần tử  $x_{n/2}$  và  $x_{n/2+1} \in \{x_i\}$  sao cho có  $(n-2)/2$  phần tử nhỏ hơn hay bằng  $x_m$  và  $(n-2)/2$  phần tử lớn hơn hay bằng  $x_m$ .

Kỹ thuật lọc trung vị được dùng để lọc nhiễu bằng cách trượt trên mặt phẳng ảnh, mỗi lần trượt di chuyển một cột điểm. Những phần tử trong cửa sổ được xem như là 1 chuỗi  $\{x_i\}$  và điểm quan tâm được thay thế bởi giá trị  $x_m$  của chuỗi. Thị dụ như chuỗi  $\{1,2,9,5,4\}$ , điểm trung tâm sẽ được thay thế bởi giá trị 4 được tính theo nguyên tắc ở trên. Rõ ràng trong ví dụ này, giá trị 9 có thể là nhiễu nhọn trong dãy tăng dần.

Lọc trung vị thường sử dụng cửa sổ kích thước 3. Tuy nhiên, nếu không có dấu hiệu quan trọng nào bị mất, kích thước cửa sổ có thể tăng lên 5, 7, v.v. và sẽ kết thúc khi quá trình lọc không làm thay đổi kết quả.

Khái niệm lọc trung vị dễ dàng mở rộng cho trường hợp hai chiều. Giả sử đầu vào là  $X(m,n)$  và đầu ra bộ lọc là  $Y(m,n)$ . Lọc trung vị hai chiều được định nghĩa:

$$Y(m,n) = \text{Median}(X(m-k,n-l)), \text{ với } k,l \in [1, L]$$

Lưu ý rằng công thức  $L_c = (L+1)/2$  còn gọi là bán kính bộ lọc. Do vậy, ta có cách viết khác tương đương  $(k,l) \in (-r,r)$  với  $2r + 1 = L$ .

Khi đó trung vị của cửa sổ vuông  $n \times n$  có thể được tính như những phần tử của chuỗi một chiều. Ta tiến hành sắp xếp dãy đó rồi thay thế phần tử tâm của sổ bằng trung vị của dãy vừa tìm được.

Thuật toán được minh họa như sau:

Giả sử ta dùng nhân chấp 3x3 và các phần tử trong cửa sổ có dạng:  $n$

Điểm xét  $X(m,n) = 78$  (nhiều)

Dãy lấy ra và sắp lại ta có:

15	15	16	17	47	17	18	20	78
1	2	3	4	5	6	7	8	9

	15	17	18
	16	78	17
	17	15	20

Trung vị của dãy là phần tử ở vị trí số 5 và có giá trị là 17.

Giá trị mới này được thay cho phần tử tại tâm (78).

Như vậy là nhiễu đã bị khử.

Với cách thức như vậy, ta lần lượt rẽ cùm số lọc đi khắp ảnh và tiến hành lọc. Lưu ý rằng các ảnh mới phải lưu trữ khác với ảnh gốc.

Với lọc trung vị, số lượng tính toán khá lớn (có thể bằng số mũ của kích thước của số lọc). Vì vậy, để khắc phục nhược điểm này, người ta dùng một phương pháp khác: lọc giả trung vị (Pseudo-Median Filter). Thí dụ với dãy 5 số: a, b, c, d, e, lọc giả trung vị được định nghĩa như sau:

$$\text{PseudoMedian}(a,b,c,d,e) = \frac{1}{2} \left\{ \begin{array}{l} \left[ \text{MAX}[\text{Min}(a,b,c), \text{Min}(b,c,d), \text{Min}(c,d,e)] \right] \\ + \left[ \text{MIN}[\text{Max}(a,b,c), \text{Max}(b,c,d), \text{Max}(c,d,e)] \right] \end{array} \right\}$$

Rõ ràng là với phương pháp này, ta chỉ phải dùng 3 chuỗi con thay vì dùng 10 chuỗi như lọc trung vị.

Một cách tổng quát, ta có thuật toán sau:

b1. *Lấy các phần tử trong cửa sổ ra mảng một chiều (L phần tử).*

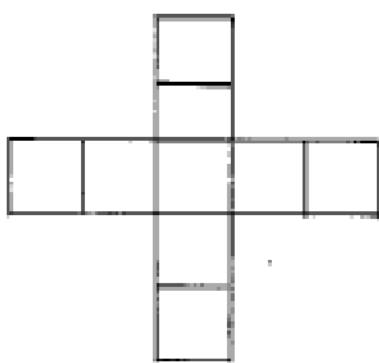
b2. *Tìm min của lần lượt các chuỗi con rồi lấy max: gọi m1 là giá trị này.*

b3. *Tìm max của lần lượt các chuỗi con rồi lấy min: gọi m2 là giá trị tìm được.*

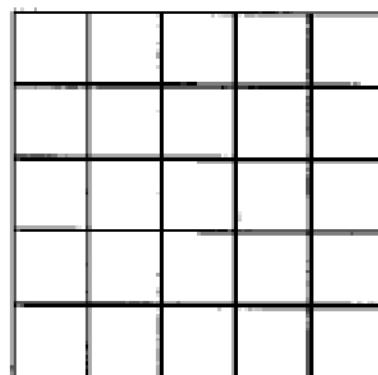
b4. *Gán giá trị điểm đang xét là trung bình cộng của m1 và m2.*

Lọc giả trung vị có nhiều điểm giống như lọc trung vị. Dãy lấy ra không cần sắp xếp và giá trị gọi là trung vị lại được tính theo trung bình cộng Max của min và Min của max.

Hai loại mặt nạ hay dùng là mặt nạ vuông và mặt nạ chữ thập. Thực tế, người ta thích loại mặt nạ vuông hơn vì nó không làm biến dạng ảnh mà lại hiệu quả. Tuy nhiên trong lọc giả trung vị, người ta lại thấy dùng cửa sổ chữ thập cho kết quả khả quan hơn nhiều.



a) mặt nạ chữ thập



b) mặt nạ vuông 5x5

Hình 3.5. Mặt nạ vuông và mặt nạ chữ thập.

Ngoài cách phân loại dựa theo tính chất tổ hợp, trên quan điểm tần số, người ta phân các bộ lọc theo đặc trưng tần số: lọc thông thấp, thông cao, chẵn dải, . . . Các kỹ thuật lọc thông thấp được dùng để lọc nhiễu. Lọc thông cao dùng để làm nổi bật các chi tiết có tần số không gian cao (thí dụ như các điểm biên, các điểm gồ) mà không ảnh hưởng đến các chi tiết có tần số thấp. Do vậy, các phần tử có tần số không gian cao sẽ được cải thiện (sáng hơn), còn các phần tử có tần số không gian thấp sẽ đen đi. Kỹ thuật lọc thông cao cũng được thực hiện nhờ thao tác nhân chập. Các nhân chập thông cao hay được sử dụng:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

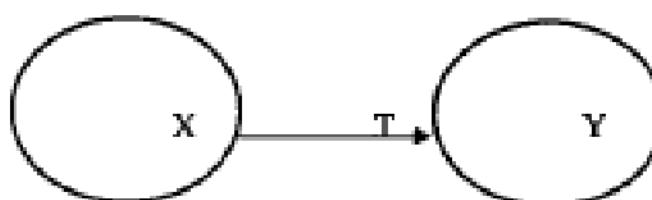
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

### 3.3. CÁC BIẾN ĐỔI KHÔNG GIAN: BIẾN ĐỔI FOURIER VÀ BIẾN ĐỔI KL (SPATIAL TRANSFORMS)

Các phép biến đổi là cách tiếp cận thứ hai được áp dụng trong tín hiệu số nói chung và trong xử lý ảnh số nói riêng. Phép biến đổi (transform) là thuật ngữ dùng để chỉ việc chuyển đổi sự biểu diễn của một đối tượng từ không gian này sang một không gian khác. Thí dụ,  $X$  là một đối tượng trong không gian  $\mathcal{X}$ , phép biến đổi  $T$  biểu diễn bởi ma trận  $A$  sẽ chuyển biểu diễn  $X$  sang  $Y$  trong không gian  $\mathcal{Y}$  như sau:

$$Y = AX$$



Không gian  $\mathcal{X}$

Không gian  $\mathcal{Y}$

Như vậy, biến đổi ảnh (Image Transform) nhằm chuyển đổi sự biểu diễn ảnh từ một không gian ban đầu sang một không gian khác sao cho việc xử lý được tiện lợi hơn.

Để theo dõi một cách có hệ thống, trước tiên ta xem xét khái niệm chung về biến đổi ảnh trong ngữ cảnh của xử lý ảnh. Ta nói khai triển chuỗi trực giao tổng quát của một ảnh số  $u(m,n)$ , kích thước  $N \times N$  là một cặp biến đổi có dạng:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m,n) a_{k,l}(m,n) \quad \text{với } k,l = 0, 1, \dots, N-1 \quad (3.14)$$

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k,l) a^*(k,l)(m,n) \quad \text{với } k,l = 0, 1, \dots, N-1 \quad (3.15)$$

Trong đó  $\{a_{k,l}(m,n)\}$  gọi là một biến đổi ảnh. Đó chính là tập các hàm cơ sở (trong xử lý ảnh gọi là các ảnh cơ sở).

Theo định nghĩa, một biến đổi tương ứng với  $A$  là unita và tách được (separable unitary transforms) nếu:

$$AA^{*T} = A^TA^* = I \quad \text{với } A \text{ là ma trận biến đổi; } A^T \text{ là ma trận chuyển vị của } A.$$

Nhìn chung, trong xử lý ảnh số, ta hay dùng biến đổi đơn vị trực giao và tách được. Trong ngữ cảnh này, viết dưới dạng ma trận ta có:

$$v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a(k,m) u(m,n) a(l,n) \Leftrightarrow V = UAU^T \quad (3.16)$$

$$u(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a^*(k,m)v(k,l)a^*(l,n) \Leftrightarrow U = A^{*T}VA^* \quad (3.17)$$

Thí dụ, cho  $A$  là ma trận của biến đổi trực giao và  $U$  là một ảnh:

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Theo công thức trên, ta có:

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}$$

và

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Có rất nhiều phép biến đổi được dùng trong xử lý ảnh như biến đổi Fourier, biến đổi Cosin, Karhunen-Loeve,... Tuy nhiên, để trọng sáng cách trình bày, trong phần dưới đây ta chỉ xét 2 biến đổi quan trọng là biến đổi Fourier TF (Fourier Transform) và biến đổi

KL(Karhunen-Loeve). Biến đổi Cosin rời hữu ích trong mén ảnh sẽ được đề cập đến trong phần nén ảnh (chương 8).

### 3.3.1. Biến đổi Fourier

Trước tiên ta xem xét các khái niệm và bản chất của biến đổi TF cho tín hiệu số một chiều và hai chiều. Vì ánh số chỉ là một phần của tín hiệu số nên phải dùng một dạng khác của biến đổi TF đó là biến đổi Fourier rời rạc DFT(Discrete Fourier Transform). Cuối cùng, sẽ trình bày sơ trinh bày thuật toán biến đổi nhanh FFT(Fast Fourier Transform) để tính các DFT.

### 3.3.1.1. Biến đổi Fourier: khái niệm và công thức

Biến đổi Fourier cho một tín hiệu có thể hình dung như sau:

$$x(t) \quad \text{TF} \quad X(f)$$

### Miền thời gian

Một số ứng dụng cần miễn phí, người ta dùng biến đổi phức (biến đổi z):

$x(n) \xrightarrow{TZ} X(z)$  với  $z$  là biến phức

Biến đổi Fourier cho một tín hiệu một chiều gồm một cặp biến đổi:

- Biến đổi thuận: chuyển sự biểu diễn từ không gian thực sang không gian tần số (phổ và pha). Các thành phần tần số này được gọi là cách biểu diễn trong không gian Fourier của tín hiệu.
  - Biến đổi ngược: chuyển đổi sự biểu diễn của đối tượng từ không gian Fourier sang không gian thực.

a) Không gian một chiều

Cho một hàm  $f(x)$  liên tục. Biến đổi Fourier của  $f(x)$ , kí hiệu  $F(u)$ ,  $u$  biểu diễn tần số không gian, được định nghĩa:

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i ux} dx \quad (3.18)$$

trong đó:

$f(x)$ : biểu diễn biến đổi tín hiệu

$e^{-2\pi i \alpha}$ : biểu diễn pha.

Biến đổi ngược của  $F(u)$  cho  $f(x)$  được định nghĩa:

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{2\pi i ux} du \quad (3.19)$$

### b) Không gian hai chiều

Cho  $f(x,y)$  hàm biểu diễn ảnh liên tục trong không gian 2 chiều, cặp biến đổi Fourier cho  $f(x,y)$  được định nghĩa:

- Biến đổi thuận  $F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)e^{-2\pi i (ux+vy)} dx dy \quad (3.20)$

$u,v$  biểu diễn tần số không gian.

- Biến đổi ngược  $f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v)e^{2\pi i (ux+vy)} du dv \quad (3.21)$

#### 3.3.1.2. Biến đổi Fourier rời rạc - DFT

Biến đổi DFT được phát triển dựa trên biến đổi Fourier cho ảnh số. Ở đây, ta dùng tổng thay cho tích phân. Biến đổi DFT tính các giá trị của biến đổi Fourier cho một tập các giá trị trong không gian tần số được cách đều.

### a) DFT cho tín hiệu một chiều

Với tín hiệu một chiều, người ta biểu diễn bởi một chuỗi trực giao các hàm cơ sở. Với các hàm liên tục, khai triển chuỗi trực giao sẽ cung cấp chuỗi các hệ số dùng trong nhiều quá trình khác nhau hay trong phân tích hàm. Khai triển Fourier rời rạc DFT cho một dãy  $\{u(n), n = 0, 1, \dots, N-1\}$  định nghĩa bởi:

$$v(k) = \sum_{n=0}^{N-1} u(n) W_N^{nk} \text{ với } k = 0, 1, \dots, N-1 \quad (3.22)$$

với  $W_N = e^{j2\pi/N}$

và biến đổi ngược  $u(n) = \frac{1}{N} \sum_{k=0}^{N-1} v(k) W_N^{-kn} , n = 0, 1, \dots, N-1 \quad (3.23)$

Thực tế trong xử lý ảnh người ta hay dùng DFT đơn vị:

$$v(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} u(n) W_N^{kn} , k = 0, 1, \dots, N-1 \quad (3.24)$$

$$u(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} v(k) W_N^{kn}, n = 0, 1, \dots, N-1 \quad (3.25)$$

Các DFT và DFT đơn vị có tính đối xứng. Hơn nữa khai triển DFT và DFT đơn vị của một chuỗi và biến đổi ngược lại của nó có tính chu kỳ và chu kỳ N.

### b) DFT cho tín hiệu hai chiều (ảnh số)

DFT hai chiều của một ảnh  $M \times N : \{u(m,n)\}$  là một biến đổi tách được và được định nghĩa:  $v(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} u(m,n) W_N^{km} W_M^{ln} \quad 0 \leq k \leq N-1$  (3.26)

và biến đổi ngược:

$$u(m,n) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} v(k,l) W_N^{-km} W_M^{-ln} \quad 0 \leq m, n \leq N-1 \quad (3.27)$$

Cặp DFT đơn vị hai chiều được định nghĩa:

$$v(k,l) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} u(m,n) W_N^{km} W_M^{ln} \quad 0 \leq k, l \leq N-1 \quad (3.28)$$

$$u(m,n) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} v(k,l) W_N^{-km} W_M^{-ln} \quad 0 \leq m, n \leq N-1 \quad (3.29)$$

Viết lại công thức 3.27 và 3.28, ta có:

$$v(k,l) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} u(m,n) W_N^{(km+ln)} \quad 0 \leq k, l \leq N-1 \quad (3.30)$$

$$u(m,n) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} v(k,l) W_N^{-(km+ln)} \quad 0 \leq m, n \leq N-1 \quad (3.31)$$

Ở đây,  $W_N^{(km+ln)}$  là ma trận ảnh cơ sở. Nhắc lại rằng  $e^{j\alpha} = \cos(\alpha) + j\sin(\alpha)$  (công thức Euler).

Do vậy:

$$W_N^{(km+ln)} = e^{j2\pi(km+ln)/N} = \cos(2\pi(km+ln)/N) + j\sin(2\pi(km+ln)/N).$$

Như vậy, các hàm cơ sở trong ma trận ảnh cơ sở của biến đổi Fourier là các hàm cosine và hàm sine. Theo tính toán trên, ta thấy biến đổi Fourier biểu diễn ảnh trong không gian mới theo các hàm sine và cosine.

### 3.3.1.3. Một số tính chất và áp dụng

#### a) Tính chất

- Đổi xứng và đơn vị

$$\bar{F}^T = \bar{F}, \quad \bar{F}^{-1} = \bar{F}^*$$

- Chu kỳ

$$v(k+N, l+N) = v(k, l) \quad \forall k, l \quad (3.32)$$

$$u(k+N, l+N) = u(k, l) \quad \forall k, l \quad (3.33)$$

Việc chứng minh tính có chu kỳ của biến đổi Fourier coi như một bài tập nhỏ. Bạn đọc tự kiểm nghiệm lại.

- Phổ Fourier mẫu hóa

$$\text{nếu } \bar{U}(m,n) = \begin{cases} U(m,n) & 0 \leq m, n \leq N-1 \\ 0 & \text{nếu không} \end{cases}$$

thì  $\bar{U}(2k/N, 2l/N) = \text{DFT}[u(m,n)] = v(k,l)$  với  $\bar{U}(w_1, w_2)$  là biến đổi Fourier của  $u(m,n)$ .

- Biến đổi nhanh

Vì DFT hai chiều là tách được, do đó biến đổi  $V = FUF^*$  tương đương với DFT đơn vị 1 chiều  $2N$ .

- Liên hiệp đổi xứng:

DFT và DFT đơn vị của một ảnh thực có tính đổi xứng liên hợp:

$$v(N/2 \pm k, N/2 \pm l) = v^*(N/2 \pm k, N/2 \pm l) \quad 0 \leq k, l \leq N/2-1 \quad (3.34)$$

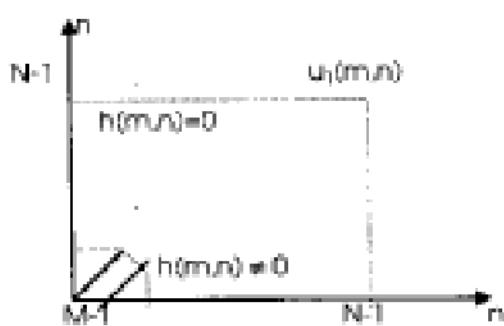
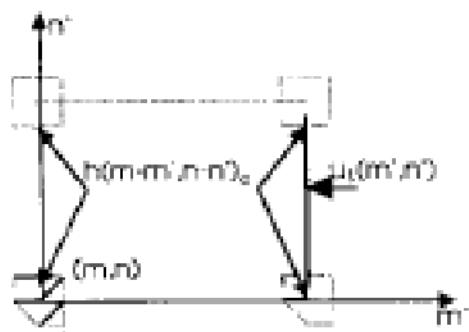
$$\text{hay} \quad v(k, l) = v^*(N-k, N-l) \quad \text{với} \quad 0 \leq k, l \leq N/2-1 \quad (3.35)$$

#### b) Định lý chập cuộn 2 chiều

DFT của chập cuộn hai chiều của hai ma trận bằng tích DFT của chúng:

$$u(m, n) = \sum_{m'=0}^{N-1} \sum_{n'=0}^{N-1} h(m-m', n-n') u_1(m', n') \quad 0 \leq m, n \leq N-1 \quad (3.36)$$

Với  $h(m, n)$ ,  $u_1(m, n)$  là ma trận  $N \times N$  và  $h(m, n)_c = h(m \bmod N, n \bmod N)$ . Hình 3.6 cho thấy ý nghĩa của chập tròn. Chúng là như nhau khi chu kỳ mở rộng của  $h(m, n)$  là chập trên miền  $N \times N$  với  $u_1(m, n)$ .

a) ma trận  $h(m,n)$ b) chập tròn  $h(m,n)$  với  $u_1(m,n)$ trên miền  $N \times N$ 

Hình 3.6. Chập cuộn tròn.

## c) Thuật toán biến đổi nhanh -FFT(Fast Fourier Transform)

## - Trường hợp 1 chiều

Từ công thức  $v(k) = \frac{1}{N} \sum_{n=0}^{N-1} u(n) W_N^{kn}$  với  $k=0, 1, \dots, N-1$ , ta nhận thấy:

với mỗi giá trị  $k$  ta cần  $N$  phép nhân và  $N$  phép cộng. Suy ra rằng để tính  $N$  giá trị của  $v(k)$  ta cần  $N^2$  phép nhân. Để tính toán một cách hiệu quả, người ta dùng thuật toán tính nhanh gọi là FFT với độ phức tạp tính toán là  $O(N \log_2 N)$ .

Thuật toán tính nhanh có thể tóm tắt như sau:

- giả sử  $N = 2^n$

- giả sử  $W_N$  là nghiệm thứ  $N$  của đơn vị:  $W_N = e^{-j2\pi/N}$  và  $M = \frac{N}{2}$  ta có:

$$v(k) = \frac{1}{2M} \sum_{n=0}^{M-1} u(n) W_{2M}^{nk}$$

- Khai triển công thức trên ta được:

$$v(k) = \left( \frac{1}{M} \sum_{n=0}^{M-1} u(2n) W_{2M}^{2nk} + \frac{1}{M} \sum_{n=0}^{M-1} u(2n+1) W_{2M}^{(2n+1)k} \right) / 2 \quad (3.37)$$

vì  $W_{2M}^{2nk} = W_{2M}^{nk}$ , do đó:

$$v(k) = \frac{1}{2} [u_{even}(n) + u_{odd}(n)]$$

Chú ý rằng  $v(k)$  với  $k = [0, M-1]$  là một DFT trên  $M = N/2$ . Thực chất thuật toán FFT là dùng nguyên tắc chia đôi và tính chu kỳ để tính DFT. Với  $k = [0, M-1]$  ta dùng công

thức 3.37; với  $k = [M, 2M-1]$  ta dùng phép trừ trong công thức 3.37. Có thể dùng thuật toán này có sửa đổi một chút để tính DFT ngược. Bạn đọc coi như một bài tập.

- **Trường hợp 2 chiều**

Do DFT 2 chiều là tách được nên từ công thức (3.29), ta có:

$$v(k,l) = \frac{1}{N} \sum_{m=0}^{N-1} W_N^{-km} \sum_{n=0}^{N-1} u(m,n) W_N^{-ln} \quad (3.38)$$

Từ công thức 3.38, ta có cách tính DFT hai chiều như sau:

- Tính DFT 1 chiều với mỗi giá trị của  $x$  (theo cột).
- Tính DFT 1 chiều theo hướng ngược lại (theo hàng) với giá trị thu được ở trên.

### 3.3.2 Biến đổi KL

Biến đổi KL có nguồn gốc từ khai triển chuỗi của các quá trình ngẫu nhiên liên tục. Biến đổi KL cũng còn được gọi là biến đổi Hotelling hay phương pháp thành phần chính. Để tiện theo dõi ta cũng cần nhắc lại một số khái niệm và định nghĩa trong xử lý thống kê.

#### 3.3.2.1 Một số định nghĩa và khái niệm

$X$  là một biến vector ngẫu nhiên gồm  $n$  thành phần  $x_i$ ,  $i = 1, 2, \dots, n$ . Mỗi thành phần  $x_i$  là giá trị ngẫu nhiên. Người ta định nghĩa:

- Ký vọng toán học (*Trung bình số học*)  $E[x] = \int_{-\infty}^{\infty} x P(x) dx$  (3.39)

với  $P(x)$  là hàm mật độ xác suất và  $x$  là biến ngẫu nhiên liên tục.

- Mômen toán học

$$m_k = \int_{-\infty}^{\infty} x^k P(x) dx = E[x^k] \quad (3.40)$$

$m_k$  gọi là mô men bậc  $k$  của  $x$ .

- Tính tương quan: một tín hiệu phụ thuộc vào thời gian.
- Hàm tự tương quan của 1 tín hiệu  $x(t)$  được định nghĩa:

$$\Psi_{xx} = E[x(t)x(t+\tau)] \quad (3.41)$$

- Hàm tương quan của 2 tín hiệu:

$$\Psi_{xy} = E[x(t)y(t+\tau)] \quad (3.42)$$

- Cho tập các đối tượng  $X$ , ma trận tương quan của tập các đối tượng ký hiệu là  $R$  và được định nghĩa  $R = E[X X^T] = \langle XX^T \rangle$ . Viết dưới dạng ma trận ta có:

$$R = \begin{bmatrix} E[x_{11}] & E[x_{12}] & \dots & E[x_{1n}] \\ E[x_{21}] & E[x_{22}] & \dots & E[x_{2n}] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_{m1}] & E[x_{m2}] & \dots & E[x_{mn}] \end{bmatrix} \quad (3.43)$$

\* *Má trận hiệp biến*, ký hiệu  $A = E[(X-M)(X-M^T)]$

$$= \langle (X-M)(X-M)^T \rangle \quad (3.44)$$

Trong một số trường hợp  $A = \langle XX^T \rangle - \langle MM^T \rangle = R - \langle MM^T \rangle$  (\*). Nếu đối tượng không tương quan (độc lập) lúc đó ma trận A là ma trận đường chéo. Có nghĩa là:

$$a_{ii} = x_i^2 + m_i^2 \neq 0 \text{ còn } a_{ij} = 0 \text{ với } i \neq j.$$

### 3.3.2.2 Cơ sở lý thuyết của biến đổi KL

Đây là phép biến đổi không gian n chiều thành không gian m chiều, với  $m < n$ . Mỗi thành phần của véc-tơ miêu tả một đặc tính của đối tượng. Nếu ta biến đổi được từ không gian n chiều về không gian m chiều, như vậy ta sẽ làm giảm được thông tin dư thừa (theo thuật ngữ trong xử lý ảnh hay nhận dạng gọi là *giảm thiểu nguyên*).

Mục đích của biến đổi KL là chuyển từ không gian n chiều sang không gian trực giao m chiều sao cho sai số bình phương là nhỏ nhất. Gọi U là tập các véc-tơ cơ sở trong không gian trực giao  $U = \{u_1, u_2, \dots, u_n\}$ ,

$$u_i = \begin{bmatrix} u_{1j} \\ u_{2j} \\ \vdots \\ u_{nj} \end{bmatrix}$$

với  $j = 1, 2, \dots, n$  và

$$u_i \cdot u_k = \begin{cases} 0 & \text{nếu } i \neq k \\ 1 & \text{nếu } i = k. \end{cases}$$

Mỗi véc-tơ y trong không gian trực giao có thể viết:

$$y = \varphi_1 u_1 + \varphi_2 u_2 + \dots + \varphi_n u_n = \Phi U \text{ với } \Phi = [\varphi_1, \varphi_2, \dots, \varphi_n]$$

$$\Rightarrow \Phi = U^T y.$$

Gọi  $\bar{X}$  là kết quả thu được trong không gian m chiều và  $\bar{X} = \varphi_1 u_1 + \varphi_2 u_2 + \dots + \varphi_m u_m \approx X$ .

$$\text{Sai số trong phép biến đổi } \hat{\mathbf{x}} = \mathbf{X} \cdot \mathbf{x} = \sum_{i=1}^n \varphi_i u_i \cdot \sum_{i=1}^m \varphi_i u_i = \sum_{i=m+1}^n \varphi_i u_i \quad (3.45)$$

$$\text{Sai số trung bình bình phương } \zeta = E[\epsilon^2] = E[(\bar{\mathbf{x}} - \mathbf{x})^T(\bar{\mathbf{x}} - \mathbf{x})] \quad (3.46)$$

$$\begin{aligned} &= \langle (\bar{\mathbf{x}} - \mathbf{x})^T(\bar{\mathbf{x}} - \mathbf{x}) \rangle \\ &= \left\langle \sum_{i=m+1}^n \varphi_i^2 \right\rangle \\ &= \sum_{i=m+1}^n \langle \varphi_i^2 \rangle \end{aligned} \quad (3.47)$$

$$\text{mà } \Phi = \mathbf{U}^T \mathbf{X}, \text{ do đó } \zeta = \sum_{i=m+1}^n (u_i^T \mathbf{X})(u_i \mathbf{X})^T = \sum_{i=m+1}^n u_i^T \langle \mathbf{X} \mathbf{X}^T \rangle u_i \quad (3.48)$$

$$\text{Theo định nghĩa của R, phương trình 3.48 trở thành } \zeta = \sum_{i=m+1}^n u_i^T R u_i \quad (3.49)$$

$\zeta$  đạt min khi (3.49) đạt min.

$$\text{Đặt } \eta = \zeta + \sum_{i=m+1}^n \lambda_i (1 - u_i^T u_i). \quad (3.50)$$

Như vậy  $\eta$  đạt min khi (3.50) min. Để tìm min của (3.50) ta dùng phương pháp đạo hàm và dẫn đến việc giải phương trình:

$$(R - \lambda I)u_i = 0 \quad (3.51)$$

Phương trình (3.51) gọi là phương trình đặc trưng của R với  $\lambda_i$  là các trị riêng và  $u_i$  là các vectơ riêng tương ứng. Đây chính là cơ sở lý thuyết của biến đổi KL.

### 3.3.2.3. Biến đổi KL

#### Định nghĩa và khái niệm

Cho  $u$  là một vectơ các số thực ngẫu nhiên; vectơ cơ sở của biến đổi KL là các vectơ riêng trực giao của ma trận hiệp biến R (định nghĩa trong phần 3.3.2.1) cho bởi phương trình:  $R\varphi_k = \lambda_k \varphi_k; \quad 0 \leq k \leq N-1$

$$\text{Biến đổi KL của } u \text{ là } v = \Phi^{*T} u \quad (3.52)$$

$$\text{và biến đổi ngược } u = \Phi v = \sum_{k=0}^{N-1} v(k) \Phi_k \quad (3.53)$$

$u$  là vectơ cột,  $v$  là vectơ hàng và  $\Phi_k$  là cột thứ  $k$  của ma trận  $\Phi$ .

Biến đổi  $\Phi$  đưa R về dạng đường chéo :

$$\Phi^{-T} R \Phi = \lambda = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix}$$

Thường người ta hay làm việc với ma trận A hơn.

### Biến đổi KL của ảnh

Nếu một ảnh  $u(m,n)$  NxN được biểu diễn bởi trường ngẫu nhiên, ma trận A cho bởi:

$$E[u(m,n)u(m',n')] = r(m,n;m',n') \quad 0 \leq m,m',n,n' \leq N-1 \quad (3.54)$$

thì ảnh cơ sở của biến đổi KL là các hàm riêng, chuẩn và trực giao  $\phi_{k,l}$  là lời giải của phương trình:

$$\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} r(m,n;m',n') \phi_{k,l} = \lambda_{k,l} \phi_{k,l} \quad (3.55)$$

$$\text{Theo ký pháp ma trận ta có: } R\Psi_i = \lambda_i \phi_i \quad i = 0, 1, \dots, N^2-1 \quad (3.56)$$

với  $\phi_i$  là véctơ  $N^2 \times 1$  biểu diễn của  $\Psi_i$ , và R là ma trận  $N^2 \times N^2$  ánh xạ vào véctơ u,  $R = E[uu]$ .

Nếu R là tách được thì ma trận  $\Psi_{N^2 \times N^2}$  mà các cột là  $\Psi_i$  sẽ tách được:

$$\phi_{k,l}(m,n) = \phi_1 \otimes \phi_2 \text{ hay } R = R_1 \otimes R_2 \quad (3.57)$$

Biến đổi KL của U là  $V = \Psi^{*T}u = \phi_1^{*T} \otimes \phi_2^{*T}$

$$\text{và biến đổi ngược } U = \phi_1 V \phi_2 \quad (3.58)$$

### 3.4. TOÁN TỬ XỬ LÝ ĐIỂM ẢNH

Ảnh thô có cấu trúc đơn giản, song lại rất phức tạp về nội dung. Như chúng ta biết, ảnh là một tập hợp các điểm ảnh, chứa một lượng thông tin khá lớn. Thường để xử lý ảnh, người ta hay biểu diễn ảnh dưới một dạng khác để có thể làm rõ một số tính chất của chúng. Xử lý điểm ảnh thực chất là dùng các ảnh xạ nhằm biến đổi giá trị của một điểm chỉ dựa vào giá trị của chính nó mà không quan tâm tới các giá trị của các điểm ảnh khác. Một cách toán học, ảnh xạ đó được định nghĩa như sau:

$$v(m,n) = f(u(m,n)L)$$

- trong đó:
- $u(m,n)$  thể hiện giá trị cường độ sáng tại toạ độ  $(m,n)$ ;
  - $v(m,n)$  là giá trị cường độ sáng thu được sau phép biến đổi;
  - $f$  là hàm biến đổi. Nó có thể là hàm liên tục hay hàm rời rạc.

Chi tiết về các hàm này và cách vận dụng được trình bày kỹ trong chương 4 (4.1.1).

Xử lý điểm ảnh là một trong các phép xử lý cơ bản và đơn giản. Có 2 cách tiếp cận trong cách xử lý này: dùng một hàm thích hợp tuỳ theo mục đích cải thiện ảnh để biến đổi giá trị của điểm ảnh (mức xám) sang một giá trị khác (mức xám mới). Cách thứ hai là dựa vào kỹ thuật biến đổi lược đồ xám (histogram).

### 3.4.1. Xử lý điểm ảnh bằng ảnh xá biến đổi

Bản chất của xử lý điểm ảnh như đã nói trên là nhằm biến đổi giá trị của một điểm ảnh bằng một hàm tuyến tính hay phi tuyến (hàm mũ, hàm logarit). Các phép xử lý này là cơ sở cho biến đổi độ tương phản của ảnh: co giãn, tăng giảm và biến đổi độ tương phản vì độ tương phản trên một ảnh chỉ phụ thuộc vào độ sáng của mỗi điểm ảnh. Giả sử ta dùng một hàm phi tuyến dạng  $f = a \log()$ :

$$Y[m,n] = a \log(X[m,n]).$$

Nếu ảnh có kích thước  $512 \times 512$  ta cần  $512^2$  phép biến đổi. Một cách tổng quát, nếu ảnh có kích thước  $N \times N$  thì phép biến đổi sẽ có độ phức tạp tính toán là  $O(N^2)$ . Nếu chú ý rằng ảnh gồm  $N \times N$  điểm song chỉ có  $L$  mức xám ( $L$  rất nhỏ so với  $N^2$ ) và phép biến đổi chỉ nhằm biến đổi một mức xám  $i \in L$  sang một mức xám  $i' \in L'$  (mức xám kết quả) thì ta có thể thực hiện nhanh hơn. Do vậy, ta có cách tính sau:

- Tính  $L$  giá trị của hàm  $f$  và lưu vào một bảng:  $y_i = f(x_i)$  với  $i=1,2,\dots,L$ .
- Duyệt toàn bộ ảnh, với mỗi điểm ảnh ta tra giá trị trong bảng (không cần tính) và thu được ảnh mới.

Kỹ thuật này có tên gọi là kỹ thuật bảng tra - LUT(Look Up Table). Để minh họa, xét thí dụ sau. Cho ảnh số X:

$$X = \begin{bmatrix} 2 & 1 & 2 & 3 \\ 3 & 1 & 2 & 5 \\ 2 & 2 & 3 & 4 \\ 3 & 2 & 3 & 2 \end{bmatrix}$$

Ảnh này có 16 điểm song chỉ có 5 mức xám. Hàm biến đổi là hàm  $\log()$ . Bảng tra có giá trị:

Mức xám	Bảng tra (LUT)
1	a log(1)
2	a log(2)
3	a log(3)
4	a log(4)
5	a log(5)

Ảnh thu được sau phép biến đổi:

$$X = \begin{bmatrix} a \log(2) & a \log(1) & a \log(2) & a \log(3) \\ a \log(3) & a \log(1) & a \log(2) & a \log(5) \\ a \log(2) & a \log(2) & a \log(3) & a \log(4) \\ a \log(3) & a \log(2) & a \log(3) & a \log(2) \end{bmatrix}$$

Thuật toán biến đổi được mô tả như sau:

(Bảng tra có tên là LUT và có L phần tử)

#### a) Tính bảng LUT

For k = 1 to L do  $LUT[k] := f(x_k)$

#### b) Biến đổi

For each pixel  $X[i,j]$  do  $Y[i,j] := LUT(X[i,j])$

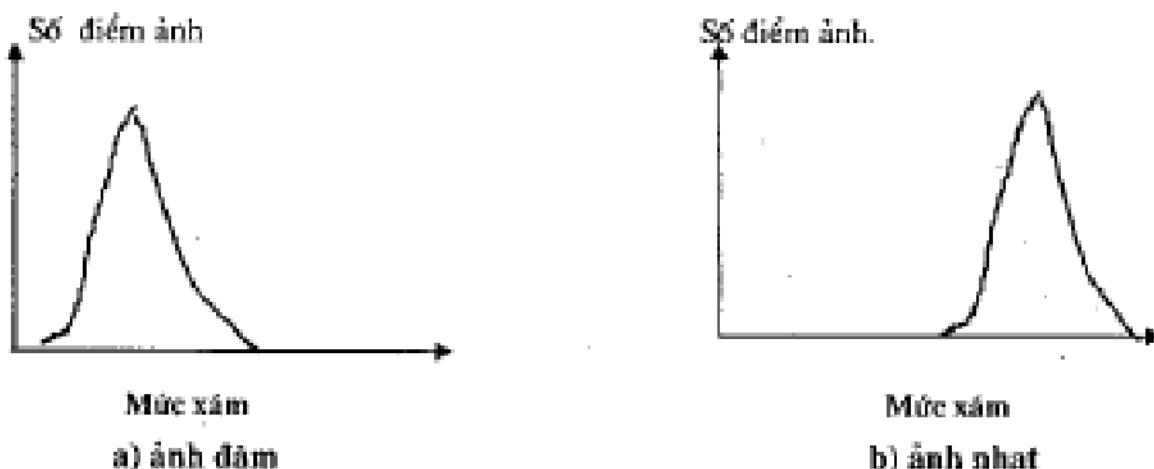
Như vậy, để có thể lập trình, phụ thuộc vào các hàm biến đổi khác nhau, ta chỉ cần viết hàm tính bảng tra (tham số là hàm) còn phép biến đổi là như nhau.

### 3.4.2 Lược đồ mức xám (histogram)

Lược đồ mức xám của một ảnh, từ nay về sau ta qui ước gọi là *lược đồ xám*, là một hàm cung cấp tần suất xuất hiện của mỗi mức xám (grey level).

Lược đồ xám được biểu diễn trong một hệ toạ độ vuông góc x,y. Trong hệ toạ độ này, trục hoành biểu diễn số mức xám từ 0 đến N, N là số mức xám (256 mức trong trường hợp chúng ta xét). Trục tung biểu diễn số điểm ảnh cho một mức xám (số điểm ảnh có cùng mức xám). Cũng có thể biểu diễn khác một chút: trục tung là tỷ lệ số điểm ảnh có cùng mức xám trên tổng số điểm ảnh.

Lược đồ xám cung cấp rất nhiều thông tin về phân bố mức xám của ảnh. Theo thuật ngữ của xử lý ảnh gọi là *tinh động* của ảnh. Tinh động của ảnh cho phép phân tích trong khoảng nào đó phân bố phần lớn các mức xám của ảnh: ảnh rất sáng hay ảnh rất đậm.



Nếu ảnh sáng, luợc đồ xám nằm bên phải (mức xám cao), còn ảnh đậm luợc đồ xám nằm bên trái (mức xám thấp).

Theo định nghĩa của luợc đồ xám, việc xây dựng nó là khá đơn giản. Thuật toán xây dựng luợc đồ xám có thể mô tả như sau:

#### Bắt đầu

$H$  là bảng chứa luợc đồ xám (là vector có  $N$  phần tử)

#### *a. Khởi tạo bảng*

For  $i := 1$  to  $N$  do  $H[i] := 0$ ;

#### *b. Tạo bảng*

Với mỗi điểm ảnh  $I(x,y)$  tính  $H[I(x,y)] = H[I(x,y)] + 1$

#### *c. Tính giá trị Max của bảng $H$ . Sau đó hiện bảng trong khoảng từ 0 đến Max.*

#### Kết thúc

Luợc đồ xám là một công cụ hữu hiệu dùng trong nhiều công đoạn của xử lý ảnh như *tăng cường ảnh* (xem chương 4). Dưới đây ta xem xét một số biến đổi luợc đồ xám hay dùng.

### 3.4.3. Biến đổi luợc đồ xám

Trong tăng cường ảnh, các thao tác chủ yếu dựa vào phân tích luợc đồ xám. Trước tiên ta xét bảng tra LUT(Look Up Table). Bảng tra LUT là một bảng chứa biến đổi một mức xám  $i$  sang mức xám  $j$  như đã nói trong phần 3.4.1. Một cách toán học, LUT được định nghĩa như sau:

- Cho  $G_i$  là tập các mức xám ban đầu  $G_i = \{0, 1, \dots, N_i\}$

- Cho  $G_f$  là tập các mức xám kết quả  $G_f = \{0, 1, \dots, N_f\}$

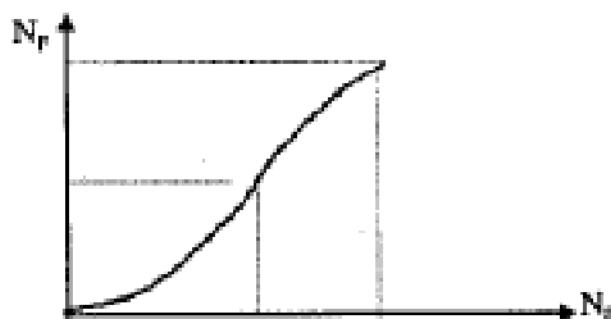
để cho tiện ta cho  $N_i = N_f = 255$ .

-  $f$  là ánh xạ từ  $G_i$  vào  $G_f$ :  $\forall g_i \in G_i$  sẽ  $\exists g_f \in G_f$  mà  $g_f = f(g_i)$

Với mỗi giá trị của mức xám ban đầu ứng với một giá trị kết quả. Việc chuyển đổi một mức xám ban đầu về một mức xám kết quả tương ứng có thể dễ dàng thực hiện được nhờ một bảng tra.

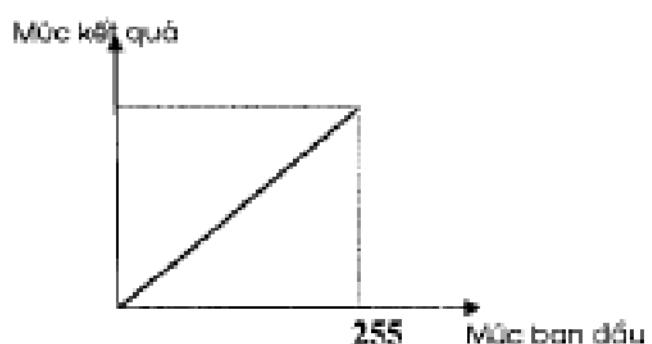
Khi đã xây dựng được bảng, việc sử dụng bảng là khá đơn giản. Người ta xem xét mức xám của mỗi điểm ảnh, nhờ bảng tra tính được mức xám kết quả. Gọi là bảng tra,

thực ra là một véc-tơ có  $N_i + 1$  phần tử. Mỗi phần tử của bảng chứa một giá trị mức xám kết quả. Có hai kiểu bảng tra: bảng đồng nhất và bảng nghịch đảo. Với bảng đồng nhất, giá trị



Hình 3.9. Ánh xạ biến đổi lược đồ xám.

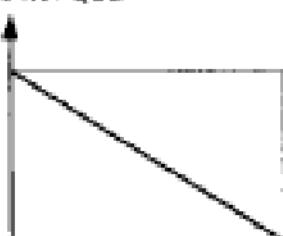
mức xám ban đầu cũng chính là giá trị mức xám kết quả; còn với bảng nghịch đảo, nếu giá trị mức xám ban đầu là  $g_i$  thì giá trị mức xám kết quả là  $255 - g_i$ .



0	1	2	.....	254	255
0	1	2	.....	254	255

Hình 3.10 a. LUT đồng nhất.

Mục kết quả



Mục ban đầu



Hình 3.10 b.  
LUT nghịch đảo.

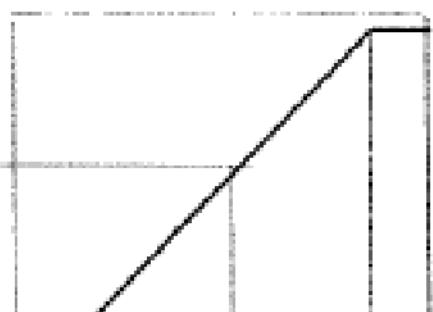
Một trong những ứng dụng phổ biến của LUT là viền khung động. Một số ảnh ban đầu hoặc có thể là rất đậm hay rất nhạt, hoặc độ tương phản thấp. Điều này có thể là do trong ảnh ban đầu, các mức xám có thể vượt lên cao hoặc xuống dưới tỷ lệ, hay tập trung lại trong một vùng rất hẹp (trên lược đồ xám thể hiện rõ điều này).

Mục đích của LUT là phân bổ lại mức xám để chúng có thể phù trên toàn dài - đó chính là viền khung động. Việc chọn giá trị Min và Max là phụ thuộc vào từng ứng dụng.

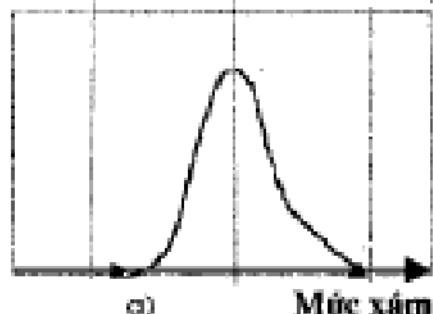
Một ứng dụng khác của LUT là làm nổi bật một số dài mức xám của ảnh. Điều này có thể thực hiện được nhờ viền khung động tại miền quan tâm, bên ngoài miền đặt giá trị là 0 hay nhị phân hóa ảnh (binarisation).



c)



b)



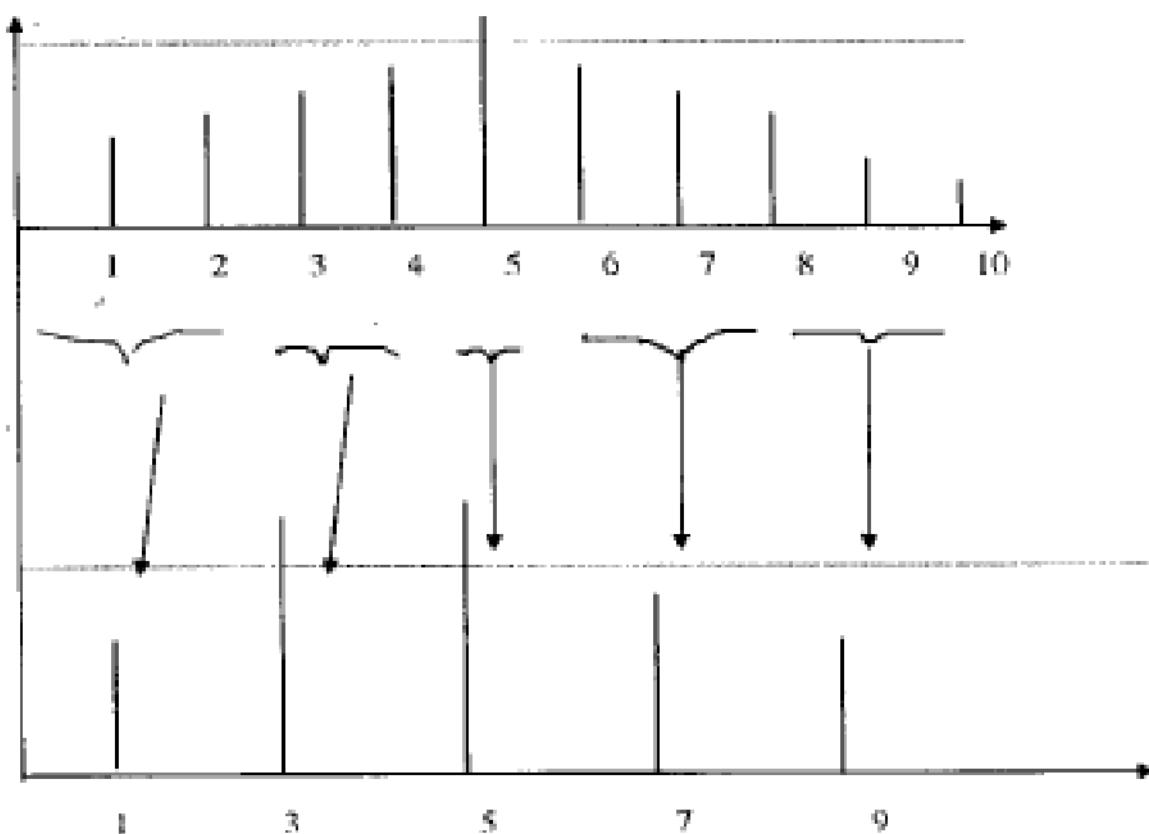
c)

Hình 3.11. Nguyên tắc viền khung động:

- Lược đồ ảnh gốc;
- LUT của viền khung động;
- Lược đồ xám của ảnh được viền.

Với một ảnh tự nhiên được lượng hoá một cách tuyến tính, phần lớn các điểm ảnh có giá trị thấp hơn độ sáng trung bình. Trong miền tối, ta khó có thể cảm nhận các chi tiết của ảnh. Thực tế cần phải khắc phục nhược điểm này bằng cách biến đổi lược đồ xám. Người ta biến đổi lược đồ sao cho tiến gần tới lược đồ định trước. Có nhiều phương pháp, trong đó phương pháp phổ dụng là *san bằng lược đồ* (histogram equalisation).

Nếu ảnh có kích thước  $p \times p$  và ảnh kết quả được mã hoá trên  $N_p$  mức xám, thì số điểm ảnh cho 1 mức xám trong lược đồ cần bằng lý tưởng sẽ là hằng số và bằng  $p^2/N_p$  ( $N_p$  là số mức xám đầu ra). Trên thực tế,  $N_p$  thường nhỏ hơn  $N_i$  (số mức xám ban đầu). Nguyên tắc san bằng lược đồ được minh họa trong hình 3.12.



Hình 3.12. Cân bằng lược đồ.

Việc san bằng lược đồ được thực hiện theo thuật toán:

/\*

Ima: ảnh gốc cần san bằng

Histo: lược đồ xám của ảnh

Transfo: bảng san bằng lược đồ

BatDau, KetThuc : điểm bắt đầu và điểm kết thúc mỗi dài xé.

Band, CentreBande: độ rộng băng và trung điểm của dài

$N_p$ : Số mức xám của ảnh gốc

$N_f$ : Số mức xám của ảnh kết quả \*/

### a. Khởi tạo

TBLituong <- pxp/ $N_p$ ;

Band <-  $N_p/N_f$ ;

CentreBande <- Bande/2;

BatDau, KetThuc <- 0;

### b. Tính và biến đổi lược đồ

While KetThuc <  $N_f$  do

Begin Tong <- 0;

While (Tong < TBLituong) and(KetThuc < N\_f ) do

Begin Tong <- Tong + Histo(KetThuc);

Inc(KetThuc)

End;

For i := BatDau to KetThuc -1 do Transfo[i] <- CentreBande;

CentreBande <- CentreBande + Bande;

ebut <- KetThuc;

End

### c. Tính ảnh kết quả

For i := 1 to N do

For j := 1 to N do Begin

Pic <- Ima[i,j];

Ima[i,j] <- Transfo[Pic]

End

Lưu ý rằng,  $N_p$  càng hạn chế thì việc tích hợp càng quan trọng vì giá trị  $p^2/N_p$  sẽ tăng. Trên thực tế, người ta hay dùng  $N_p = N_f/2$  và lặp lại nhiều lần quá trình san bằng. thí dụ với một ảnh 128 x 128 mã hóa trên 256 mức xám, nếu muốn lược đồ san bằng trên 64 mức xám, số lượng trung bình các điểm ảnh lý tưởng sẽ tiệm cận  $128^2/64 = 256$ .

### 3.5. MÔ HÌNH THỐNG KÊ

Mô hình thống kê có một ý nghĩa rất quan trọng trong biểu diễn ảnh cũng như trong nhiều quá trình của xử lý ảnh. Trong mô hình này, mỗi điểm ảnh được xem như một biến ngẫu nhiên  $u$ . Một ảnh là một hàm mẫu của một ma trận biến ngẫu nhiên còn gọi là trường ngẫu nhiên (random field). Thực tế, số biến ngẫu nhiên là rất lớn (262144 biến cho một ảnh 512 x 512). Điều này gây không ít khó khăn vì để đặc tả một hàm mật độ phải cần một khối lượng đo hay quan sát rất lớn. Vì vậy, người ta nghĩ đến sử dụng các đại lượng đặc trưng của phân bố xác suất như: kỳ vọng toán học, moment (đã nêu trong phần 3.3.3).

Những đặc trưng này rất có ích trong kỹ thuật xử lý ảnh không chỉ cho một ảnh mà là cho một lớp ảnh.

- **Mô hình hiệp biến (covariance model)**

Trong mô hình này, người ta chọn kỳ vọng toán học là hằng số  $\mu$ , còn hiệp biến biểu diễn bởi mô hình mỗ tách được hay không tách được (separable or nonseparable). Trong mô hình tách được, hiệp biến 2 chiều có thể biểu diễn bởi tích của hai hiệp biến 1 chiều:

$$r(m,n;m',n') = r_1(m,n) r_2(m',n') \quad (3.59)$$

$$r(m,n) = r_1(m) r_2(n) \quad (3.60)$$

Phương trình 3.59 biểu diễn mô hình không ổn định, còn 3.60 biểu diễn mô hình ổn định. Trong xử lý ảnh, người ta hay dùng hiệp biến ổn định tách được dưới dạng mỗ:

$$r(m,n) = \sigma^2 p_1 |m| p_2 |n| \quad (3.61)$$

với  $|p_1| < 1$ ,  $|p_2| < 1$  và  $\sigma^2$  là phương sai của trường ngẫu nhiên;

$$p_1 = r(1,0)/\sigma^2, p_2 = r(0,1)/\sigma^2.$$

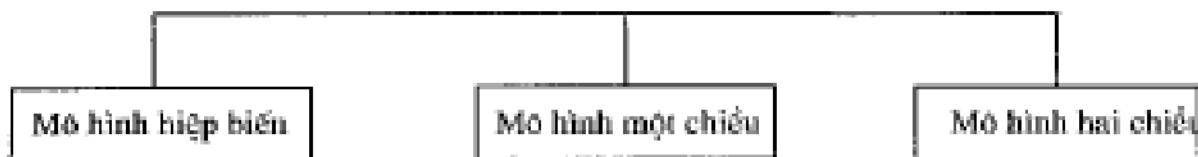
Hiệp biến không tách được cũng được biểu diễn dưới dạng mỗ:

$$r(m,n) = \sigma^2 \sqrt{\alpha_1 m^2 + \alpha_2 n^2} \quad (3.62)$$

Khi  $\alpha_1 = \alpha_2 = \alpha$ ,  $r(m,n)$  trở thành khoảng cách Euclidean và  $r(m,n) = \alpha^2 p^2$ , với:  $p = \exp(-|\alpha|)$ . Điều này lý giải tại sao lại gọi là mô hình mỗ.

Mô hình hiệp biến tách được rất thuận tiện cho việc phân tích các thuật toán xử lý ảnh. Mô hình hiệp biến không tách được là một mô hình tốt hơn, tuy vậy nó không thuận tiện cho việc phân tích. Mô hình hiệp biến rất có ích trong việc biến đổi ảnh, khôi phục và nén ảnh.

Đối ngược với biểu diễn trường ngẫu nhiên dùng kỳ vọng toán học và hiệp biến, một cách khác là coi nó như đầu ra của một hệ thống tuyến tính mà đầu vào là trường ngẫu nhiên với một số tính chất thống kê đã biết (thí dụ như nhiễu trắng đầu vào). Hệ thống tuyến tính biểu diễn bởi phương trình vi phân và hữu ích trong việc phát triển các thuật toán xử lý ánh với hiệp biến được biết đến với tên gọi “*phân tích phổ*”. Hình 3.13 dưới đây tổng kết các mô hình thống kê trong xử lý ánh:



Hình 3.13. Một số mô hình thống kê.

### 3.5.1. Mô hình 1 chiều nhận quả (one dimensional causal model)

Một cách đơn giản để đặc trưng một ảnh là xem xét tín hiệu một chiều xuất hiện ở đầu ra của một lưỡi quét hàng có nghĩa là một chuỗi hàng hay cột. Nếu sự phụ thuộc giữa hàng/cột là không tính đến, hệ thống tuyến tính một chiều tỏ ra rất có ích cho việc mô hình hóa các tín hiệu như vậy.

Giả sử  $u(n)$  là một chuỗi các số thực ngẫu nhiên với trung bình 0 và hiệp biến là  $r(n)$ . Nếu  $U(n)$  được coi như đầu ra của một hệ thống tuyến tính bất biến ổn định  $H(z)$  mà đầu vào là một chuỗi ngẫu nhiên dừng trung bình 0:  $e(n)$ , hàm phân bố rời rạc có dạng:

$$S(z) = H(z)S_e(f)H(z^{-1}) \quad (3.63)$$

với  $S_e(z)$  là hàm phân bố rời rạc của  $e(n)$ .

Chuỗi ngẫu nhiên  $u(n)$  trung bình 0 gọi là một quá trình tự điều chỉnh bậc p khi nó có thể được khởi tạo như đầu ra của hệ thống:

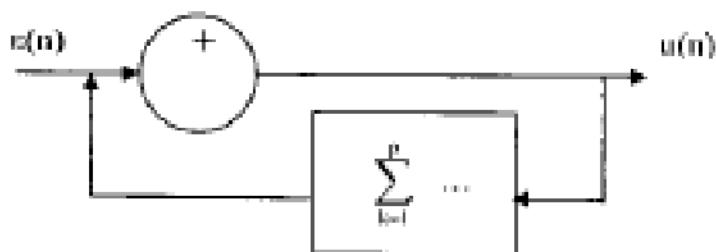
$$u(n) = \sum_{k=1}^p a(k)u(n-k) + e(n) \quad \forall n \quad (3.64)$$

$$E[e(n)] = 0, E[e(n)^2] = \beta^2, E[e(n)u(m)] = 0 \quad m < n$$

$$\text{Gọi } \pi(n) = \sum_{k=1}^p a(k)u(n-k) \quad (3.65)$$

Là dự đoán bình phương trung bình tuyến tính tốt nhất của  $u(n)$  dựa vào tất cả những cái trước song chỉ phụ thuộc duy nhất vào mẫu gần nhất. Nếu  $u(n)$  là chuỗi Gauss, điều đó có nghĩa là AR bậc  $p$  của một quá trình Markov và phương trình 3.64 trở thành:

$$u(n) = \pi(n) + \varepsilon(n) \quad (3.66)$$



Hình 3.14. Mô hình quay lui AR.

Điều đó có nghĩa là mẫu ở thời điểm  $n$  là tổng của các ước lượng dự đoán sai số tối thiểu  $\varepsilon(n)$ . Thí dụ: hiệp biến của lưới quét hàng của một ảnh có thể thu được khi xem xét hiệp biến giữa 2 điểm trên cùng một hàng.

Giải phương trình:

$$\sigma^2 \begin{pmatrix} 1 & p \\ p & 1 \end{pmatrix} \times \begin{pmatrix} a[1] \\ a[2] \end{pmatrix} = \sigma^2 \begin{pmatrix} p \\ p^2 \end{pmatrix}$$

ta sẽ thu được nghiệm  $a[1] = p$ ,  $a[2] = 0$  và  $\beta^2 = \delta^2 (1-p^2)$ . Ta suy ra biểu diễn tương ứng của lưới quét hàng của một ảnh có điểm trung bình  $\mu$  là một mô hình AR bậc nhất:

$$\begin{aligned} x(n) &= px(n-1) + \varepsilon(n) \\ u(n) &= x(n) - \mu \\ r(n) &= \sigma^2(1-p^2) \delta(n) \end{aligned} \quad (3.67)$$

Qua thí dụ này, chúng ta thấy mô hình AR rất hữu ích trong biểu diễn ảnh quét theo hàng. Còn nhiều biến đổi của mô hình AR, bạn đọc quan tâm xem trong Anil.K.Jain [1] như mô hình AR đồng nhất, MA (Moving Average), ARMA, v.v.

### 3.5.2. Mô hình nhân quả hai chiều

Khái niệm nhân quả không mở rộng một cách tự nhiên cho mô hình hai chiều hay nhiều chiều. Kỹ thuật xử lý theo từng dòng áp dụng khá tốt cho mô hình một chiều như đã trình bày ở trên, song lại không áp dụng cho cấu trúc hai chiều khi sự phụ thuộc giữa các

hàng. Bởi vì quan hệ nhân quả không phải là cái chủ yếu trong cấu trúc hai chiều, do vậy ta phải tìm những cấu trúc khác đặc trưng cho mô hình hai hay nhiều chiều. Người ta đưa ra 3 dạng chính tắc: dạng nhân quả, dạng bán nhân quả và dạng không nhân quả.

Mỗi dạng đều có đặc trưng riêng. Thí dụ dạng nhân quả cho những thuật toán dễ qui trong nén dữ liệu bởi phương pháp mã hoá điều xung vi phân (DPCM).

Dạng bán nhân quả là nhân quả theo một chiều song lại không nhân quả theo chiều kia. Trong mô hình này, người ta dùng các thuật toán dễ qui cho chiều nhân quả, còn chiều khác có thể dùng biến đổi dài tương quan đơn vị.

Mô hình không nhân quả dẫn tới các thuật toán dựa vào các biến đổi như biến đổi KL (trình bày ở trên) hay mô hình MVR. Nhiều toán tử xử lý ảnh không gian là không nhân quả như kỹ thuật mặt nạ Gradient sẽ nói tới trong chương 5. Nhìn chung lý thuyết phần này khá phức tạp và phần nào vượt quá khuôn khổ giáo trình.

### Bài tập chương 3

1. Cho ma trận hiệp biến R trong không gian 3 chiều:

$$R = \begin{bmatrix} -3 & -1 & 0 \\ -1 & 3 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

Hãy xác định các trị riêng và vectơ riêng tương ứng trong không gian trực giao 3 chiều.

2. Cho ảnh số:

a)

$$I = \begin{bmatrix} 1 & 0 & 1 & 2 \\ 3 & 1 & 4 & 5 \\ 7 & 0 & 6 & 4 \\ 8 & 1 & 5 & 3 \end{bmatrix}$$

Hãy tính ảnh đầu ra với nhân chập  $H_3$  trong tài liệu  $Y = H_3 \otimes I$

b) Cho nhân chập  $H_2$ :

$$H_2 = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

**Tính tích chập  $H_1 \otimes I$** 

3. Viết thủ tục tính lược đồ xám của một ảnh đã cho và vẽ đồ thị biểu diễn lược đồ. Số mức xám nhập vào theo tham số.
4. Viết thủ tục xây dựng bảng tra LUT của một ảnh và biến đổi ảnh theo LUT.
5. Viết thủ tục thực hiện việc san bằng lược đồ xám của một ảnh dựa vào giải thuật san bằng lược đồ xám.
6. Dựa vào giải thuật lọc trung vị hãy viết một thủ tục thực hiện lọc trung vị cho một ảnh.

**4**

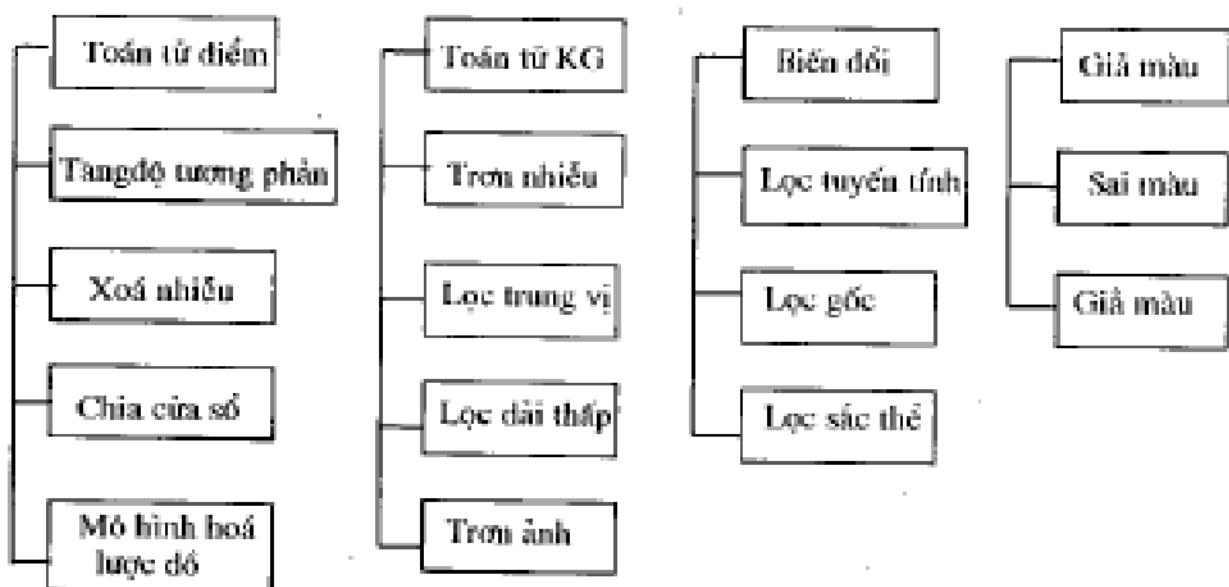
## XỬ LÝ VÀ NÂNG CAO CHẤT LƯỢNG ẢNH

IMAGE ENHANCEMENT

Nâng cao chất lượng ảnh là một bước quan trọng, tạo tiền đề cho xử lý ảnh. Mục đích chính là nhằm làm nổi bật một số đặc tính của ảnh như thay đổi độ tương phản, lọc nhiễu, nổi biến, làm tròn biến ảnh, khuếch đại ảnh, ... . Tăng cường ảnh và khôi phục ảnh là 2 quá trình khác nhau về mục đích. *Tăng cường ảnh* bao gồm một loạt các phương pháp nhằm hoàn thiện trạng thái quan sát của một ảnh. Tập hợp các kỹ thuật này tạo nên giai đoạn tiền xử lý ảnh. Trong khi đó, *khôi phục ảnh* nhằm khôi phục ảnh gần với ảnh thực nhất trước khi nó bị biến dạng do nhiều nguyên nhân khác nhau.

### 4.1. Các kỹ thuật tăng cường ảnh (Image Enhancement)

Nhiệm vụ của tăng cường ảnh không phải là làm tăng lượng thông tin vốn có trong ảnh mà làm nổi bật các đặc trưng đã chọn làm sao để có thể phát hiện tốt hơn, tạo thành quá trình tiền xử lý cho phân tích ảnh.



Hình 4.1. Các kỹ thuật cải thiện ảnh.

Tăng cường ảnh bao gồm: điều khiển mức xám, dãn độ tương phản, giảm nhiễu, làm tròn ảnh, nới suy, phóng đại, nới biên v.v. Các kỹ thuật chủ yếu trong tăng cường ảnh được mô tả qua hình 4.1.

#### 4.1.1. Cải thiện ảnh dùng toán tử điểm

Toán tử điểm là toán tử không bộ nhớ, ở đó một mức xám  $u \in [0, N]$  được ảnh xạ sang một mức xám  $v \in [0, N]$ :  $v = f(u)$  (chuong 3 — phần 3.4). Ứng dụng chính của toán tử điểm là nhằm biến đổi độ tương phản của ảnh. Ảnh xạ  $f$  tùy theo các ứng dụng khác nhau sẽ có dạng khác nhau và được liệt kê trong bảng sau:

##### 1) Tăng độ tương phản

$$f(u) = \begin{cases} \alpha u & \alpha \leq u < a \\ \beta(u - a) + v_s & a \leq u < b \\ \gamma(u - b) + v_b & B \leq u < L \end{cases}$$

Các độ dốc  $\alpha, \beta, \gamma$  xác định độ tương phản tương đối.  $L$  là số mức xám cực đại.

##### 2) Tách nhiễu và phân ngưỡng

$$f(u) = \begin{cases} 0 & 0 \leq u < a \\ \alpha u & a \leq u < b \\ L & u \geq b \end{cases}$$

Khi  $a = b = t$  gọi là phân ngưỡng

##### 3) Biến đổi âm bản

$$f(u) = L - u \quad \text{tạo âm bản}$$

##### 4) Cắt theo mức

$$f(u) = \begin{cases} L & a \leq u \leq b \\ 0 & \text{kết khác} \end{cases}$$

##### 5) Trích chọn bit

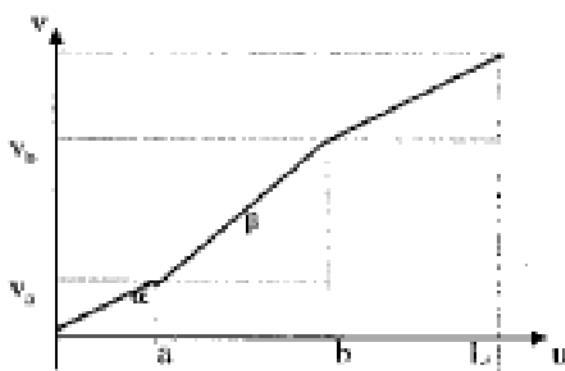
$$f(u) = (i_n - 2i_{n-1})L, \text{ với } i_n = \text{Int}[it/2^{n-1}], n = 1, 2, \dots, B$$

#### 4.1.1.1. Tăng độ tương phản (stretching contrast)

Trước tiên cần làm rõ khái niệm độ tương phản. Ảnh số là tập hợp các điểm, mà mỗi điểm có giá trị độ sáng khác nhau. Ở đây, độ sáng để mắt người dễ cảm nhận ảnh song không phải là quyết định. Thực tế chỉ ra rằng hai đối tượng có cùng độ sáng nhưng đặt trên hai nền khác nhau sẽ cho cảm nhận khác nhau. Như vậy, độ tương phản biểu diễn

sự thay đổi độ sáng của đối tượng so với nền. Một cách nôm na, độ tương phản là độ nổi của điểm ảnh hay vùng ảnh so với nền. Với định nghĩa này, nếu ảnh của ta có độ tương phản kém, ta có thể thay đổi tùy ý theo ý muốn.

Ảnh với độ tương phản thấp có thể do điều kiện sáng không đều hay không đều, hoặc do tính không tuyến tính hay biến động nhỏ của bộ cảm nhận ảnh. Để điều chỉnh lại độ tương phản của ảnh, ta điều chỉnh lại biến độ trên toàn dải hay trên dải có giới hạn bằng cách biến đổi tuyến tính biến độ đầu vào (dùng hàm biến đổi là hàm tuyến tính) hay phi tuyến (hàm mũ hay hàm lôgarít). Khi dùng hàm tuyến tính các độ dốc  $\alpha, \beta, \gamma$  phải chọn *lớn hơn một trong miền cần điều*. Các tham số  $a$  và  $b$  (các cận) có thể chọn khi xem xét lược đồ xám của ảnh.



Hình 4.2. Đồ thị độ tương phản.

Chú ý, nếu dân độ tương phản bằng hàm tuyến tính ta có:

$\alpha = \beta = \gamma = 1$	anh kết quả trùng với ảnh gốc
$\alpha, \beta, \gamma > 1$	dân độ tương phản
$\alpha, \beta, \gamma < 1$	co độ tương phản

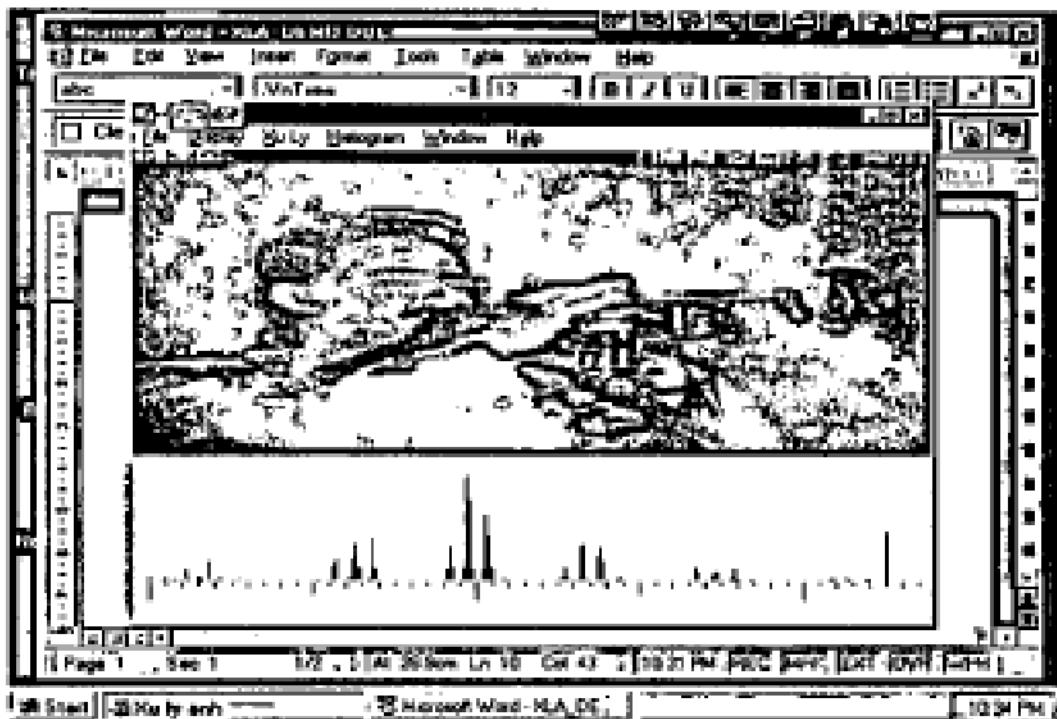
Hàm mũ hay dùng trong dân độ tương phản có dạng:

$$f = (X[m,n])^p$$

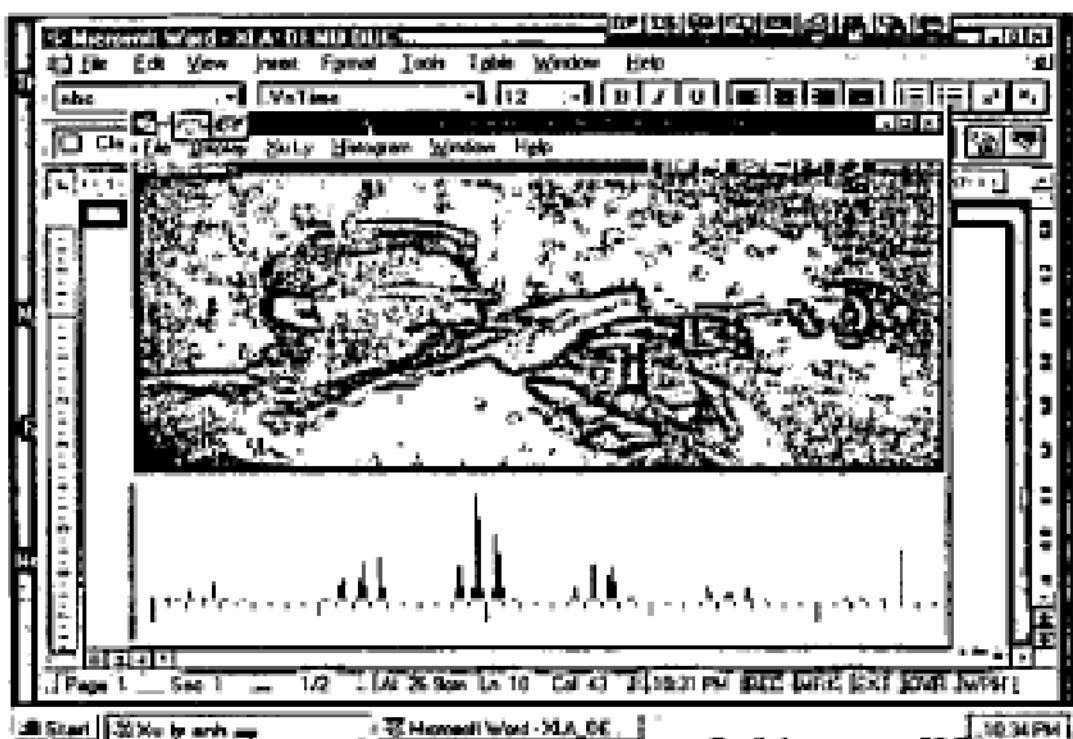
Với các ảnh dạng động nhỏ, p thường chọn bằng 2.

#### 4.1.1.2. Tách nhiễu và phân ngưỡng

Tách nhiễu là trường hợp đặc biệt của dân độ tương phản khi hệ số góc  $\alpha = \gamma = 0$ . Tách nhiễu được ứng dụng một cách hữu hiệu để giảm nhiễu khi biết tín hiệu vào nằm trên khoảng  $[a,b]$ .



A) Ảnh nguồn cùng lược đồ xám. Chỉ số màu cao nhất là 97



b) Ảnh sau khi dàn độ tương phản với  $\alpha = 3$ ,  $\beta = 2$  và  $\gamma = 1$ .

Hình 4.3. Ảnh gốc và ảnh kết quả sau khi dàn.



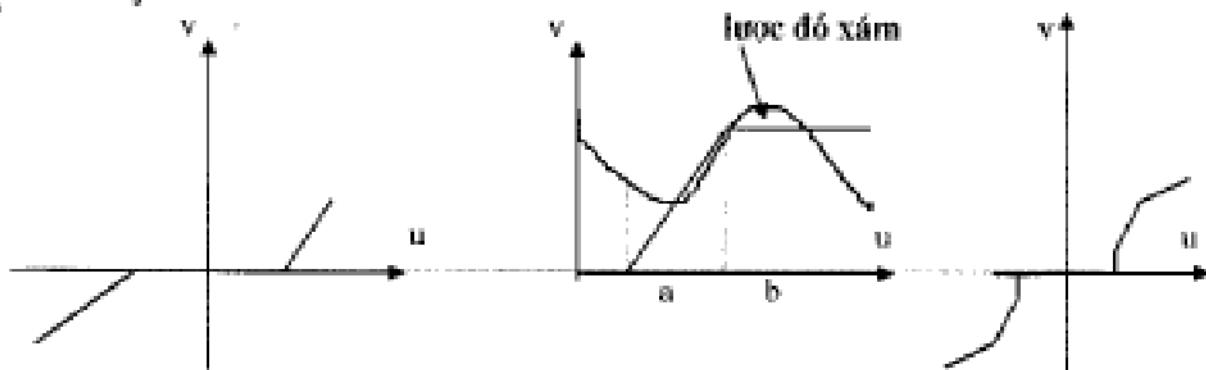
c) ảnh gốc(ảnh vệ tinh — TIFF)



d) ảnh sau khi dán độ tương phản

Hình 4.3. Ảnh gốc và ảnh kết quả sau khi dán.

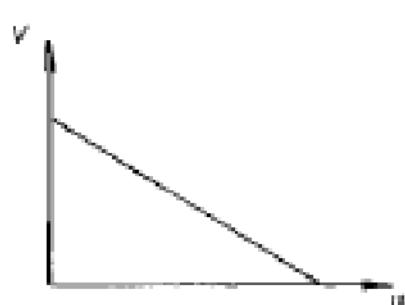
Phản nguồng là trường hợp đặc biệt của tách nhiễu khi  $a = b = \text{const}$  và rõ ràng trong trường hợp này, ảnh đầu ra là ảnh nhị phân (vì chỉ có 2 mức). Phản nguồng hay dùng trong kỹ thuật in ảnh 2 màu vì ảnh gần nhị phân không thể cho ra ảnh nhị phân khi quét ảnh bởi có sự xuất hiện của nhiễu do bộ cảm biến và sự biến đổi của nền. Thí dụ như trường hợp ảnh vân tay.



Hình 4.4. Tách nhiễu và phản nguồng.

#### 4.1.1.3 Biến đổi ám bản (Digital Negative)

Biến đổi ám bản nhận được khi dùng phép biến đổi  $f(u) = 255 - u$ . Biến đổi ám bản rất có ích khi hiện các ảnh y học và trong quá trình tạo các ảnh ám bản.



Hình 4.5. Biến đổi ám bản.

#### 4.1.1.4. Cắt theo mức (Intensity Level Slicing)

Kỹ thuật này dùng 2 phép ảnh xạ khác nhau cho trường hợp có nền và không nền

- Có nền

$$f(u) = \begin{cases} L & \text{nếu } a \leq u \leq b \\ u & \text{khác} \end{cases}$$

- Không nền

$$f(u) = \begin{cases} L & \text{nếu } a \leq u \leq b \\ 0 & \text{khác} \end{cases}$$



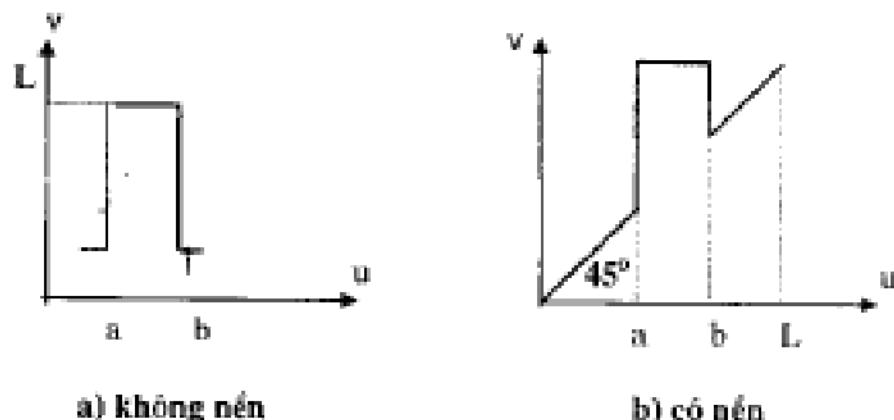
a) Ảnh màu cùng với lược độ xám. Chỉ số màu cao nhất: 243.



b) Ảnh ám bản cùng với lược độ xám (ứng với phép biến đổi  $f(x) = L - x$ ).

Chỉ số màu cao nhất: 12

Hình 4.6. Ảnh gốc và ảnh ám bản.



#### **High 4.7. Kỹ thuật cắt theo mực.**

Biến đổi này cho phép phân đoạn một số mức xám từ phần còn lại của ảnh. Nó hữu dụng khi nhiều đặc tính khác nhau của ảnh nằm trên nhiều miền mức xám khác nhau.

#### **4.1.1.5. Trick chon bit (Bit Extraction)**

Như đã trình bày trên, mỗi điểm ảnh thường được mã hoá trên B bit. Nếu  $B = 8$  ta có  $2^8 = 256$  mức xám (ảnh nhí phản ứng với  $B = 1$ ). Trong các bit mã hoá này, người ta chia làm 2 loại: *bit bậc thấp* và *bit bậc cao*. Với bit bậc cao, độ bảo toàn thông tin cao hơn nhiều so với bit bậc thấp. Các bit bậc thấp thường biểu diễn nhiễu hay nền. Trong kỹ thuật này, ta có:

$$u = k_1 2^{B-1} + k_2 2^{B-2} + \dots + k_{B-1} 2 + k_B$$

Nếu ta muốn trích chọn bit có nghĩa nhất: bit thứ  $n$  và hiện chúng, ta dùng biến đổi:

$$f(u) = \begin{cases} L & \text{nếu } k_n = 1 \\ 0 & \text{khác } 1 \end{cases}$$

và dễ dàng thấy  $k_n = i_n - 2 i_{n-1}$  với  $i_n$  cho ở bảng trên.

#### 4.1.1.6. Trygghet

Trừ ảnh được dùng để tách nhiễu khỏi nền. Người ta quan sát ảnh ở 2 thời điểm khác nhau, so sánh chúng để tìm ra sự khác nhau. Người ta đóng thắng 2 ảnh rồi trừ đi và thu được ảnh mới. Ảnh mới này chính là sự khác nhau. Kỹ thuật này hay được dùng trong dự báo thời tiết, trong y học.

#### **4.1.1.7. Nén dữ liệu**

Đôi khi do dài động của ảnh lớn, việc quan sát ảnh không thuận tiện. Cần phải thu nhỏ dài độ sáng lại mà ta gọi là nén dài độ sáng. Người ta dùng phép biến đổi Joga sau:

$$v(m,n) = c \log_{10}(\delta + u(m,n))$$

với  $c$  là hằng số tỉ lệ,  $\delta$  là rất nhỏ so với  $u(m,n)$ . Thường  $\delta$  chọn cỡ  $10^{-2}$ .

#### 4.1.1.8. Mô hình hoá và biến đổi luồng đố xám

Về ý nghĩa của luồng đố xám và một số phép biến đổi luồng đố đã được trình bày trong chương 3 (phần 3.4). Ở đây, ta xét đến một số biến đổi hay dùng:

- $f(u) = \sum_{x_i=0}^u p_i(x_i)$  (4.1)

với  $p_i(x_i) = \frac{h(x_i)}{\sum_{i=0}^L h(x_i)}$   $i = 0, 1, \dots, L-1$  (4.2)

$h(x_i)$  là luồng đố mức xám  $x_i$ ; có nghĩa là số điểm ảnh có mức xám  $x_i$ . Trong biến đổi này,  $u$  là mức xám đầu vào; còn đầu ra sẽ được lượng hoá đều theo sơ đồ:



Biến đổi này được dùng trong xem bảng luồng đố.

- Ngoài biến đổi như trên, người ta còn dùng một số biến đổi khác. Trong các biến đổi này, mức xám đầu vào  $u$ , trước tiên được biến đổi phi tuyến bởi một trong các hàm sau:

$$f(u) = \frac{\sum_{i=0}^n p_i^{(n)}(x_i)}{\sum_{i=0}^L p_i^{(n)}(x_i)} \quad \text{với } n=2, 3, \dots \quad (4.3)$$

$$f(u) = \log(1+u) \quad u \geq 0 \quad (4.4)$$

$$(u) = u^n \quad u \geq 0, n = 2, 3, \dots \quad (4.5)$$

sau đó đầu ra được lượng hoá đều. Ba phép biến đổi này được dùng trong lượng hoá ảnh.

Nhìn chung, các biến đổi luồng đố nhằm biến đổi luồng đố từ một đường không thuần nhất sang một đường đồng nhất để tiện cho việc phân tích ảnh.

#### 4.1.2. Cải thiện ảnh dùng toán tử không gian

Cải thiện ảnh là làm cho ảnh có chất lượng tốt hơn theo ý đồ sử dụng. Thường là ảnh thu nhận có nhiều cản phải loại bỏ nhiều hay ảnh không sắc nét bị mờ hoặc cần làm rõ các chi tiết như biến. Các toán tử không gian dùng trong kỹ thuật tăng cường ảnh được phân theo nhóm theo công dụng: làm tròn nhiễu, nổi biến. Để làm tròn nhiễu hay tách nhiễu người ta sử dụng các bộ lọc tuyến tính (lọc trung bình, thông thấp) hay lọc phi tuyến (trung vị, giả trung vị, lọc đồng hình). Do bản chất của nhiễu là ứng với tần số cao và cơ sở lý thuyết của lọc là bộ lọc chỉ cho tín hiệu có tần số nào đó thông qua (dài tần bộ lọc). Do vậy để lọc nhiễu ta dùng lọc thông thấp (theo quan điểm tần số không gian) hay lấy tổ hợp tuyến tính để san bằng (lọc trung bình). Để làm nổi cạnh (ứng với tần số cao), người ta dùng các bộ lọc thông cao, Laplace. Chi tiết và các cách áp dụng được trình bày dưới đây.

Trước khi xem xét chi tiết các kỹ thuật áp dụng, cũng hữu ích khi phân biệt một số các loại nhiễu hay can thiệp trong quá trình xử lý ảnh. Thực tế có nhiều loại nhiễu, tuy nhiên người ta thường xem xét 3 loại nhiễu chính: nhiễu cộng, nhiễu nhân và nhiễu xung:

- *Nhiễu cộng*: nhiễu cộng thường phân bố khắp ảnh. Nếu ta gọi ảnh quan sát (ảnh thu được) là  $X_{qs}$ , ảnh gốc là  $X_{ps}$  và nhiễu là  $\eta$ . Ảnh thu được có thể biểu diễn bởi:

$$X_{qs} = X_{ps} + \eta$$

- *Nhiễu nhân*: nhiễu nhân thường phân bố khắp ảnh. Nếu ta gọi ảnh quan sát (ảnh thu được) là  $X_{qs}$ , ảnh gốc là  $X_{ps}$  và nhiễu là  $\eta$ , ảnh thu được có thể biểu diễn bởi:

$$X_{qs} = X_{ps} \times \eta$$

- *Nhiễu xung*: nhiễu xung thường gây đột biến tại một số điểm của ảnh.

##### 4.1.2.1. Làm tròn nhiễu bằng lọc tuyến tính: lọc trung bình và lọc dài thông thấp

Vì có nhiều loại nhiễu can thiệp vào quá trình xử lý ảnh nên cần có nhiều bộ lọc thích hợp. Với nhiễu cộng và nhiễu nhân ta dùng các bộ lọc thông thấp, trung bình và lọc đồng hình (homomorphic); với nhiễu xung ta dùng lọc trung vị, giả trung vị, lọc ngoài (outlier).

###### a) Lọc trung bình không gian

Với lọc trung bình, mỗi điểm ảnh được thay thế bằng trung bình trọng số của các điểm lân cận và được định nghĩa như sau:

$$v(m,n) = \sum_{(k,l) \in W} a(k,l) y(m-k, n-l) \quad (4.6)$$

Nếu trong kỹ thuật lọc trên, ta dùng các trọng số như nhau, phương trình 4.6 trở thành:

$$v(m,n) = \frac{1}{N_w} \sum_{(k,l) \in W} y(m-k, n-l) \quad (4.7)$$

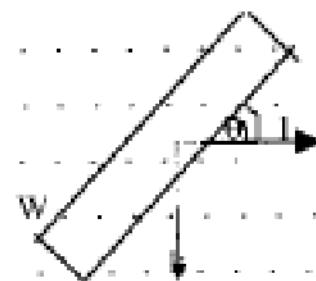
với -  $y(m,n)$  : ảnh đầu vào

-  $v(m,n)$  : ảnh đầu ra

-  $w(m,n)$  : là cửa sổ lọc

-  $a(k,l)$  : là trọng số lọc

với  $a_{k,l} = \frac{1}{N_w}$  và  $N_w$  là số điểm ảnh trong cửa sổ lọc  $W$ .



Hình 4.8.

Lọc trung bình có trọng số chính là thực hiện chập ảnh đầu vào với nhân chập  $H$ . Nhân chập  $H$  trong trường hợp này có dạng:

$$H = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Trong lọc trung bình, đôi khi người ta ưu tiên cho các hướng để bảo vệ biên của ảnh khỏi bị mờ đi do làm tròn ảnh. Các kiểu mặt nạ như đã liệt kê trong chương trước được sử dụng tùy theo các trường hợp khác nhau. Các bộ lọc trên là bộ lọc tuyến tính theo nghĩa là điểm ảnh ở tam giác của sổ sẽ được thay bởi tổng của các điểm lân cận chập với mặt nạ.

Giả sử ảnh đầu vào biểu diễn bởi ma trận  $I$ :

$$I = \begin{bmatrix} 4 & 7 & 2 & 7 & 1 \\ 5 & 7 & 1 & 7 & 1 \\ 6 & 6 & 1 & 8 & 3 \\ 5 & 7 & 5 & 7 & 1 \\ 5 & 7 & 6 & 1 & 2 \end{bmatrix}$$

ảnh số thu được bởi lọc trung bình  $Y = H \otimes I$  có dạng:

$$Y = \frac{1}{9} \begin{bmatrix} 23 & 26 & 31 & 19 & 16 \\ 35 & 39 & 46 & 31 & 27 \\ 36 & 43 & 49 & 34 & 27 \\ 36 & 48 & 48 & 34 & 22 \\ 24 & 35 & 33 & 22 & 11 \end{bmatrix}$$

Một bộ lọc trung bình không gian khác cũng hay được sử dụng và phương trình của bộ lọc có dạng:

$$Y[m,n] = \frac{1}{2} \left[ X[m,n] + \frac{1}{4} (X[m-1,n] + X[m+1,n] + X[m,n-1] + X[m,n+1]) \right]$$

Ở đây, nhân chập  $H$  là nhân chập  $2*2$  và mỗi điểm ảnh kết quả có giá trị bằng trung bình cộng của nó với trung bình cộng của 4 lân cận (4 lân cận gần nhất).

Lọc trung bình trọng số là một trường hợp riêng của lọc thông thấp.

#### b) Lọc thông thấp

Lọc thông thấp thường được sử dụng để làm trơn nhiễu. Về nguyên lý giống như đã trình bày trên. Trong kỹ thuật này người ta hay dùng một số nhân chập sau:

$$H_0 = \frac{1}{8} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$H_b = \frac{1}{(b+2)^2} \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}$$

Ta dễ dàng thấy khi  $b=1$ ,  $H_b$  chính là nhân chập  $H_1$  (lọc trung bình); còn khi  $b=2$ ,  $H_b$  chính là nhân chập  $H_3$  trong phần trước (mục 3.2 chương 3). Để hiểu rõ hơn bản chất khử nhiễu cộng của các bộ lọc này, ta viết lại phương trình thu nhận ảnh dưới dạng:

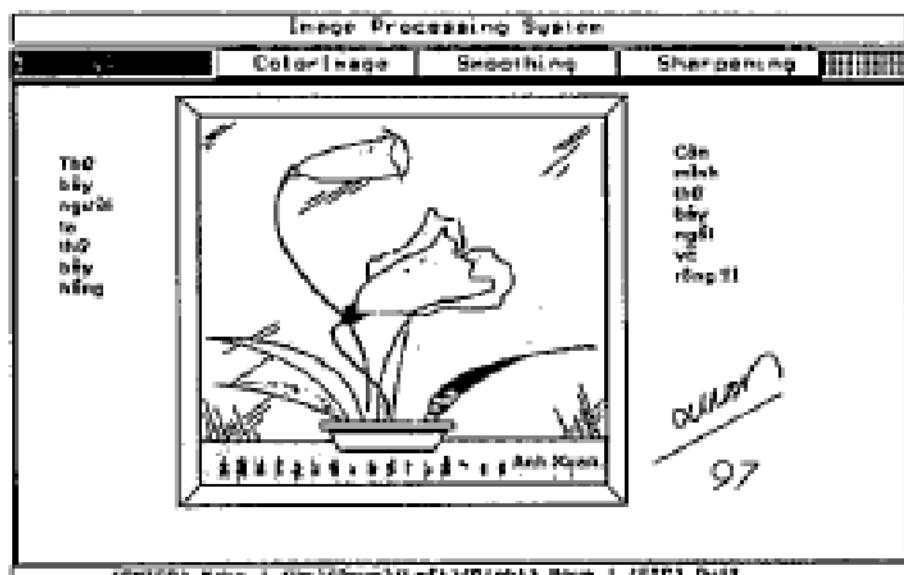
$$X_p[m,n] = X_{ps}[m,n] + \eta[m,n]$$

trong đó  $\eta[m,n]$  là nhiễu cộng có phương sai  $\sigma_\eta^2$ . Như vậy, theo cách tính của lọc trung bình ta có:

$$Y[m,n] = \frac{1}{N_s} \sum_{k,l \in W} x_{ps}(m-k, n-l) + \eta[\bar{m}, \bar{n}] \quad (4.8)$$

$$\text{hay } Y[m,n] = \frac{1}{N_s} \sum_{k,l \in W} x_{s\omega}(m-k, n-l) + \frac{\sigma_s^2}{N_s} \quad (4.9)$$

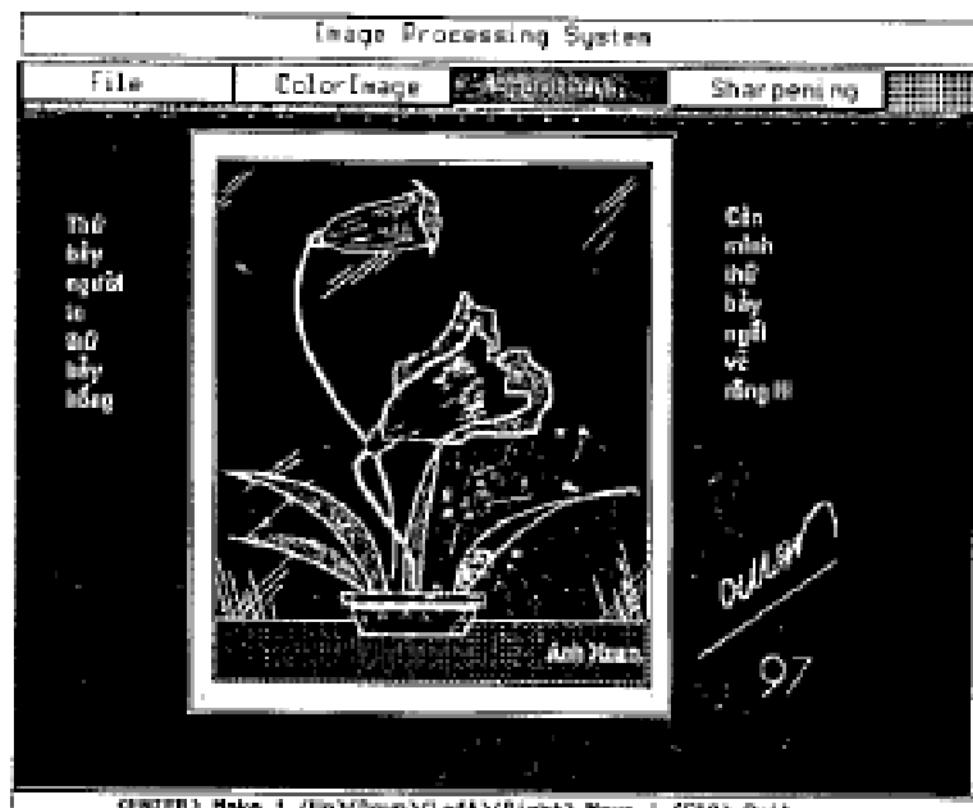
Như vậy nhiễu công trong ảnh đã giảm đi  $N_s$ , lần. Hình 4.9 minh họa tác dụng cải thiện ảnh bằng lọc thông thấp.



a) Ảnh gốc (chuyển đổi từ ảnh màu sang ảnh mức xám).



b) Ảnh qua lọc trung bình.



c) Ảnh thu được qua lọc thông thấp.

Hình 4.9 Ảnh gốc và ảnh kết quả.

#### c) Lọc đồng hình (Homomorphic filter)

Kỹ thuật lọc này hiệu quả với ảnh có nhiều nhân. Thực tế là ảnh quan sát được gồm ảnh gốc nhân với một hệ số nhiễu. Gọi  $\bar{X}(m,n)$  là ảnh thu được,  $X(m,n)$  là ảnh gốc và  $\eta(m,n)$  là nhiễu. Như vậy:

$$X(m,n) = \bar{X}(m,n) + \eta(m,n)$$

Lọc đồng hình thực hiện lấy logarit của ảnh quan sát. Do vậy ta có kết quả sau:

$$\log(X(m,n)) = \log(\bar{X}(m,n)) + \log(\eta(m,n))$$

Rõ ràng là nhiễu nhân có trong ảnh sẽ bị giảm. Sau quá trình lọc tuyến tính ta lại chuyển về ảnh cũ bằng phép biến đổi hàm e mũ. Ảnh thu được qua lọc đồng hình sẽ tốt hơn ảnh gốc.

#### 4.1.2.2. Làm tròn nhiễu bằng lọc phi tuyến

Các bộ lọc phi tuyến cũng hay được dùng trong tăng cường ảnh. Trong kỹ thuật này người ta dùng bộ lọc trung vị, giá trung vị, lọc ngoài. Với lọc trung vị, điểm ảnh đầu

vào sẽ được thay thế bởi trung vị các điểm ảnh. Còn lọc giả trung vị sẽ dùng trung bình cộng của 2 giá trị "trung vị" (trung bình cộng của max và min).



Hình 4.9. d) Ảnh qua bằng lọc Homomorphic.

a) Lọc trung vị.

Nhắc lại rằng khái niệm "trung vị" đã nêu trong chương 3 và được viết:

$$v(m,n) = \text{Trungvi}(y(m-k, n-l)) \text{ với } (k,l) \in W \quad (4.8)$$

Kỹ thuật này đòi hỏi giá trị các điểm ảnh trong cửa sổ phải xếp theo thứ tự tăng hay giảm dần so với giá trị trung vị. Kích thước cửa sổ thường được chọn sao cho số điểm ảnh trong cửa sổ là lẻ. Các cửa sổ hay dùng là cửa sổ  $3 \times 3$ ,  $5 \times 5$  hay  $7 \times 7$ . Thí dụ:

Nếu  $y(m) = \{2, 3, 8, 4, 2\}$  và cửa sổ  $W = (-1, 0, 1)$ , ảnh kết quả thu được sau lọc trung vị sẽ là  $v(m) = (2, 3, 4, 4, 2)$ .

Thực vậy: mỗi lần ta so sánh một dãy 3 điểm ảnh đầu vào với trung vị, không kể điểm biên. Do đó:

$$v[0] = 2 <\text{giá trị biên}>$$

$$v[1] = \text{Trungvi}(2,3,8) = 3$$

$$v[2] = \text{Trungvi}(3,8,4) = 4$$

$$v[3] = \text{Trungvi}(8,4,2) = 4$$

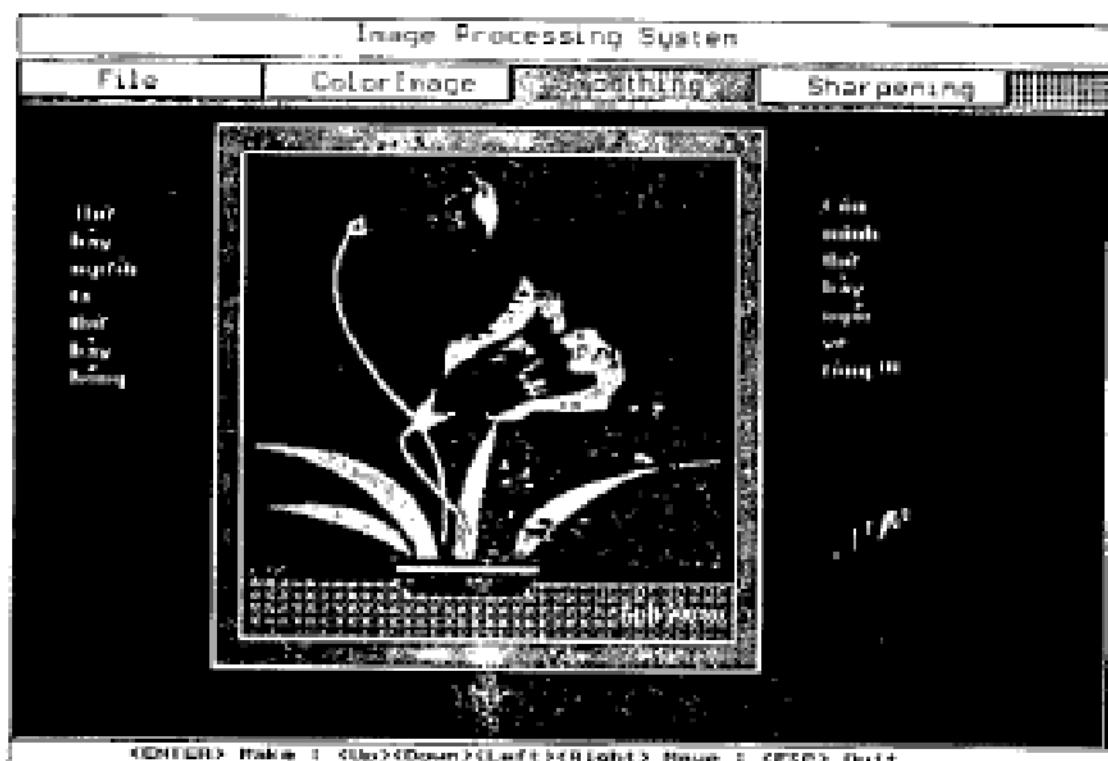
$$v[4] = 2 <\text{giá trị biên}>$$

Tính chất của lọc trung vị:

Lọc trung vị là phi tuyến vì:

$$\text{Trungvi}((x(m)+y(m))) \neq \text{Trungvi}(x(m)) + \text{Trungvi}(y(m)).$$

- Hữu ích cho việc loại bỏ các điểm ảnh huy các hàng mà vẫn bảo toàn độ phân giải.
- Hiệu quả giảm khi số điểm nhiễu trong cùa số lớn hơn hay bằng một nửa số điểm trong cùa số. Điều này dễ giải thích vì trung vị là  $(N_w + 1)/2$  giá trị lớn nhất nếu  $N_w$  lẻ. Lọc trung vị cho trường hợp 2 chiều coi như lọc trung vị tách được theo từng chiều, có nghĩa là người ta tiến hành lọc trung vị cho cột tiếp theo cho hàng.



Hình 4.10. Ảnh thu được qua lọc trung vị với ảnh gốc trong 4.9a.

#### b) Lọc ngoài (Outlier Filter)

Giả thiết rằng có một mức ngưỡng nào đó cho các mức nhiễu (có thể dựa vào lược đồ xám). Tiến hành so sánh giá trị của một điểm ảnh với trung bình số học 8 lân cận của nó. Nếu sự sai lệch này lớn hơn ngưỡng, điểm ảnh này được coi như nhiễu. Trong trường hợp này ta thay thế giá trị của điểm ảnh bằng giá trị trung bình 8 lân cận vừa tính được.

Bộ lọc ngoài có thể biểu diễn bằng công thức sau:

$$Y(m,n) = \begin{cases} \alpha(w) & \text{nếu } |u(m,n) - \alpha(w)| \leq \delta \\ u(m,n) & \text{nếu khác} \end{cases}$$

Với  $\alpha(w)$  là trung bình cộng các điểm trong lân cận  $w$ ,  $\delta$  là ngưỡng ngoài.

Các cửa sổ tính toán thường là  $3 \times 3$ . Tuy nhiên cửa sổ có thể mở rộng đến  $5 \times 5$  hay  $7 \times 7$  để đảm bảo tính tương quan giữa các điểm ảnh. Vấn đề quan trọng là xác định ngưỡng để loại nhiễu mà vẫn không làm mất thông tin.

#### 4.1.2.3. Mặt nạ gờ sai phân và làm nhẵn (Unsharp Masking and Crispering)

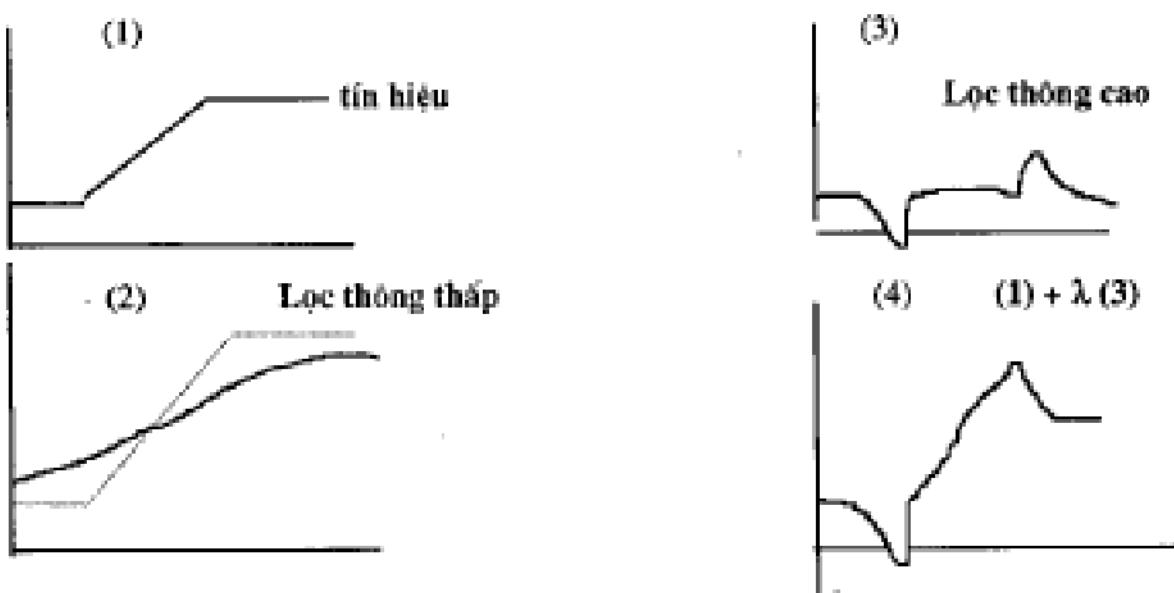
Mặt nạ gờ sai phân dùng khá phổ biến trong công nghệ in ảnh để làm đẹp ảnh. Với kỹ thuật này, tín hiệu đầu ra thu được bằng tín hiệu ra của bộ lọc gradient hay lọc dài cao bổ sung thêm dấu vào:

$$v(m,n) = u(m,n) + \lambda g(m,n) \quad (4.9)$$

với  $\lambda > 0$ ,  $g(m,n)$  là gradient tại điểm  $(m,n)$ . Hàm gradient dùng là hàm Laplace(sẽ trình bày trong chương 5)

$$g(m,n) = u(m,n) - \{u(m-1,n) + u(m+1,n) + u(m,n+1)\}/2 \quad (4.10)$$

Đây chính là mặt nạ chữ thập đã nói trong chương 3.



#### 4.1.2.4 Lọc thông thấp, thông cao và lọc dài thông

Toán tử trung bình không gian nói tới trong mục 4.1.2.1 là lọc thông thấp. Nếu

$h_{LP}(m,n)$  biểu diễn bộ lọc thông thấp FIR (Finite Impulse Response) thì bộ lọc thông cao  $h_{HP}(m,n)$  có thể được định nghĩa:

$$h_{HP}(m,n) = \delta(m,n) \cdot h_{LP}(m,n) \quad (4.11)$$

Như vậy, bộ lọc thông cao có thể cài đặt một cách đơn giản như trên hình 4.12.

Bộ lọc dài thông có thể định nghĩa như sau:  $h_{HP} = h_{LP}(m,n) - h_{LZ}(m,n)$  với  $h_{LP}$ ,  $h_{LZ}$  là các bộ lọc thông thấp.



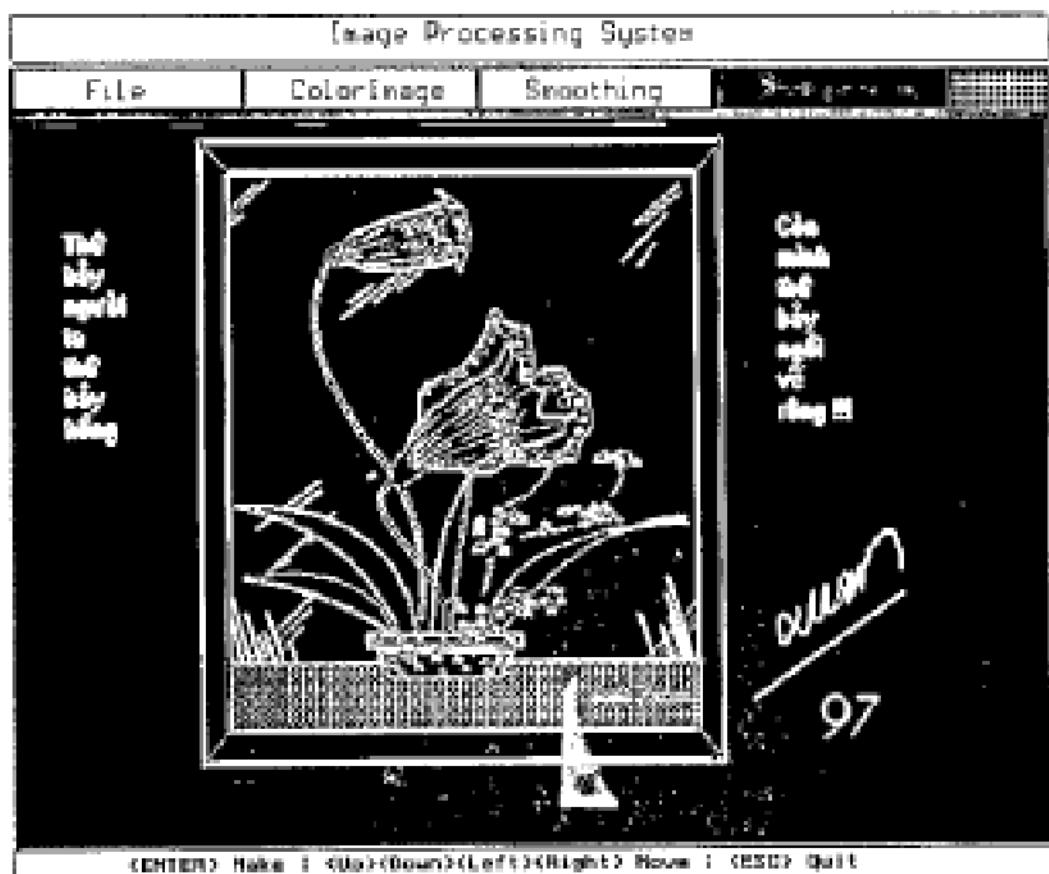
Hình 4.12. Sơ đồ bộ lọc thông cao.

Bộ lọc thông thấp thường dùng làm tròn nhiễu và nội suy. Bộ lọc thông cao dùng trong trích chọn biên và làm tròn ảnh, còn bộ lọc dài thông có hiệu quả làm nỗi cạnh. Vẽ biên sẽ được trình bày kỹ trong chương 5. Tuy nhiên, dễ dàng nhận thấy rằng biên là điểm có độ biến thiên nhanh về giá trị mức xám. Theo quan điểm về tần số tín hiệu, như vậy các điểm biên ứng với các thành phần tần số cao. Do vậy, ta có thể dùng bộ lọc thông cao để cải thiện: lọc các thành phần tần số thấp và chỉ giữ lại thành phần tần số cao. Vì thế, lọc thông cao thường được dùng làm tròn biên trước khi tiến hành các thao tác với biên ảnh. Dưới đây là một số matrix dùng trong lọc thông cao:

$$(1) \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & 1 \end{pmatrix} \quad (2) \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad (3) \begin{pmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

Hình 4.13. Một số nhân chập trong lọc thông cao.

Các nhân chập thông cao có đặc tính chung là tổng các hệ số của bộ lọc bằng 1. Nguyên nhân chính là ngăn cản sự tăng quá giới hạn của các giá trị mức xám (các giá trị điểm ảnh vẫn giữ được giá trị của nó một cách gần đúng không thay đổi quá nhiều với giá trị thực).



Hình 4.14. Ảnh qua lọc thông cao (ảnh gốc hình 4.9a).

#### 4.1.2.5. Khuếch đại và nội suy ảnh

Có nhiều ứng dụng cần thiếp phái phóng đại một vùng của ảnh. Có nghĩa là lấy một vùng của ảnh đã cho và cho hiện lên như một ảnh lớn. Có 2 phương pháp được dùng là lặp (Replication) và nội suy tuyến tính (linear interpolation).

##### Phương pháp lặp

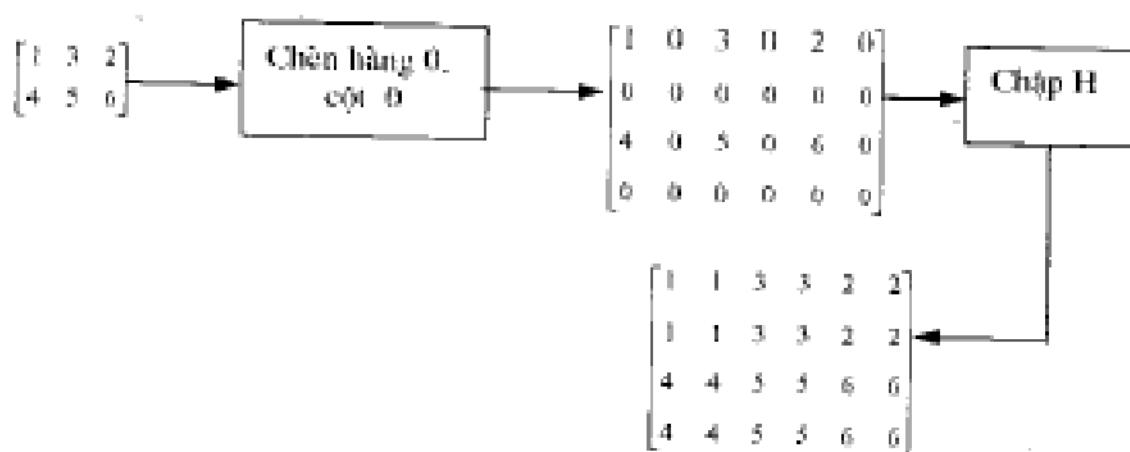
Người ta lấy một vùng của ảnh kích thước  $M \times N$  và quét theo hàng. Mỗi điểm ảnh nằm trên đường quét sẽ được lặp lại 1 lần và hàng quét cũng được lặp lại 1 lần nữa. Như vậy ta sẽ thu được ảnh với kích thước  $2N \times 2N$ . Điều này tương đương với chèn thêm một hàng 0 và một cột 0 rồi chụp với mặt nạ  $H$ .

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

Kết quả thu được  $v(m,n) = u(k,l)$  với  $k = \lfloor m/2 \rfloor$  và  $l = \lfloor n/2 \rfloor$  (4.13)

Ở đây phép toán  $\lfloor \cdot \rfloor$  là phép toán lấy phần nguyên của một số.

Hình 4.15 dưới đây minh họa nội suy theo phương pháp lấp:



Hình 4.15. Khuếch đại bột kíp 2x2.

### Phương pháp nội suy tuyến tính

Trước tiên, hàng được đặt vào giữa các điểm ảnh theo hàng. Tiếp sau, mỗi điểm ảnh dọc theo cột được nội suy theo đường thẳng. Thí dụ với khuếch đại  $2 \times 2$ , nội suy tuyến tính theo hàng sẽ tính theo công thức:

$$\begin{aligned} v_i(m,n) &= u(m,n) \\ v_i(m,2n+1) &= u(m,n) + u(m,n+1) \end{aligned} \quad (4.14)$$

với  $0 \leq m \leq M-1$ ,  $0 \leq n \leq N-1$ ,

và nội suy tuyến tính của kết quả trên theo cột:

$$\begin{aligned} v_j(2m,n) &= v_i(m,n) \\ v_j(2m+1,n) &= v_i(m,n) + v_i(m+1,n) \end{aligned} \quad (4.15)$$

với  $0 \leq m \leq M-1$ ,  $0 \leq n \leq N-1$ .

Nếu dùng mặt nạ:

$$H = \begin{pmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{pmatrix}$$

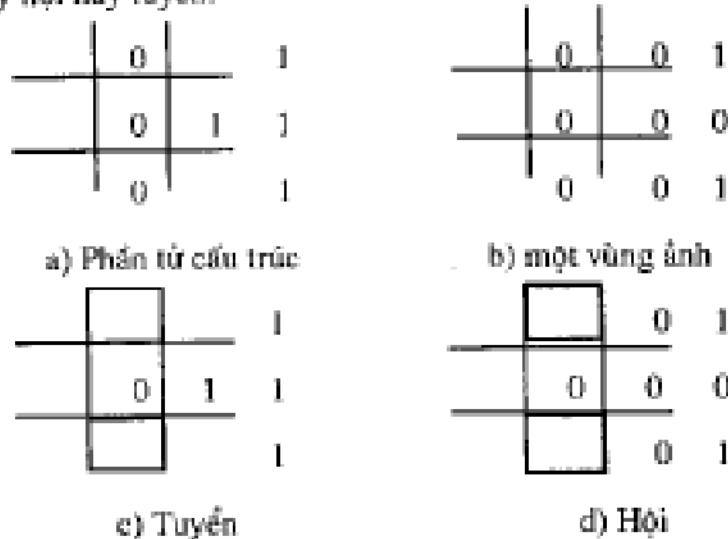
ta cũng thu được kết quả trên.

Nội suy với bậc cao hơn cũng có thể áp dụng cách trên. Thí dụ, nội suy với bậc p (p nguyên), ta chèn p hàng các số 0, rồi p cột các số 0. Cuối cùng, tiến hành nhân chập p lần ảnh với mặt nạ H ở trên [1].

#### 4.1.3. Một số kỹ thuật cải thiện ảnh nhị phân

Với ảnh nhị phân, mức xám chỉ có 2 giá trị là 0 hay 1. Do vậy, ta coi một phần tử ảnh như một phần tử logic và có thể áp dụng các toán tử hình học (morphology operators) dựa trên khái niệm biến đổi hình học của một ảnh bởi một phần tử cấu trúc (structural element).

Phần tử cấu trúc là một mặt nạ dạng bất kỳ mà các phần tử của nó tạo nên một môtíp. Người ta tiến hành rê mặt nạ đi khắp ảnh và tính giá trị điểm ảnh bởi các điểm lân cận với môtíp của mặt nạ theo cách lấy hội hay lấy tuyển. Hình 4.16 dưới đây, chỉ ra một phần tử cấu trúc và cách lấy hội hay tuyển:



Hình 4.16. Cải thiện ảnh nhị phân.

Dựa vào nguyên tắc trên, người ta sử dụng 2 kỹ thuật: dãn ảnh (dilatation) và co ảnh (erosion).

##### 4.1.3.1. Dãn ảnh

Dãn ảnh nhằm loại bỏ điểm đen bị vây bởi các điểm trắng. Trong kỹ thuật này, một cửa sổ  $N+1 \times N+1$  được rê đi khắp ảnh và thực hiện đổi sánh một pixel của ảnh với  $(N+1)^2 - 1$  điểm lân cận (không tính điểm ở tâm). Phép đổi sánh ở đây thực hiện bởi phép tuyển logic. Thuật toán biến đổi được tóm tắt như sau:

For all pixels  $I(x,y)$  do

Begin

```

    . Tính  $F_{OR}(x,y)$  {tính or lô gíc}
    - if  $F_{OR}(x,y)$       then  $ImaOut(x,y) \leftarrow 1$ 
    else  $ImaOut(x,y) \leftarrow ImIn(x,y)$ 

End

```

#### 4.1.3.2. Co ảnh

Co ảnh là thao tác đổi ngẫu của dân ảnh nhằm loại bỏ điểm trắng bị vẩy bởi các điểm đen. Trong kỹ thuật này, một cửa sổ  $(N+1) \times (N+1)$  được rẽ đi khắp ảnh và thực hiện sánh một pixel của ảnh với  $(N+1)^2 - 1$  điểm lân cận. Sánh ở đây thực hiện bởi phép hối logic. Thuật toán biến đổi được tóm tắt như sau:

```

For all pixels  $I(x,y)$  do
    Begin
        . Tính  $F_{AND}(x,y)$  {Tính và lô gíc}
        - if  $F_{AND}(x,y)$       then  $ImaOut(x,y) \leftarrow 1$ 
        else  $ImaOut(x,y) \leftarrow ImIn(x,y)$ 
    End

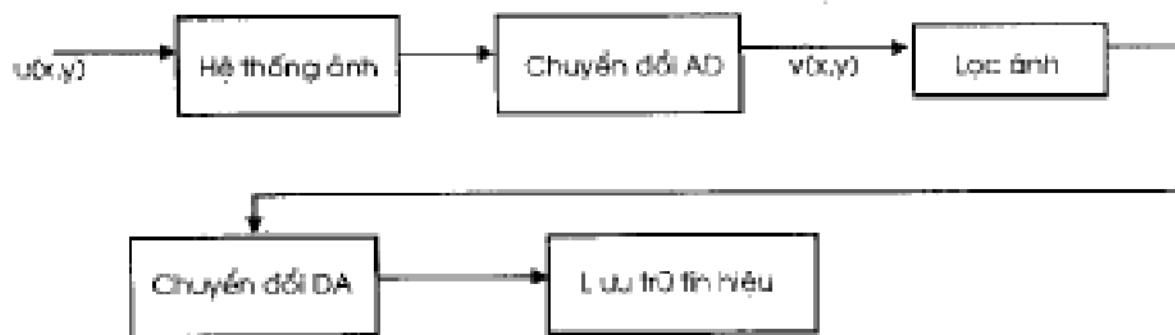
```

*Áp dụng:* Người ta thường vận dụng kỹ thuật này cho các ảnh nhị phân như vân tay, chữ viết. Để không làm ảnh hưởng đến kích thước của đối tượng trong ảnh, người ta tiến hành n lán dâm và n lán eo.

## 4.2. KHỎI PHỤC ẢNH (IMAGE RESTORATION)

Khỏi phục ảnh để cập tới các kỹ thuật loại bỏ hay tối thiểu hoá các ảnh hưởng của môi trường bên ngoài hay các hệ thống thu nhận, phát hiện và lưu trữ ảnh đến ảnh thu nhận được. Ở đây, ta có thể liệt kê nguyên nhân các biến dạng (degradations): do nhiều bộ cảm nhận tín hiệu, ảnh mờ do camera, nhiễu ngẫu nhiên của khí quyển, v.v. Khỏi phục ảnh bao gồm nhiều quá trình như: lọc ảnh, khử nhiễu nhằm làm giảm các biến dạng để có thể khỏi phục lại ảnh gần giống ảnh gốc tùy theo các nguyên nhân gây ra biến dạng. Một hệ thống khỏi phục ảnh số có thể minh họa như hình 4.17.

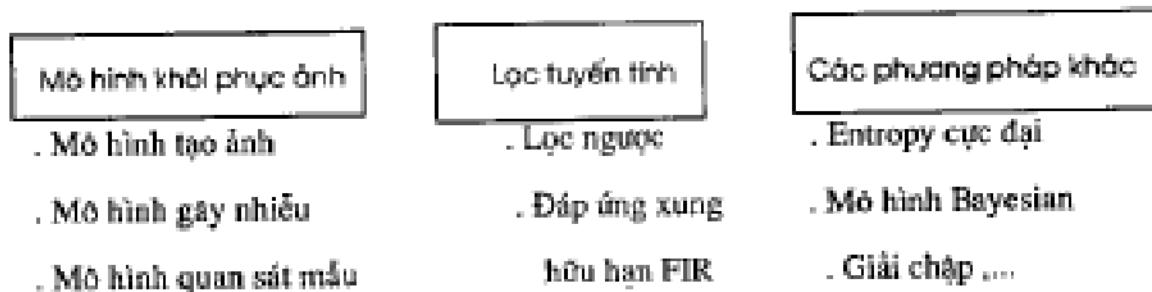
Về nguyên tắc, khỏi phục ảnh nhằm xác định mô hình toán học của quá trình đã gây ra biến dạng, tiếp theo là dùng ảnh xạ ngược để xác định lại ảnh. Việc xác định mô hình có thể thực hiện theo 2 hướng: trước và sau.



Hình 4.17. Hệ thống khôi phục ảnh số.

Theo hướng thứ nhất, một mô hình sẽ được xây dựng từ các ảnh kiểm nghiệm để xác định đáp ứng xung của hệ thống nhiễu.

Theo hướng thứ hai, người ta thực hiện các phép đo trên ảnh. Nói chung là mô hình không biết trước. Các mô hình toán học dùng cho cả hai phương pháp là rất phức tạp.



Hình 4.18. Các kỹ thuật khôi phục ảnh.

#### 4.2.1. Các mô hình quan sát và tạo ảnh

Như đã nêu trên, quá trình gây ra biến dạng ảnh gốc phụ thuộc vào hệ thống quan sát và tạo ảnh. Do vậy, trước hết ta cần xem ảnh quan sát được biểu diễn thế nào, trên cơ sở đó mô hình hoá nhiễu sinh ra. Tiếp theo là dùng biến đổi ngược - chính là lọc ngược để khử nhiễu và thu lấy ảnh gốc. Đó là cơ sở lý thuyết của kỹ thuật khôi phục ảnh.

Lưu ý rằng đây là quá trình ngược: Từ tín hiệu quan sát được gồm tín hiệu vào (ảnh gốc) và các biến dạng (nhiễu). Nếu biết tín hiệu ra thường là ảnh thu nhận được qua hệ thống ảnh (xem chương 2), biết các loại tác động (phụ thuộc vào hệ thống và thiết bị) ta suy ra ảnh gốc.

Nếu gọi:

- $v(x,y)$  là ảnh thu nhận được;

- $\eta(x,y)$  là nhiễu,
- $w(x,y)$  ảnh gốc chưa biết,

-  $f, g$  là các biến đổi nói chung là phi tuyến đặc trưng cho cơ chế phát hiện và lưu ảnh, ta có mô hình sau:

$$v(x,y) = g[w(x,y)] + \eta(x,y) \quad (4.16)$$

$$w(x,y) = \int \int_{-\infty}^{\infty} h(x,y; x',y') u(x',y') dx' dy' \quad (4.17)$$

$$\eta(x,y) = f[g(w(x,y))] \eta_1(x,y) + \eta_2(x,y) \quad (4.18)$$

với:

- $h(x,y)$  là đáp ứng xung tại điểm  $(x,y)$  như đã nêu trong chương 3,
- $w(x,y)$  là tín hiệu đầu ra của hệ thống tuyến tính với đáp ứng xung  $h$ .

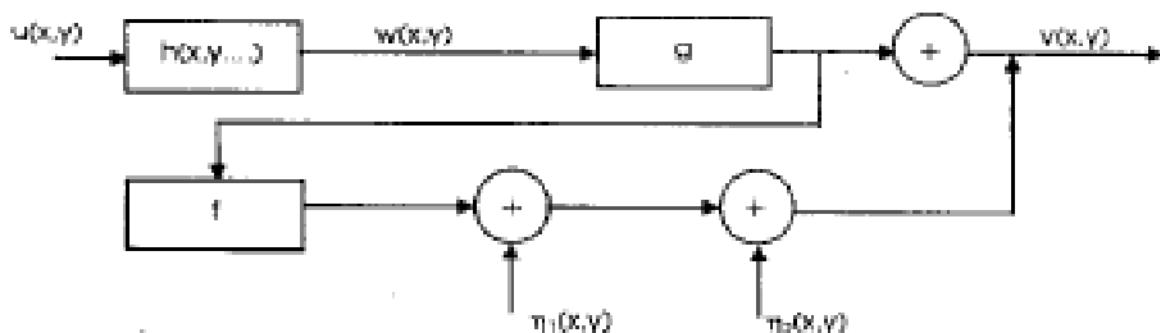
Nhiễu gồm 2 thành phần:

- Thành phần nhiễu phụ thuộc kiểu thiết bị quan sát và tạo ảnh  $\eta_1(x,y)$ ,
- Thành phần nhiễu ngẫu nhiên độc lập  $\eta_2(x,y)$ .

Mô hình quan sát ảnh trên được thể hiện trên hình 4-19.

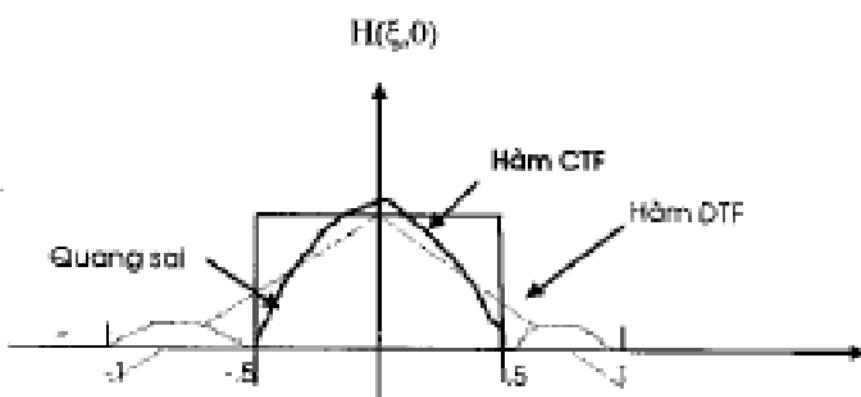
Tùy theo hệ thống, người ta có thể liệt kê một số biến dạng trong quá trình thu nhận ảnh:

- Sự vận pha trong hàm truyền CTF (Coherent Transfer Function) và gọi là *quang sai* (aberration).



Hình 4.19. Mô hình hệ thống quan sát ảnh.

Hình 4.20 dưới đây cho ta thấy sự quang sai của một hệ thống quang học với ống kính vuông.



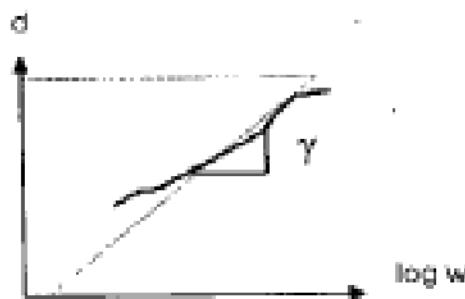
Hình 4.20. Sự biến dạng do nhiễu loạn và quang sai.

- Rung động mờ xảy ra khi có rung động tương đối giữa đối tượng và ống kính thu trong quá trình thu nhận ảnh.
- Sự nhiễu loạn ngẫu nhiên của môi trường xung quanh đối tượng và hệ thống ảnh (đối tượng ảnh thiên văn).

Chúng ta biết rằng, sự đáp ứng của hệ thống phát hiện và lưu ảnh thường là không tuyến tính. Trong phim ảnh, máy quét ảnh hay thiết bị hiện ảnh, sự đáp ứng được biểu diễn bởi công thức :

$$g = \alpha w^\beta \quad (4.19)$$

trong đó  $\alpha, \beta$  là các hằng số phụ thuộc thiết bị;  $w$  là biến đầu vào. Thí dụ trong trường hợp máy ảnh, người ta hay dùng mô hình  $d = \gamma \log w - d_0$ . (4.20)



Hình 4.21. Mô hình đáp ứng tín hiệu ảnh

với  $\gamma$  là hệ số phim,  $w$  biểu diễn độ sáng tối và  $d$  gọi là mật độ quang học.

#### Mô hình nhiễu

Phương trình 4-18 cho ta một mô hình chung của nhiễu xuất hiện trong nhiễu tinh huống. Tuy nhiên, trong một số hệ thống ta có thể biểu diễn nó một cách tường minh hơn. Thí dụ, với một hệ thống quang điện, nhiễu trong chùm tia điện tử thường được biểu diễn bởi:

$$\eta(x,y) = \sqrt{g(x,y)} \eta_1(x,y) + \eta_2(x,y) \quad (4.21)$$

trong đó:  $g$  cho bởi 4-19;  $\eta_1$  và  $\eta_2$  là nhiễu trắng Gauss độc lập tương hỗ với trung bình 0.

Thành phần nhiễu phụ thuộc thiết bị  $\eta_1$  tăng lên là do quá trình phát hiện và lưu ảnh kéo theo sự truyền điện tử ngẫu nhiên. Sự truyền điện tử ngẫu nhiên này có thể biểu diễn bằng phân bố Poisson có trung bình  $g$ . Trong một số trường hợp phân bố này tiệm cận đến phân bố Gauss. Vì phân bố Poisson có trung bình và độ sai lệch là như nhau, do đó thành phần phụ thuộc có phương sai là  $\sqrt{g}$  nếu  $\eta_1$  có độ sai lệch là đơn vị. Thành phần độc lập  $\eta_2$  biểu diễn nhiễu do nhiệt và có thể mô hình hóa theo kiểu nhiễu trắng. Trong một số hệ thống không có nhiễu do nhiệt như hệ thống phim, mô hình nhiễu có thể viết:

$$\eta(x,y) = \sqrt{g(x,y)} \eta_1(x,y) \quad (4.22)$$

Một mô hình khác dành cho nhiễu hạt trong phim là:

$$\eta(x,y) = \varepsilon(g(x,y))^2 \eta_1(x,y) \quad (4.23)$$

với  $\varepsilon$  là hệ số chuẩn hóa,  $\varepsilon \in [1/3, 1/2]$ .

Nói chung, thành phần nhiễu phụ thuộc  $\eta_1(x,y)$  gây rất nhiều khó khăn cho các thuật toán khôi phục ảnh. Do vậy người ta hay dùng trung bình không gian  $\mu_w$  thay cho  $w$  trong  $f[g(x,y)]$  và 4-18 trở thành:

$$\eta(x,y) = f[g(\mu_w)]\eta_1(x,y) + \eta_2(x,y) \quad (4.24)$$

và  $\eta(x,y)$  trở thành mô hình nhiễu trắng Gauss. Nếu sự phát hiện ảnh thực hiện trên một miền tuyến tính với thiết bị quang điện, mô hình quan sát tuyến tính có dạng:

$$v(x,y) = w(x,y) + \sqrt{\mu_w} \eta_1(x,y) + \eta_2(x,y) \quad (4.25)$$

với máy ảnh  $(\gamma=1)$  ta có  $v(x,y) = -\log w + \alpha \eta_1(x,y)$  (4.26)

Ngoài các biểu diễn trên, trong các hệ thống ảnh kết cổ còn xuất hiện một loại nhiễu khác gọi là nhiễu đốm (speckle noise). Với các đối tượng có độ phân giải thấp nó tăng lên gấp bội và xảy ra nếu bề mặt đối tượng có độ lỗi lõm bậc bước sóng:

$$v(x,y) = w(x,y)s(x,y) + \eta(x,y) \quad (4.27)$$

trong đó:  $s(x,y)$  là cường độ nhiễu đốm - nó là trường nhiễu trắng ngẫu nhiên có mật độ hàm mũ:

$$s(x,y) = \begin{cases} \frac{1}{\sigma^2} \exp\left(-\frac{\xi}{\sigma^2}\right) & \text{với } \xi \geq 0 \\ p_1(\xi) = 0 & \text{khi }\xi < 0 \end{cases} \quad (4.28)$$

Trong một số trường hợp mẫu hoà đều, mô hình cho bài 4-16 đến 4-18 có thể thành một xấp xỉ rời rạc:

$$v(x,y) = g|w(x,y) + \eta(x,y) \quad (4.29)$$

$$w(x,y) = \sum_{(m,n) \in W} h(m,n;k,l) u(k,l) \quad (4.30)$$

$$\eta(x,y) = f(g(w(x,y))) | \eta_1(x,y) + \eta_2(x,y) \quad (4.31)$$

Sau khi đã nghiên cứu các mô hình thu nhận ảnh để xác định các biến dạng, tiếp theo ta dùng các bộ lọc ngược để khôi phục ảnh gốc. Có hai kỹ thuật chính là lọc tuyến tính và lọc phi tuyến. Các kỹ thuật lọc này đã trình bày kỹ trong chương 3. Sau đây ta chỉ xem xét riêng một số kỹ thuật lọc dùng trong khôi phục ảnh.

#### 4.2.2. Kỹ thuật lọc tuyến tính

Kỹ thuật lọc tuyến tính gồm nhiều loại: lọc ngược, lọc giả ngược, lọc Wiener, làm tròn bằng kỹ thuật Spline, lọc sai số bình phương nhỏ nhất có điều kiện, v.v. Các kỹ thuật này sẽ được mô tả dưới đây.

##### 4.2.2.1. Kỹ thuật lọc ngược (Inverse filter)

Lọc ngược là kỹ thuật lọc khôi phục đầu vào của một hệ thống khi biết đầu ra (ảnh thu được hay ảnh quan sát). Để đơn giản, ta giả thiết rằng hệ thống không có nhiễu và việc khôi phục  $u(x,y)$  được dựa vào  $v(x,y)$ . Quá trình đó được mô hình hóa như sau:



Hình 4.22. Mô hình lọc ngược.

Với một hệ thống như thế ta có:

$$g^T(x) = g^{-1}(x); \quad g^{-1}[g(x)] = x \quad (4.32)$$

$$\text{và} \quad h^T(x,y;k,l) = h^{-1}(x,y;k,l) \quad (4.33)$$

nghĩa là:

$$\sum_{k,l=-\infty}^{\infty} h^T(x,y;k',l') h(k',l';k,l) = \delta(x-k, y-l) \quad (4.34)$$

Lọc ngược rất có ích cho quá trình tiền hiệu chỉnh tín hiệu vào trước những biến dạng gây nên bởi hệ thống. Việc thiết kế một bộ lọc ngược là rất khó khăn vì nó không ổn định, do vậy ta có thể dùng biến đổi Fourier 2 vế của 4.34:

$$H^T(w_1, w_2) H(w_1, w_2) = 1 \quad (4.35)$$

$$\text{do đó } H^T(w_1, w_2) = 1 / H(w_1, w_2)$$

Chú ý rằng  $H^T(w_1, w_2)$  không phải luôn luôn tồn tại vì  $H(w_1, w_2)$  có thể nhận giá trị 0. Đây chính là nhược điểm lớn của kỹ thuật lọc ngược.

#### 4.2.2.2. Lọc giả ngược (Pseudoinverse Filter)

Do nhược điểm của lọc ngược là không ổn định (vì  $H^T$  có thể không tồn tại), người ta nghĩ đến cách cải tiến nó. Điều đơn giản là làm sao cho  $H^T$  luôn tồn tại. Bộ lọc giả ngược được định nghĩa:

$$H^T(w_1, w_2) = \begin{cases} 1 / H(w_1, w_2) & \text{nếu } H \neq 0 \\ 0 & H=0 \end{cases} \quad (4.36)$$

Trong thực tế, người ta coi  $H^T$  là 0 khi  $|H|$  nhỏ hơn một lượng  $\epsilon$  cho trước ( $\epsilon > 0$ ).

#### 4.2.2.3. Lọc Wiener

Lọc ngược và giả ngược có một yếu điểm là nhạy cảm với nhiễu. Vì thế khi áp dụng kiểu lọc này ta giả định là hệ thống lý tưởng không có nhiễu. Song trên thực tế điều này là không có. Do vậy, người ta nghĩ đến dùng kỹ thuật khác dùng cho các hệ thống có nhiễu gọi là lọc Wiener.

Gọi  $u(m,n)$  và  $v(m,n)$  là các chuỗi ngẫu nhiên bất kỳ, có trung bình 0. Người ta muốn tìm một xấp xỉ  $\hat{u}(m,n)$  của  $u(m,n)$  sao cho sai số trung bình bình phương là cực tiểu.

$$\text{Gọi } \sigma_e^2 = E\{|u(m,n) - \hat{u}(m,n)|^2\} \quad (4.37)$$

là sai số trung bình bình phương khi xấp xỉ  $u(m,n)$  bởi  $\hat{u}(m,n)$ . Giá trị tốt nhất của xấp xỉ  $\hat{u}(m,n)$  được biết khi trung bình có điều kiện của  $u(m,n)$  cho bởi  $v(m,n)$  với mỗi cặp  $(m,n)$ , có nghĩa là:

$$\hat{u}(m,n) = E\{|u(m,n) / v(k,l)| \forall k,l\} \quad (4.38)$$



Hình 4.23. Lọc ngược và giả ngẫu nhiên.

Nhìn chung 4-37 là rất khó giải vì không tuyến tính. Người ta nghĩ đến sử dụng một dạng tuyến tính khác của xấp xỉ 1:

$$\hat{u}(m,n) = \sum_{k,l=-\infty}^{\infty} g(m,n;k,l) v(k,l) \quad (4.39)$$

với  $g$  là đáp ứng xung được xác định sao cho sai số trung bình bình phương của 4-37 là cực tiểu.

Nếu giả thiết thêm rằng  $u, v$  là các chuỗi Gauss cùng nhau, thì lời giải của 4-37 là tuyến tính. Việc cực tiểu hóa 4-37 yêu cầu điều kiện:

$$\forall (m,n), (m',n') \quad E\{|u(m,n) \cdot \hat{u}(m,n)|^2\} = v(m',n') \quad (4.40)$$

Sử dụng định nghĩa của hiệp biến chéo (cross-covariance):

$$r_{u,v}(m,n;k,l) = E[u(m,n)b(k,l)] \quad (4.41)$$

cho 2 chuỗi ngẫu nhiên bất kỳ và cho 4-39, ròng buộc 4-40 trở thành:

$$\sum_{k,l=-\infty}^{\infty} g(m,n;k,l) r_{u,v}(k,l; m',n') = r_{u,v}(m,n; m',n') \quad (4.42)$$

Phương trình 4-37 và 4-42 là phương trình của bộ lọc Wiener.

$$\text{Nếu } u \text{ và } v \text{ là đồng cùng nhau thì: } r_{uv}(m,n;m',n') = r_{uu}(m-m';n-n') \quad (4.43)$$

Điều này cho phép đơn giản hóa g thành bộ lọc bắt biến không gian và nếu ký hiệu bởi  $g(m-k,n-l)$  thì 4-42 trở thành:

$$\sum_{k,l=-\infty}^{\infty} g(m-k,n-l)r_{uu}(k,l) = r_{uu}(m,n) \quad (4.44)$$

Biến đổi Fourier cho 2 vế của 4-44 ta có:

$$G(w_1, w_2) = S_{uu}(w_1, w_2)S_{uu}^{-1}(w_1, w_2) \quad (4.45)$$

với G là biến đổi Fourier của g,  $S_{uu}$  là biến đổi của  $r_{uu}$ , và  $S_{uu}^{-1}$  là biến đổi của  $r_{uu}$ . Phương trình trên gọi là đáp ứng tần số của bộ lọc Wiener và phương trình lọc trở thành:

$$\hat{u}(m,n) = \sum_{k,l=-\infty}^{\infty} g(m,n;k,l)v(k,l) \quad (4.46)$$

$$\hat{u}(m,n) = G(w_1, w_2)V(w_1, w_2) \quad (4.47)$$

$$\text{và} \quad v(m,n) = \sum_{k,l=-\infty}^{\infty} h(m-k,n-l)u(k,l) + \eta(x,y) \quad (4.48)$$

#### 4.2.2.4. Lọc Wiener với đáp ứng xung hữu hạn FIR(Finite Impulse Response)

Về lý thuyết, bộ lọc Wiener có đáp ứng xung vô hạn và do đó đòi hỏi DFT có kích thước lớn. Tuy nhiên, đáp ứng xung có hiệu quả chỉ là một phần nhỏ của kích thước đối tượng.

Nói chung việc thiết kế một FIR tối ưu là khá phức tạp. Người ta cài đặt một FIR như là tích chập của một bộ lọc có trọng số g, làm cực tiểu sai số trung bình bình phương E với  $v(m,n)$ :

$$\hat{u}(m,n) = \sum_{k,l \in W} g(k,l)v(m-k,n-l) \quad (4.49)$$

W là cửa sổ từ  $-M$  đến  $M$ :  $(k,l) \in W$  có nghĩa là  $-M \leq (k,l) \leq M$

Ràng buộc trực giao của 4-48 định nghĩa bởi:

$$\forall (k,l) \in W: E[u(m,n) - \hat{u}(m,n)]v(m-k,n-l) = 0 \quad (4.50)$$

sẽ làm giảm số phương trình xuống còn  $(2M+1)^2$

$$\forall (k,l) \in W: r_{\text{ss}}(m,n) - \sum_{i,j \in W} g(k,i)r_{\text{ss}}(m-k,n-l) = 0 \quad (4.51)$$

(kết quả này suy ra từ 4-40, 4-41 và 4-42).

Áp dụng 4-47 và giả thiết thêm rằng  $\eta(m,n)$  là nhiễu trắng có trung bình 0 và độ lệch  $\sigma^2$ , ta có:

$$r_{\text{ss}}(k,l) = r_{\text{ss}}(k,l) \otimes u(k,l) + \sigma^2 \alpha(k,l) \quad (4.52)$$

$$\alpha(k,l) = h(k,l) * h(k,l) = \sum_{i,j=-\infty}^{\infty} h(i,j)h(i+k,j+l) \quad (4.53)$$

$$r_{\text{ss}}(k,l) = h(k,l) * r_{\text{ss}}(k,l) = \sum_{i,j=-\infty}^{\infty} h(i,j)r_{\text{ss}}(i+k,j+l) \quad (4.54)$$

Định nghĩa hàm tương quan:

$$r_d(k,l) = \frac{r_{\text{ss}}(k,l)}{r_{\text{ss}}(0,0)} = \frac{r_{\text{ss}}(k,l)}{\sigma^2} \quad (4.55)$$

Tùy theo các hệ thống và nhu cầu khôi phục ảnh, người ta còn sử dụng nhiều biến đổi khác cho bộ lọc Wiener [Anil.K.Jain trang 236-294].

#### 4.2.2.5. Kỹ thuật làm tròn spline và nội suy

Kỹ thuật spline là dùng một đường cong để xấp xỉ một hàm liên tục từ các giá trị mẫu (giá trị quan sát được) trên một lưới. Trong các quá trình xử lý ảnh, hàm spline được dùng để khuếch đại ảnh, làm tròn nhiễu. Trước tiên, ta xử lý mỗi điểm ảnh theo hàng ngang và khuếch đại ngang cho hàng, tiếp theo áp dụng chính thủ tục này cho cột. Như vậy, ảnh sẽ làm tròn và nội suy bởi một hàm tách được (theo nghĩa thực hiện riêng cho từng chiều).

Gọi  $y_i$  là một chuỗi các giá trị quan sát được của một hàm liên tục mẫu đều trên khoảng  $[0,N]$ , với  $i = 0, 1, 2, \dots, N$ :

$$x_i = x_0 + ih \quad i > 0$$

$$\text{và} \quad y_i = f(x_i) + \eta(x_i) \quad (4.56)$$

với  $\eta(x_i)$  biểu diễn sai số của quá trình quan sát.

Đường spline điều chỉnh một hàm tròn  $g(x)$  trên toàn bộ tập giá trị quan sát được mà độ gồ ghề của nó do bởi phổ năng lượng trên khoảng  $[0,N]$  sao cho sai số là cực tiểu. Hơn nữa, nếu sai số bình phương nhỏ nhất tại các điểm quan sát là bị chặn, có nghĩa là:

$$g_i = g(x_i) \text{ và } F = \sum_{i=0}^N \frac{|g_i - y_i|}{\sigma^2} \leq S \quad (4.57)$$

Nếu  $S=0$ , có nghĩa là đường spline trùng hoàn toàn với đường cong biểu diễn các điểm quan sát được. Đặc biệt, nếu  $\sigma^2$  là giá trị trung bình bình phương của nhiễu,  $S$  được chọn nằm trong khoảng  $(N-1) = \sqrt{2}(N+1)$ . Khoảng này gọi là *khoảng tin cậy* của  $S$ . Phụ thuộc vào giá trị của  $S$ , ta có 2 lời giải:

- Nếu  $S$  là khá lớn và 4.57 thoả (xấp xỉ) bởi 1 đường thẳng:

$$\begin{aligned} g(x) &= a + bx \quad x_0 \leq x \leq x_N \\ b &= \frac{\mu_{xy} - \mu_x \mu_y}{\mu_{xx} - \mu_x^2}, \quad a = \mu_x - b\mu_y \end{aligned} \quad (4.58)$$

$\mu$  biểu diễn giá trị trung bình của mẫu, thí dụ  $\mu_x = \frac{\sum_{i=0}^N x_i}{(N+1)}$

- Ràng buộc 4.57 là chặt chẽ và chỉ có một ràng buộc đẳng thức có thể thoả mãn. Lời giải sẽ là một đường spline bậc 3 định nghĩa bởi:

$$g(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3 \quad x_i \leq x \leq x_{i+1} \quad (4.59)$$

Các hệ số  $a_i, b_i, c_i$  và  $d_i$  là nghiệm của phương trình:

$$\begin{cases} (P + \lambda Q)c = \lambda v \\ v = L^T y \quad c_0 = c_N = 0 \\ d_i = (c_{i+1} - c_i)/3h \quad 0 \leq i \leq N-1, d_0 = d_N = 0 \\ b_i = \frac{a_{i+1} - a_i}{h} - hc_i - h^2d_i, \quad b_N = 0, 0 \leq i \leq N-1 \end{cases} \quad (4.60)$$

với:

- $a, y$  và  $d$  là các vectơ có  $N+1$  phần tử;
- $c$  là vectơ có  $N-1$  phần tử ( $i=1, 2, \dots, N-1$ );

$$Q = h/3 \begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix}$$

$$L = 1/h \begin{bmatrix} 1 & 0 & & \\ -2 & 1 & & \\ 1 & -2 & 1 & \\ 0 & 1 & -1 & \end{bmatrix}$$

$$P = \delta^2 L^T L$$

-  $Q$  là ma trận 3 đường chéo Toeplitz:  $(N-1) \times (N-1)$ ;

-  $L$  là ma trận tam giác dưới  $(N+1) \times (N-1)$ .

Thí dụ: Cho dãy giá trị quan sát  $y = \{1, 3, 4, 2, 1\}$

-  $h=1$  và  $\delta=1$

-  $N=4$  và  $x_0=0$ ,  $x_4=4$  (gồm 5 điểm)

- nội suy tuyến tính là đường thẳng

-  $\mu_x = 2$ ,  $\mu_y = 2.2$ ,  $\mu_{xy} = 4.2$  và  $\mu_{xx} = 6$

Với các giá trị trên, ta tính được  $b = -1$ ,  $a = 2.4$  và  $g = 2.4 - 0.1x$ .

Sai số bình phương nhỏ nhất  $\sum_{i=0}^N (y_i - g)^2 = 6.7$

Khoảng tin cậy cho  $S$  là  $[1.8; 8.16]$ . Tuy nhiên nếu chọn  $S = 5$ , chúng ta sẽ có đường xấp xỉ là đường spline bậc 3. Với số liệu trên, ta dễ dàng tính được:

$$P = L^T L = \begin{pmatrix} 6 & -4 & 1 \\ -4 & 6 & -4 \\ 1 & -4 & 6 \end{pmatrix} \quad V = L^T y = \begin{pmatrix} -1 \\ -3 \\ 1 \end{pmatrix}$$

Giải phương trình  $P \cdot S = 0$  với  $S = 5$  ta thu được  $x = 0,0274$ . Giải phương trình 4.60 ta thu được các vectơ  $a$ ,  $b$ ,  $c$  và  $d$ . Giá trị sai số nhỏ nhất kiểm nghiệm lại là  $4.998 \approx 5$ . Như vậy chấp nhận được.

$$a = \begin{pmatrix} 2,199 \\ 2,397 \\ 2,415 \\ 2,180 \\ 1,808 \end{pmatrix} \quad b = \begin{pmatrix} 0,198 \\ 0,157 \\ 0,194 \\ -0,357 \\ 0,000 \end{pmatrix} \quad c = \begin{pmatrix} 0,000 \\ -0,033 \\ -0,040 \\ -0,022 \\ 0,000 \end{pmatrix} \quad d = \begin{pmatrix} 0,000 \\ -0,005 \\ 0,009 \\ 0,007 \\ 0,000 \end{pmatrix}$$

#### 4.2.3. Kỹ thuật lọc phi tuyến trong khôi phục ảnh

Một số kỹ thuật lọc phi tuyến đã được mô tả trong chương 3, nhưng ở đây sẽ đưa thêm kỹ thuật lọc đóng hình để khử nhiễu dốm, kỹ thuật Entropy cực đại, giảichap mù, mô hình Bayesian, v.v.

#### 4.2.3.1. Lọc nhiễu đóm

Như đã nói ở trên, nhiễu đóm này sinh khi các tia đơn sắc được tán xạ từ bề mặt nhám mà độ gó ghê bằng bước sóng. Trong không gian tự do, nhiễu đóm có thể coi như tổng vô hạn các pha đồng nhất, độc lập mà pha và biên độ là ngẫu nhiên. Như vậy ta có thể biểu diễn:

$$u(x,y) = u_0(x,y) + j s(x,y) \quad (4.61)$$

với  $u_0$ ,  $s$  là các biến ngẫu nhiên độc lập theo phân bố Gauss, trung bình 0 cho mỗi cặp  $(x,y)$  với độ sai lệch  $\sigma^2$ . Trường cường độ  $S$ :

$$S(x,y) = |u(x,y)| = \sqrt{u_0^2 + s^2} \quad (4.62)$$

có phân bố mũ với lượng sai lệch  $\sigma^2 = 2\sigma^2$  và trung bình  $\mu_s = E[s] = \sigma^2$ . Với một loại nhiễu đóm, tỉ lệ tương phản được định nghĩa:

$$\gamma = \text{phương sai } s/\text{trung bình } s \quad (4.63)$$

Một phương pháp đơn giản để giảm nhiễu trắng là N-Look, có nghĩa là lấy ảnh ở N thời điểm khác nhau rồi tính trung bình các lần quan sát. Giả sử sự phát hiện nhiễu là thấp và ảnh lần thứ nhất được viết:

$$v_i(x,y) = u(x,y)s_i(x,y) \quad i = 1, 2, \dots, N \quad (4.64)$$

Như vậy trung bình tức thời của N quan sát là:

$$\bar{u}_N(x,y) = \frac{1}{N} \sum_{i=1}^N v_i(x,y) = u(x,y) \bar{s}_N(x,y) \quad (4.65)$$

trong đó:  $\bar{s}_N(x,y)$  là trung bình N quan sát của trường nhiễu đóm. Đó cũng là ước lượng gần giống nhất của  $s_i(x,y)$ ,  $i=1,\dots,N$  và  $E[\bar{u}_N] = \mu_s/N$ ; sai lệch  $= u^2 \mu_s^2 / N$ . Như vậy  $\gamma = 1/N$  cho  $\bar{u}_N$ .

Nếu số lượng nhìn N là nhỏ, để cho hiệu quả người ta thực hiện một số kiểu lọc không gian để giảm nhiễu đóm. Một kỹ thuật để lọc nhiễu đóm trong ra đa là tính trung bình giá trị cường độ của các điểm lân cận. Độ tương phản được cải thiện so với phương pháp N-quan sát. Do bản chất của nhiễu đóm, người ta dùng biến đổi logarit của 4.62 và thu được:

$$\log \bar{u}_N(x,y) = \log u(x,y) + \log \bar{s}_N(x,y) \quad (4.66)$$

Nếu kỹ hiệu  $W_N = \log \hat{U}_N(x,y)$ ,  $Z = \log u(x,y)$  và  $\eta_N = \log \epsilon_N(x,y)$ , ta có mô hình quan sát nhiễu phụ  $W_N(x,y) = Z(x,y) + \eta_N(x,y)$  (4.67)

với  $\eta_N(x,y)$  là nhiễu trắng dừng.

Với  $N \geq 2\eta_N$ , có thể mô hình hóa bởi trường ngẫu nhiên Gauss mà hàm mật độ phô được định nghĩa bởi:

$$S_1(\xi_1, \xi_2) = \sigma^2 = \begin{cases} \pi^2/6 & \text{nếu } N = 1 \\ 1/N & \text{nếu } N > 1 \end{cases}$$

Bây giờ  $Z(x,y)$  có thể ước lượng dễ dàng từ  $W(x,y)$  bởi bộ lọc Wiener.

#### 4.2.3.2. Kỹ thuật entropy cực đại

Người ta nhận thấy rằng: đầu vào, đầu ra và PSF của các hệ thống ảnh không kết cù thường là không âm. Các thuật toán khôi phục ảnh dựa vào trung bình bình phương hay bình phương cực tiểu không có lợi cho ảnh với các giá trị không âm. Phương pháp dựa vào entropy cực đại cho lời giải không âm. Cơ sở lập luận của phương pháp này là ở chỗ: entropy là số đo của một đại lượng không chắc chắn, do vậy nó đặt rất ít giả thiết về lời giải và tạo nên một sự tự do cực đại về các ràng buộc.

Với một ảnh quan sát được  $v = \pi u$ , với  $\pi$  là ma trận PSF;  $u, v$  là các ma trận biểu diễn đối tượng và quan sát. Kỹ thuật entropy cực đại nhằm mục đích:

$$g(u) = - \sum_n u(n) \log u(n) \quad (4.68)$$

với ràng buộc  $\frac{1}{2} \|v - \pi u\|^2 = \sigma_v^2 > 0$  (4.69)

Vì  $u(n)$  không âm, do vậy ta có thể chuẩn hoá cho  $\sum u(n) = 1$ . Như vậy ta có thể xử lý như phân bố xác suất mà Entropy là  $S(u)$ . Dùng phương pháp tối ưu của Lagrange, lời giải của phương trình trên cho bởi:

$$\hat{u} = \exp\{-l \cdot \lambda \pi^T(v - \pi \hat{u})\} \quad (4.70)$$

với  $\lambda$  là hằng số Lagrange,  $l$  là vectơ.

Một lời giải khác tốt hơn nếu ta định nghĩa các ràng buộc:

$$\begin{cases} u(n) \geq 0 \\ \sum_{j=0}^{N-1} h(m, j)u(j) = v(m), \quad m = 0, \dots, M-1 \end{cases} \quad (4.71)$$

và lời giải cho bài:

$$\hat{u}(n) = \frac{1}{c} \exp \left[ -\sum_{l=0}^{M-1} h(l, n)\lambda(l) \right] \quad n = 0, 1, \dots, N-1 \quad (4.72)$$

với  $\lambda(l)$  là hằng số Lagrange.

#### 4.2.3.3. Phương pháp Bayesian

Trong nhiều tình huống ảnh, thí dụ như hệ thống ghi phim, mô hình quan sát là không tuyến tính và có dạng:

$$v = f(\pi u + \eta) \quad (4.73)$$

với  $f(x)$  là một hàm không tuyến tính của  $x$  và  $\eta$  biểu diễn nhiễu.

Công thức nổi tiếng của Bayes về xác suất có điều kiện cho bài:

$$p(u | v) = p(v | u) p(u) / p(v) \quad (4.74)$$

Nó rất có ích để xác định nhiều kiểu ước lượng khác nhau cho một vectơ ngẫu nhiên  $u$  từ một vectơ quan sát  $v$ . Có một số kiểu ước lượng chính sau:

- MMSE: ước lượng trung bình bình phương cực tiểu của  $u$ .
- MAP: ước lượng xác suất có điều kiện cực đại  $p(u | v)$ .
- ML: ước lượng gần đúng nhất  $p(v | u)$ .

mà các đối tượng sử dụng là xác suất có điều kiện  $p(v | u)$  hay  $p(u | v)$ .

Vì rất khó xác định  $p(v)$  ngay cả khi  $u$  và  $\eta$  là phân bố Gauss, nên người ta hay sử dụng MAP và ML vì nó không đòi hỏi  $p(v)$ . Nếu giả thiết  $u$  và  $\eta$  là phân bố Gauss với hiệp biến  $R_u$  và  $R_\eta$ , các ước lượng ML, MAP có thể tính được khi giải các phương trình sau:

$$\hat{u}_{ML}: \pi \mathcal{D} R^{-1} [v - f(\pi \hat{u}_{ML})] = 0 \quad (4.75)$$

với  $\mathcal{D}$  là ma trận đường chéo = Diag  $\{df(x)/dx \text{ với } x = wi\}$  (4.76)

$w_i$  là các phần tử của  $W = \bar{\pi} \hat{u}_{ML}$ .

và  $\hat{u}_{MAP} = \mu_v + R_v \pi^T \mathcal{D} R_v^{-1} [v - f(\pi \hat{u}_{MAP})]$  (4.77)

Nếu  $f(x)$  là tuyến tính, thí dụ  $f(x)=x$ ,  $R_v = \sigma_v^2$ , thì  $\hat{u}_{ML}$  là lời giải của phương trình:

$$\pi^T \pi \hat{u}_{ML} = \pi^T v \quad (4.78)$$

và  $\hat{u}_{MAP} = \mu_v + G(v - \mu_v) \quad (4.79)$

với  $G = (R_v^{-1} + \pi^T R_v^{-1} \pi)^{-1} \pi^T R_v^{-1} \quad (4.80)$

Trong thực tế,  $\mu$  có thể lấy giá trị là trung bình cục bộ của  $v$  và  $\mu_v = \pi^T f^{-1}(\mu_v)$ .  $\pi^*$  là biến đổi ngược của  $\pi$ .

#### 4.2.3.4. Giải chập mù (Blind deconvolution)

Việc khôi phục ảnh khi PSF không biết là một vấn đề khôi phục phi tuyến khó khăn. Với một hệ ảnh bất biến không gian, mật độ phổ năng lượng của ảnh quan sát tuân theo:

$$\begin{aligned} S_{vv}(w1, w2) &= \|H(w1, w2)\|^2 S_{vv}(w1, w2) + S_{vn}(w1, w2) \\ S_{vn}(w1, w2) &= H^*(w1, w2) S_{vv}(w1, w2) \\ \text{và } \log \|H\|^2 &= \log(S_{vv} - S_{vn}) - \log S_{vv} \end{aligned} \quad (4.81)$$

Nếu nhiễu cộng là nhỏ, chúng ta có thể dùng ước lượng:

$$\log \|H\| = \frac{1}{2M} \sum_{k=1}^M \log |vk|^2 - \log |uk|^2 = \frac{1}{M} \sum_{k=1}^M \log |vk| - \log |uk| \quad (4.82)$$

với  $v_k$  và  $u_k$  là các khối của ma trận biểu diễn  $v(m,n)$  và  $u(m,n)$  trong biến đổi Fourier ( $k=1, 2, \dots, M$ ).

Phương pháp này dựa vào ước lượng năng lượng phổ chưa biết nên có tên gọi là "giải chập mù". Trong một số tình huống, pha của  $H$  là 0 hay là 1 không quan trọng, lúc đó  $H$  biểu diễn trung bình nhiễu loạn môi trường, camera không hội tụ, trễ pha, v.v.

Trong khôi phục ảnh, các mô hình toán là rất nặng nề và phức tạp. Trên đây chỉ để cập một phần cơ sở lý thuyết và một số kỹ thuật lọc trong khôi phục ảnh. Bạn đọc quan tâm xin tham khảo tài liệu [1].



**Bài tập chương 4****Câu 1**

Cho ảnh số và các nhân chập sau:

$$I = \begin{pmatrix} 4 & 7 & 2 & 7 & 1 \\ 5 & 7 & 1 & 7 & 1 \\ 6 & 6 & 30 & 8 & 3 \\ 5 & 7 & 6 & 1 & 2 \\ 5 & 7 & 6 & 1 & 2 \end{pmatrix} \quad H_s = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad H_a = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

1. Ảnh trên có nhiễu không? Đó là loại nhiễu gì?
2. Minh họa khử nhiễu trên bằng bộ lọc thông thấp  $H_s$ .
3. Hãy tính kết quả của nhân chập ảnh với nhân chập  $H_a$ .
4. Hãy biến đổi ảnh sau khi khử nhiễu về ảnh nhị phân (dùng kỹ thuật phân nguồng hay dựa vào lược đồ xám).

**Câu 2**

1. Viết thủ tục dùng kỹ thuật lọc trung vị sử dụng bộ lọc chữ thập kích thước  $3 \times 3$  và  $5 \times 5$ . Việc sắp xếp các điểm theo thuật toán tùy chọn (chọn đơn giản, chèn tuyến tính hay đổi chỗ).
2. Viết thủ tục cải thiện ảnh dùng kỹ thuật lọc theo mô hình Gauss.
3. Viết thủ tục thực hiện việc dán ảnh bằng kỹ thuật Dialstation.
4. Viết thủ tục thực hiện việc ăn mòn ảnh bằng kỹ thuật Erosion.

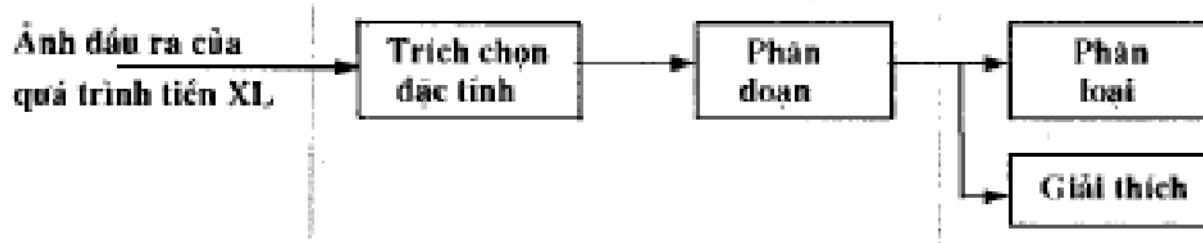
**5**

# CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN

EDGE DETECTION

## 5.1. TỔNG QUAN VỀ PHÂN TÍCH ẢNH (IMAGE ANALYSIS)

Các chương trước đã trình bày về các vấn đề cơ bản của giai đoạn tiền xử lý ảnh. Sau giai đoạn này, ảnh đã được tăng cường hay được khôi phục để làm nổi các đặc trưng chủ yếu. Giai đoạn phân tích ảnh bao gồm: trích chọn các đặc tính (feature extraction), tiếp theo là phân đoạn ảnh (segmentation) thành các phần tử. Thí dụ, như phân đoạn dựa theo biên, dựa theo vùng, ... Và tùy theo các ứng dụng, giai đoạn tiếp theo có thể là nhận dạng ảnh (phân thành các lớp có miêu tả) hay là giải thích và miêu tả ảnh. Hình 5.1 mô tả tóm lược các bước của quá trình phân tích ảnh:



Hình 5.1. Các bước trong phân tích ảnh.

Các đặc trưng của ảnh thường gồm: mật độ xám, phân bố xác suất, phân bố không gian, biên ảnh. Vì biên có tầm quan trọng đặc biệt trong phân tích ảnh nên chương 5 được dành riêng để mô tả về nó cũng như các phương pháp dò biên. Các bước khác sẽ được trình bày trong các chương tiếp theo: chương 6 và chương 7.

## 5.2. KHAI QUÁT VỀ BIÊN VÀ PHÂN LOẠI CÁC KỸ THUẬT DÒ BIÊN

Trong phần này chúng ta sẽ đề cập đến một số nội dung: khái niệm về biên, phân loại các phương pháp phát hiện biên và qui trình phát hiện biên.

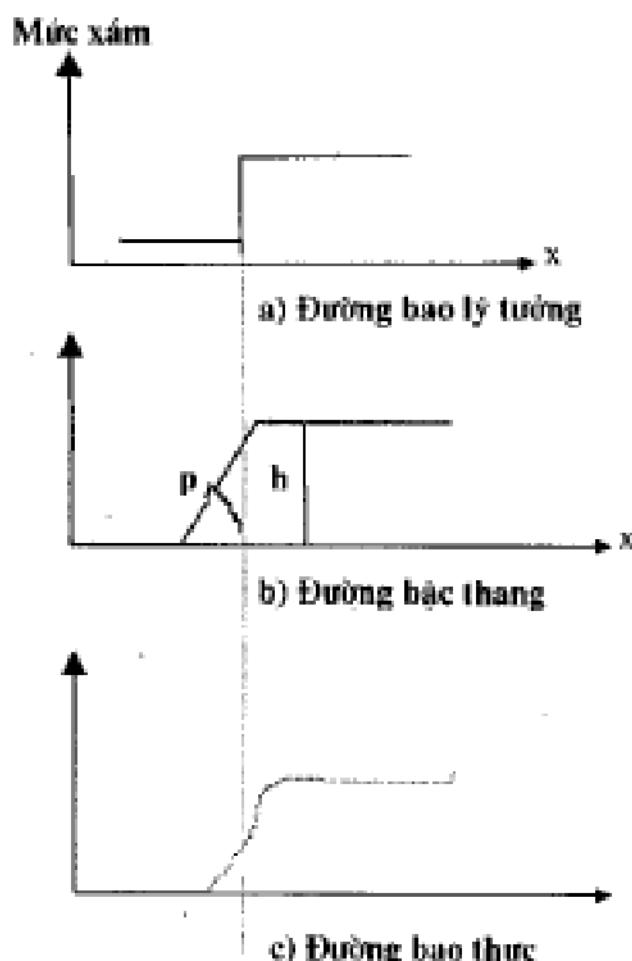
### 5.2.1. Khái niệm về biên

Biên là một vấn đề chủ yếu trong phân tích ảnh vì các kỹ thuật phân đoạn ảnh chủ

yếu đưa vào biên. Một điểm ảnh có thể coi là điểm biên nếu ở đó có sự thay đổi đột ngột về mức xám. Tập hợp các điểm biên tạo thành biên hay đường bao của ảnh (boundary). Thí dụ, trong một ảnh nhị phân, một điểm có thể gọi là biên nếu đó là điểm đen và có ít nhất một điểm trắng lân cận.

Để hình dung tầm quan trọng của biên ta xét thí dụ sau: Khi người họa sĩ vẽ một cái bàn gỗ, chỉ cần vài nét phác thảo về hình dáng như mặt bàn, chân bàn mà không cần thêm các chi tiết khác, người xem đã có thể nhận ra đó là cái bàn. Nếu ứng dụng của ta là phân lớp nhận diện đối tượng, thì coi như nhiệm vụ đã hoàn thành. Tuy nhiên nếu đòi hỏi thêm về các chi tiết khác như vân gỗ hay màu sắc,... thì với chừng ấy thông tin là chưa đủ.

Nhìn chung về mặt toán học, người ta coi điểm biên của ảnh là: điểm có sự biến đổi đột ngột về độ xám như chỉ ra trong hình 5.2 dưới đây:



Như vậy, phát hiện biên một cách lý tưởng là xác định được tất cả các đường bao trong các đối tượng. Định nghĩa toán học của biên ở trên là cơ sở cho các kỹ thuật phát hiện biên. Điều quan trọng là sự biến thiên mức xám giữa các điểm ảnh trong một vùng thường

là nhỏ, trong khi đó biến thiên mức xám của điểm ở vùng giáp ranh (khi qua biên) lại khá lớn.

### 5.2.2. Phân loại các kỹ thuật phát hiện biên

Xuất phát từ định nghĩa toán học của biên người ta thường sử dụng 2 phương pháp phát hiện biên sau:

- **Phương pháp phát hiện biên trực tiếp:** phương pháp này nhằm làm nổi biên dựa vào sự biến thiên về giá trị độ sáng của điểm ảnh. Kỹ thuật chủ yếu dùng phát hiện biên ở đây là kỹ thuật đạo hàm. Nếu lấy đạo hàm bậc nhất của ảnh ta có phương pháp gradient; nếu lấy đạo hàm bậc hai ta có kỹ thuật Laplace. Hai phương pháp trên được gọi là phương pháp dò biên cục bộ. Ngoài ra, người ta cũng sử dụng phương pháp "*đi theo đường bao*": dựa vào nguyên lý qui hoạch động và được gọi là phương pháp dò biên tổng thể. Các kỹ thuật này sẽ được mô tả chi tiết dưới đây.
- **Phương pháp gián tiếp:** nếu bằng cách nào đấy, ta phân được ảnh thành các vùng thì đường phân ranh giữa các vùng đó chính là biên. Việc phân vùng ảnh thường dựa vào kết cấu (texture) bề mặt của ảnh. Đó là chủ đề của chương 6.

Cũng cần lưu ý rằng, kỹ thuật dò biên và phân vùng ảnh là hai bài toán đối ngẫu của nhau. Thực vậy, dò biên để thực hiện phân lớp đối tượng và một khi đã phân lớp xong có nghĩa là đã phân vùng được ảnh. Và ngược lại, khi đã phân vùng, ảnh đã phân lập được thành các đối tượng, ta có thể phát hiện được biên. Phương pháp dò biên trực tiếp tỏ ra khá hiệu quả vì ít chịu ảnh hưởng của nhiễu. Song nếu sự biến thiên độ sáng không dột ngọt, phương pháp này lại tỏ ra kém hiệu quả. Phương pháp dò biên gián tiếp tuy khó cài đặt, song lại áp dụng khá tốt khi sự biến thiên độ sáng nhỏ.

### 5.2.3. Quá trình phát hiện biên

b1) Vì ảnh thu nhận thường có nhiễu, nên bước đầu tiên là phải khử nhiễu. Việc khử nhiễu được thực hiện bằng các kỹ thuật chỉ ra trong chương 4.

b2) Tiếp theo là tiến hành làm nổi biên bởi các toán tử đạo hàm.

b3) Định vị điểm biên. Vì các kỹ thuật làm nổi biên có hiệu ứng phụ là tăng nhiễu, do vậy sẽ có một số điểm biên giả cần phải loại bỏ.

b4) Liên kết và trích chọn biên.

Như trên đã nói, phát hiện biên và phân vùng ảnh là một bài toán đối ngẫu. Vì thế

một số bước trong phát hiện biên sẽ được trình bày trong chương này, một số bước tiếp theo sẽ trình bày trong chương 6.

### 5.3. PHƯƠNG PHÁP PHÁT HIỆN BIÊN CỤC BỘ

#### 5.3.1. Phương pháp gradient

Phương pháp gradient là phương pháp đàm biến cục bộ dựa vào cục đại của đạo hàm. Theo định nghĩa, gradient là một vector có các thành phần biểu thị tốc độ thay đổi giá trị của điểm ảnh theo 2 hướng x và y. Các thành phần của gradient được tính bởi:

$$\frac{\partial f(x,y)}{\partial x} = f_x \approx \frac{f(x+dx, y) - f(x, y)}{dx}$$

$$\frac{\partial f(x,y)}{\partial y} = f_y \approx \frac{f(x, y+dy) - f(x, y)}{dy}$$

với  $dx$  là khoảng cách giữa các điểm theo hướng x (khoảng cách tính bằng số điểm) và tương tự với  $dy$ . Trên thực tế, người ta hay dùng với  $dx = dy = 1$ .

Với một ảnh liên tục  $f(x,y)$ , các đạo hàm riêng của nó cho phép xác định vị trí cục đại cục bộ theo hướng của biến. Thực vậy, gradient của một ảnh liên tục, được biểu diễn bởi một hàm  $f(x,y)$ , đọc theo  $r$  với góc  $\theta$ , được định nghĩa bởi:

$$\frac{df}{dr} = \frac{\partial f}{\partial x} \frac{dx}{dr} + \frac{\partial f}{\partial y} \frac{dy}{dr} = f_x \cos \theta + f_y \sin \theta \quad (5.1)$$

$\frac{df}{dr}$  đổi với  $\theta$  đạt cục đại khi  $(df/d\theta)(df/dr) = 0$

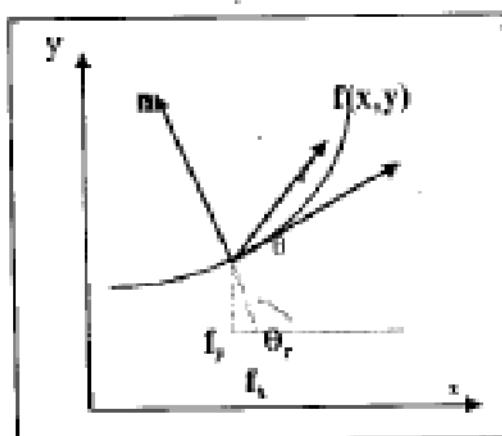
hay  $-f_y \sin \theta + f_x \cos \theta = 0$ . Do vậy ta có thể xác định hướng cục đại của nó:  $\theta_r = \tan^{-1}(f_y/f_x)$

$$\text{và } \frac{df}{dr} \text{ max} = \sqrt{f_x^2 + f_y^2}$$

Ta gọi  $\theta_r$  là hướng của biến.

Hình 5.3. gradient của ảnh theo hướng  $\theta$

Trong kỹ thuật gradient, người ta chia nhỏ thành 2 kỹ thuật (do dùng 2 toán tử khác nhau): kỹ thuật gradient và kỹ thuật la bàn. Kỹ thuật gradient dùng toán tử gradient lấy đạo hàm theo 2 hướng; còn kỹ thuật la bàn dùng toán tử la bàn lấy đạo hàm theo 8 hướng chính: Bắc, Nam, Đông, Tây và Đông Bắc, Tây Bắc, Đông Nam, Tây Nam.



Hình 5.3. Gradient của ảnh theo hướng  $\theta$ .

### 5.3.1.1. Kỹ thuật gradient

Kỹ thuật gradient sử dụng một cặp mặt nạ  $H_1$  và  $H_2$  trực giao (theo 2 hướng vuông góc). Nếu định nghĩa  $g_x, g_y$  là gradient tương ứng theo 2 hướng x và y, thì biên độ của gradient, ký hiệu là  $g$  tại điểm  $(m,n)$  được tính theo công thức:

$$A_g = g(m,n) = \sqrt{g_x^2(m,n) + g_y^2(m,n)} \quad (5.2)$$

$$\theta_g(m,n) = \tan^{-1} g_y(m,n)/g_x(m,n) \quad (5.3)$$

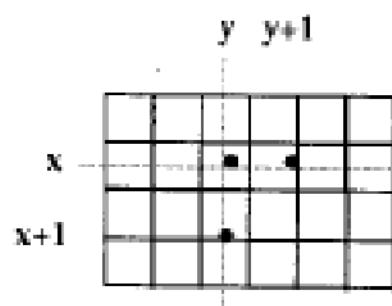
*chú ý: để giảm tính toán, công thức 5.2 được tính gần đúng bởi:*

$$A_g = |g_x(m,n)| + |g_y(m,n)|$$

Các toán tử đạo hàm được áp dụng là khá nhiều. ở đây, ta chỉ xét một số toán tử tiêu biểu: toán tử Robert, Sobel, Prewitt,...

Trước tiên, chúng ta xét toán tử Robert. Toán tử này do Robert đề xuất vào năm 1965. Nó là áp dụng trực tiếp của công thức đạo hàm tại điểm  $(x,y)$ . Với mỗi điểm ảnh  $I(x,y)$  của  $I$ , đạo hàm theo x, theo y được ký hiệu tương ứng bởi  $g_x, g_y$  được tính:

$$\begin{cases} g_x = I(x+1,y) - I(x,y) \\ g_y = I(x,y+1) - I(x,y) \end{cases} \quad (5.4)$$



Hình 5.4. Xấp xỉ  $g_x, g_y$  trong toán tử Robert.

Điều này tương đương với việc chụp ảnh với 2 mặt nạ  $H_1$  và  $H_2$ :

$$H_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} H_2 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Ta gọi  $H_1, H_2$  là mặt nạ Robert.

Trong trường hợp tổng quát, giá trị gradient biên độ  $g$  và gradient hướng  $\theta_g$  được tính bởi công thức 5.2 và 5.3. Thường để giảm thời gian tính toán, người ta còn tính gradient theo các chuẩn sau:

$$A_g = |g_x(m,n) + g_y(m,n)| \quad (5.5)$$

hoặc  $A_2 = \text{Max}(|g_x(x,y)|, |g_y(x,y)|)$  (5.6)

Cần lưu ý rằng, do lạm dụng về ngôn từ, tuy ta nói lấy đạo hàm của ảnh nhưng thực ra chỉ là mô phỏng và xấp xỉ đạo hàm bằng kỹ thuật nhân chập do ảnh số là tín hiệu rời rạc, do vậy đạo hàm không tồn tại.

Trong kỹ thuật Sobel và Prewitt người ta sử dụng 2 mặt nạ:

$$H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}; \quad H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Ngang (hướng x)      Dọc (hướng y)      Ngang (hướng x)      Dọc (hướng y)

a) Mật nạ Sobel

$$H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

b) Mật nạ Prewitt

Ngang(hướng x)      Dọc(hướng y)

c) Mật nạ đẳng hướng (Isometric)

Gradient được xấp xỉ bởi công thức:

$$G_x = H_x \otimes I \quad \text{và} \quad G_y = H_y \otimes I$$

Thực tế cho thấy rằng, các toán tử Sobel và Prewitt tốt hơn toán tử Robert bởi chúng ít nhạy cảm với nhiễu. Ta cũng thấy rằng việc lấy đạo hàm một tín hiệu có xu hướng làm tăng nhiễu trong tín hiệu đó. Độ nhạy cảm này có thể giảm xuống nhờ thao tác lấy trung bình cục bộ trong miền phủ bởi mật nạ. Lưu ý rằng, toán tử Sobel và Prewit dễ dàng chuyển đổi cho nhau bằng cách thay đổi hai hệ số.

Cần chú ý thêm là các chuẩn trong công thức 5.5 và 5.6 tạo nên sự vận xoắn trong tính toán biên độ. Thực vậy, nếu  $g_x$  hoặc  $g_y$  bằng 0, thì  $A_1 = A_2 = A_0$ ; còn nếu  $g_x = g_y$ , ta sẽ có  $A_1 = g_x$ ,  $A_2 = g_y$  và  $A_0 = \sqrt{2} g_x$ . Phương pháp Robert và Sobel dùng chuẩn  $A_1$ . Nguyên nhân của sự vận xoắn là khá đơn giản: các mật nạ của các toán tử tạo nên một xấp xỉ rời rạc của đạo hàm thực và các chuẩn áp dụng để tính biên độ lại phụ thuộc vào giá trị của các gradient thành phần.

Để giúp bạn đọc hiểu rõ cách xấp xỉ đạo hàm bậc nhất bằng kỹ thuật nhân chập hay ý nghĩa của các nhân chập nêu trên (thí dụ như nhân chập Sobel), ta tiến hành lấy đạo hàm của một hàm 2 biến liên tục  $f(x,y)$ . Về phương diện toán học, công thức tính gần đúng của đạo hàm như sau:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x} \approx \frac{f(x, y) - f(x - \Delta x, y)}{\Delta x}$$

Do vậy:  $\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x + \Delta x, y) - f(x - \Delta x, y)}{2\Delta x}$

Một cách tương tự, đạo hàm theo  $y$  được xấp xỉ bởi:

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x, y + \Delta y) - f(x, y - \Delta y)}{2\Delta y}$$

Trong không gian rời rạc, số giá  $\Delta x, \Delta y$  được thay bằng  $dx$  và  $dy$  tương ứng (khoảng cách giữa các điểm) và  $dx, dy$  thường chọn là 1. Với cách tính trên và áp dụng trong lân cận  $3 \times 3$ , đạo hàm theo  $x$ , theo  $y$  của 1 ảnh được tính:

$$f_x = G_x = H_x \otimes I \quad \text{và} \quad f_y = G_y = H_y \otimes I \quad (*)$$

với  $H_x, H_y$  là các nhân chập theo các xấp xỉ đạo hàm tương ứng. Bạn đọc tự kiểm tra lại công thức (\*) như một bài tập nhỏ và dễ thấy rằng có nhiều toán tử khác nhau trong phát hiện biên là do nhiều cách xấp xỉ đạo hàm khác nhau.

Ngoài các nhân chập  $3 \times 3$ , người ta cũng còn dùng một số nhân chập  $5 \times 5$  trong kỹ thuật phát hiện biên:

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

Với mục đích nghiên cứu các mặt nạ cho kết quả tốt hơn, người ta nghĩ đến việc xem xét các lân cận theo các hướng (8 hướng chính) - đó chính là phương pháp Kirsh và gọi là toán tử Kirsh hay toán tử la bàn (Compass Operator). Toán tử này được trình bày trong phần dưới đây.

### 5.3.1.2. Toán tử la bàn

Toán tử la bàn do gradient theo 1 số hướng đã chọn. Nếu kí hiệu  $g_i$  là gradient la bàn theo hướng  $\theta_i = \pi/2 + 2k\pi$  với  $k = 0, 1, \dots, 7$ . Như vậy ta do gradient theo tám hướng ngược chiều kim đồng hồ, mỗi hướng cách nhau  $45^\circ$  theo ngược chiều kim đồng hồ.

Có nhiều toán tử la bàn khác nhau. Nhưng ở đây, ta sẽ chỉ trình bày một cách chi tiết toán tử Kirsh. Toán tử này sử dụng mảng  $3 \times 3$ :

$$H_1 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

cho hướng gốc (hướng  $0^\circ$ ). Trên cơ sở đó định nghĩa thêm 7 mảng nay khác nhau từ  $H_2$  đến  $H_8$  cho 7 hướng còn lại:  $45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ$ .

$$H_2 = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad H_3 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad H_4 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

$$H_5 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad H_6 = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad H_7 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$$

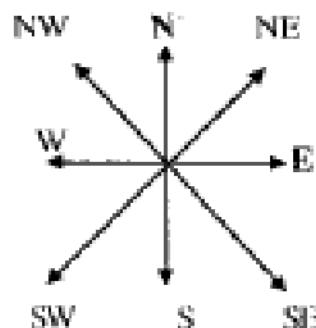
và

$$H_8 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

Nếu ta kí hiệu  $A_i$ ,  $i=1, 2, \dots, 8$  là gradient thu được theo 8 hướng bởi 8 mảng nay, biến độ gradient tại  $(x,y)$  sẽ được tính theo công thức 5-6, có nghĩa là:

$$A(x,y) = \text{Max}(|g_i(x,y)|, i=1, 2, \dots, 8) \quad (5.7)$$

Ngoài toán tử Kirsh, người ta còn sử dụng một số toán tử khác mà mảng nay ở hướng  $0^\circ$  được định nghĩa tương ứng bởi:



Hình 5.5. Mô hình 8 hướng.

$$M_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad M_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad M_3 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Các hướng khác, bạn đọc tự suy diễn tương tự như toán tử Kirsh và coi đó như bài tập tự giải. Để giảm tính toán, có thể chỉ cần tính cho 4 hướng ( $0^\circ, 45^\circ, 90^\circ$  và  $135^\circ$ ) với các mặt nạ sau:

$$\cdot \begin{bmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ -1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Trong trường hợp tổng quát, giả sử có n hướng cách đều tương ứng với các mặt nạ  $W_i$ ,  $i=1, \dots, n$  đối với ảnh I, khi đó:

$$A(x,y) = \text{Max}(|W_i^T I(x,y)|, i=1, 2, \dots, n).$$

Thực chất đây chính là chuẩn A<sub>2</sub>.

Chú ý rằng nếu đầu ra gần bằng 0, điều đó có nghĩa là không có điểm biến tại vị trí này. Nếu các  $|W_i^T I|$  có giá trị gần nhau thì các thông tin này không đáng tin cậy để xác định (x,y) là điểm biến. Thuật toán tính tương ứng với kỹ thuật này khá đơn giản vì chỉ cần một thủ tục có tham số là hướng tính.

### 5.3.2. Kỹ thuật Laplace

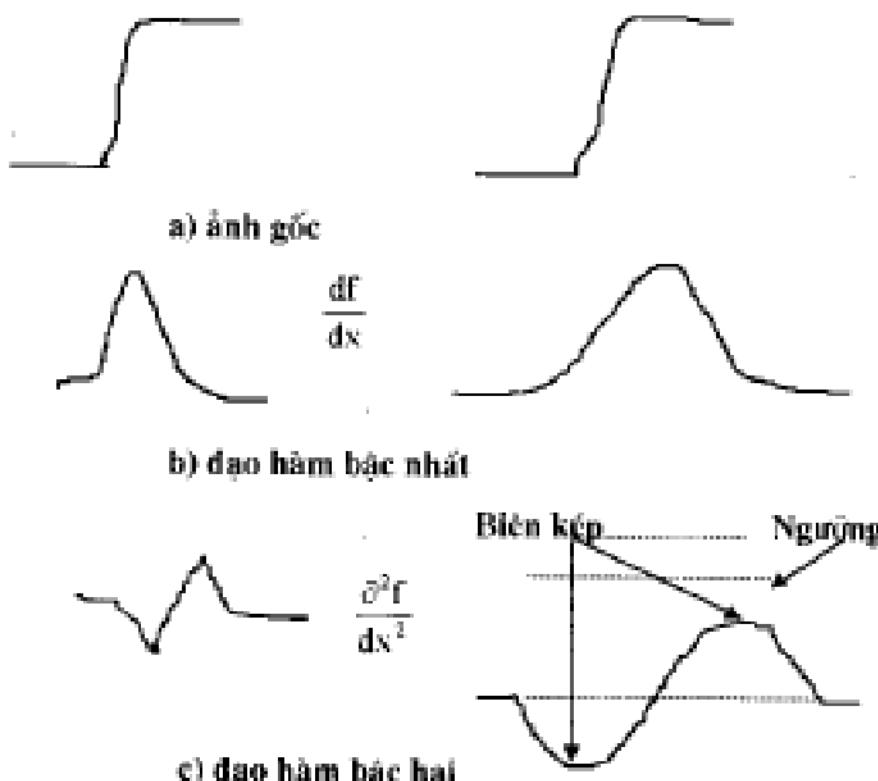
Các phương pháp đánh giá gradient ở trên làm việc khá tốt khi mà độ sáng thay đổi rõ nét. Khi mức xám thay đổi chậm, miền chuyển tiếp trôi rộng, phương pháp cho hiệu quả hơn đó là phương pháp sử dụng đạo hàm bậc hai mà trong phần trên gọi là phương pháp Laplace. Toán tử Laplace được định nghĩa như sau:

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



Hình 5.6. Ảnh qua lọc Sobel.

Hình 5.7 dưới đây cung cấp một hình ảnh về sử dụng đạo hàm bậc nhất và đạo hàm bậc hai trong việc xác định biên.



Hình 5.7. Ý nghĩa của đạo hàm bậc nhất và bậc 2 trong dò biên.

Kết quả nghiên cứu cho thấy trong phương pháp đạo hàm bậc 2, toán tử gradient rất nhạy cảm với nhiễu và tạo nên biên kép. Để khắc phục nhược điểm này, người ta mở rộng toán tử Laplace và dùng xấp xỉ Laplace-Gauss để công cụ hữu hiệu phát hiện điểm không:

$$h(m,n) = c [1 - (m^2 + n^2)/\sigma^2] \exp(-(m^2 + n^2)/2\sigma^2)$$

với  $\sigma$  là tham số điều khiển độ rộng và  $c$  là chuẩn tổng các phần tử có kích thước mặt nạ là đơn vị. Các điểm không (zero-crossing) của ảnh cho trước chụp với  $h(m,n)$  sẽ cung cấp cho ta vị trí biên của ảnh. Ta nói, trên một lưới 2 chiều, các điểm không nói là có xảy ra nếu có hiện tượng cắt điểm không theo ít nhất một hướng nào đó.

Toán tử Laplace dùng nhiều kiểu mặt nạ khác nhau để xấp xỉ rồi rắc đạo hàm bậc hai. Dưới đây là 3 kiểu mặt nạ hay dùng.

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad H_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad H_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Với mặt nạ  $H_1$ , đối khi người ta dùng phần tử ở tâm có giá trị là 8 thay vì 4 như đã chỉ ra.

Để giúp bạn đọc dễ dàng hình dung việc xấp xỉ đạo hàm bậc hai trong không gian rời rạc bởi mặt nạ  $H_1$  hay là ý nghĩa của mặt nạ  $H_1$ , ta xét chi tiết cách tính đạo hàm bậc hai.

Trong không gian rời rạc, đạo hàm bậc hai có thể tính:

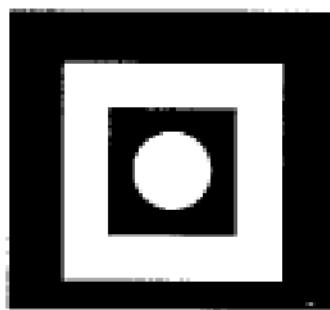
$$\frac{\partial^2 f}{\partial x^2} = 2f(x,y) - f(x-1,y) - f(x+1,y) \quad (5.8)$$

$$\frac{\partial^2 f}{\partial y^2} = 2f(x,y) - f(x,y-1) - f(x,y+1) \quad (5.9)$$

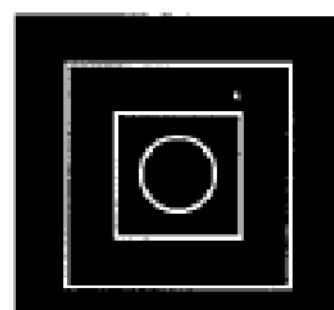
Vậy  $\nabla^2 f = -f(x-1,y) - f(x,y-1) + 4f(x,y) - f(x,y+1) - f(x+1,y)$  (5.10)

Công thức 5-10 nhận được như kết quả nhận chập ảnh  $f(x,y)$  với mặt nạ  $H_1$ . Bạn đọc nếu quan tâm hãy tự lý giải cách xấp xỉ đạo hàm bậc hai bởi mặt nạ  $H_2$  và  $H_3$  và coi đó như bài tập. Ảnh thu được sau khi áp dụng toán tử Laplace được minh họa qua hình 5-8 và hình 5.9 (trang bên).

Trong kỹ thuật Laplace, điểm biến được xác định bởi điểm cắt điểm không. Và điểm không là duy nhất do vậy, kỹ thuật này cho đường biến mảnh, tức là đường biến có độ rộng 1 pixel. Tuy nhiên, như đã nói ở trên, kỹ thuật Laplace rất nhạy cảm với nhiễu do đạo hàm bậc hai thường không ổn định.



a) ảnh gốc

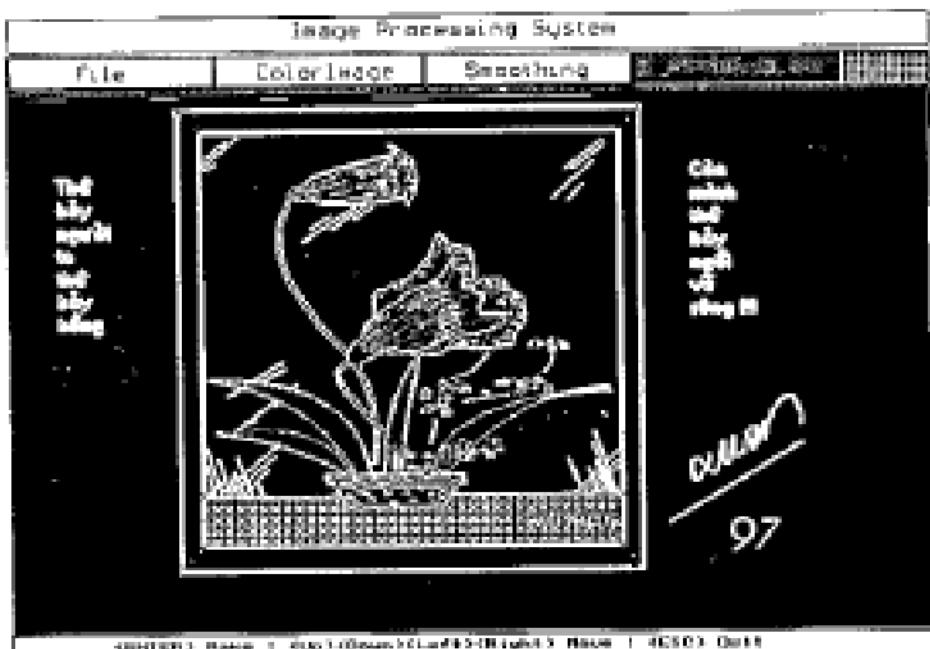


b) ảnh thu được bởi laplace hoá

Hình 5.8. Toán tử laplace hoá(ví dụ 1).

### 5.3.3. Kỹ thuật đạo hàm tích chập - phương pháp Canny

Phương pháp này do Canny ở phòng thí nghiệm MIT khởi xướng. Phương pháp này thực hiện bằng cách lấy đạo hàm của một ảnh chập với bộ lọc Gauss. Bộ lọc này được đưa ra ở chương trước. Để đơn giản cách viết, ta tạm bỏ đi phần tử  $1/(2\pi)^2$ .



Hình 5.9. Ảnh thu được sau khi dùng toán tử Laplace (ví dụ 2).

Đoạn hàm của một lệnh được lọc:

$\nabla f = \nabla(G \otimes I) = f_x + f_y$ , với  $f_x, f_y$  là đạo hàm theo x và y của f.

Do why:

$$\nabla f = \nabla(G \otimes I)_{\perp} + \nabla(G \otimes I)_{\parallel} = (G_x \otimes I) + (G_y \otimes I) \quad (5.11)$$

Lấy đạo hàm riêng theo x và y của G ta được:

$$G(x,y) = \frac{-x}{\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (5.12)$$

$$G(x,y) = \frac{-y}{\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (5.13)$$

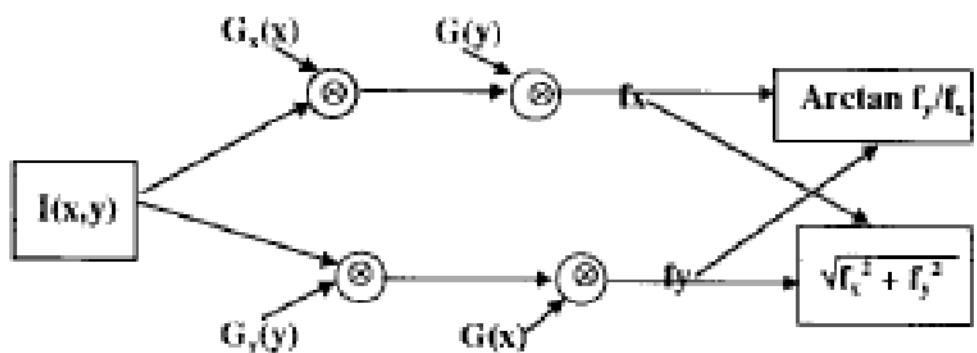
Hơn nữa bộ lọc Gauss là tách được, do vậy ta có thể thực hiện riêng biệt các tích chập theo x và y:

$$G_{\perp}(x,y) = G_{\perp}(x) \otimes G(y) \quad \text{và} \quad G_{\perp}(x,y) = G_{\perp}(y) \otimes G(x) \quad (5.14)$$

Tù 5-11 và 5-14 (a cõi)

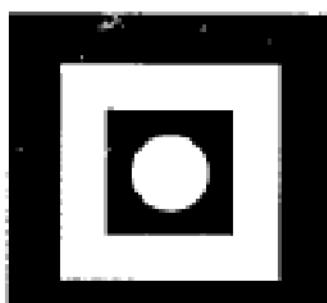
$$f(x,y) = G_x(x) \otimes G_y(y) \otimes I \quad \forall x, y \in \mathbb{R} \quad f(x,y) = G_y(y) \otimes G_x(x) \otimes I \quad (5.15)$$

với biên độ và hướng tĩnh theo công thức 5-2 và 5-3. Thuật toán được minh họa trên hình 5-10.

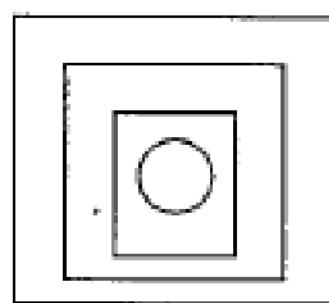


Hình 5.10. Mô hình tính của phương pháp Canny.

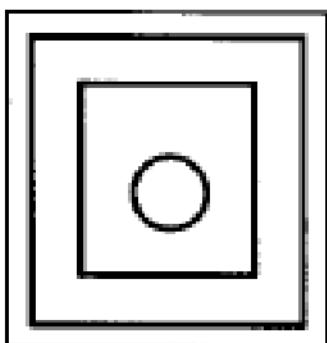
Để kiểm tra tính nhạy cảm của toán tử với nhiễu, người ta thay đổi giá trị  $\sigma$  của đường cong Gauss. Kết quả được minh họa trong hình 5-11.



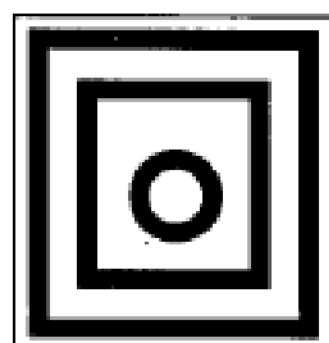
a) Ảnh gốc



b) Ảnh thu được với kích thước 3



c) Ảnh kết quả với kích thước 11



d) Ảnh kết quả với kích thước 21

Hình 5.11. Ảnh thu được theo phương pháp Canny.

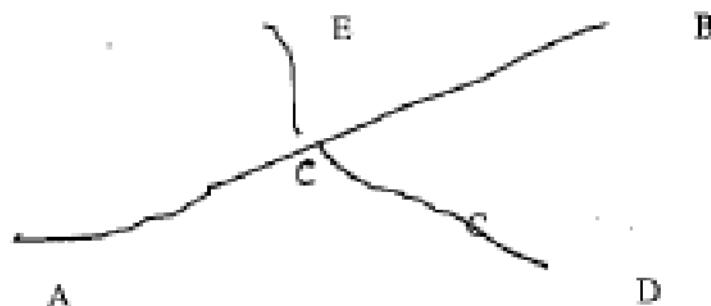
## 5.4. DÒ BIÊN THEO QUY HOẠCH ĐỘNG

### 5.4.1. Mô tả phương pháp

Như trên đã nói, dò biên theo phương pháp gradient là xác định cực trị cục bộ của gradient theo các hướng; còn phương pháp laplace dựa vào các điểm không của đạo hàm bậc

hai. Phương pháp đồ biến theo quy hoạch động là phương pháp tìm cực trị tổng thể của các quá trình nhiều bước. Nó dựa vào nguyên lý tối ưu của Bellman. Nguyên lý này phát biểu như sau: "Con đường tối ưu giữa 2 điểm cho trước cũng là tối ưu giữa 2 điểm bất kỳ nằm trên đường tối ưu đó".

Thí dụ, nếu C là một điểm trên con đường tối ưu giữa A và B thì đoạn CB cũng là con đường tối ưu từ C đến B không kể đến ta đến C bằng cách nào.



Hình 5.12. Minh họa nguyên lý Bellman.

Trong kỹ thuật này, ta giả sử rằng bản đồ biến đã được xác định và được biểu diễn dưới dạng một đồ thị liên thông N chặng. Hơn nữa, giả sử hàm đánh giá được tính theo công thức:

$$S(x_1, \dots, x_N, N) = \sum_{k=1}^N |g(x_k)| - \alpha \sum_{k=1}^N |\theta(x_k) - \theta(x_{k-1})| - \beta \sum_{k=1}^N d(x_k, \bar{x}_{k-1}) \quad (5.16)$$

với:

-  $x_k$ ,  $k=1, \dots, N$ : biểu diễn các đỉnh của đồ thị trong chặng thứ k;

-  $d(x, y)$ : khoảng cách giữa 2 đỉnh x và y tính theo các định nghĩa tương ứng về khoảng cách;

-  $|g(x_k)|$  và  $\theta(x_k)$  là gradient biến độ và gradient hướng ở đỉnh  $x_k$ ;

-  $\alpha$  và  $\beta$  là các tham số không âm.

Đường bao tối ưu sẽ nhận được bằng cách nối các đỉnh  $\bar{x}_k$ ,  $k=1, \dots, N$  nào đó thoả mãn, sao cho  $S(x_1, \dots, x_N, N)$  đạt cực đại.

Ta định nghĩa hàm  $\phi$  như sau:

$$\phi(x_N, N) = \max_{x_1, \dots, x_{N-1}} \{S(x_1, \dots, x_N, N)\} \quad (5.17)$$

Bây giờ viết lại công thức 5-16 một cách dễ quy ra cột:

$$S(x_1, \dots, x_N, N) = S(x_1, \dots, x_{N-1}, N-1) + |g(x_N)| - \alpha |\theta(x_N) - \theta(x_{N-1})| - \beta d(x_N, \bar{x}_{N-1}) \quad (5.18)$$

đặt  $f(x_{n+1}, x_n) = |g(x_n)| - \alpha |\theta(x_i) \cdot \theta_{\lambda}(x_{n+1})| - \beta d(x_n, x_{n+1})$  và thay vào 5-18 ta có:

$$S(x_1, \dots, x_N, N) = S(x_1, \dots, x_{N-1}, N-1) + f(x_{N-1}, x_N) \quad (5.19)$$

Lấy  $N = k$  trong 5-17 và 5-19 và thực hiện suy diễn ta có:

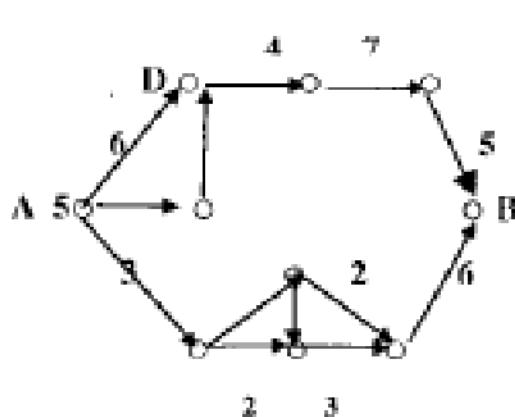
$$\begin{aligned} \phi(x_k, k) &= \max_{x_1, \dots, x_{k-1}} \{S(x_1, \dots, x_k, k) + f(x_{k-1}, x_k)\} \\ &= \max_{x_1} \{\phi(x_{k-1}, k-1) + f(x_{k-1}, x_k)\} \text{ với } k = 2, \dots, N \end{aligned} \quad (5.20)$$

$$\text{Như vậy, } S(\bar{x}_1, \dots, \bar{x}_N, N) = \max_{x_1} \{\phi(x_k, N)\} \text{ với } \phi(x_1, 1) = |g(x_1)| \quad (5.21)$$

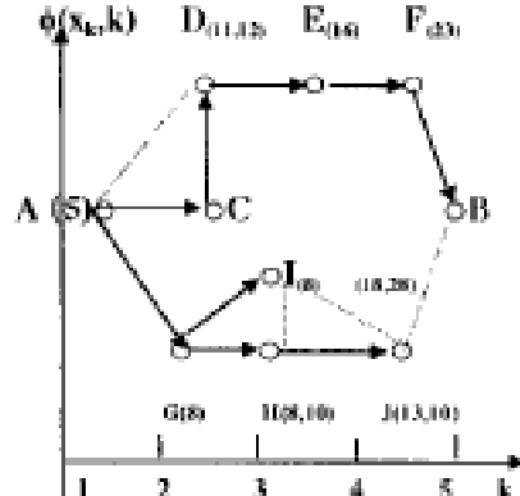
Với cách thức này, thay vì tìm tối ưu toàn cục của  $S(x_1, \dots, x_N, N)$ - một vấn đề phức tạp, ta đã tìm tối ưu của  $N$  chặng của tối ưu 2 biến. Trong mỗi chặng, với mỗi  $x_k$  ta phải tìm tối ưu  $\phi(x_k, k)$ . Để hình dung dễ dàng, ta xét thí dụ sau:

- Giả sử ta có bản đồ biến biểu diễn bởi đồ thị liên thông sau (xem hình 5.13):

Theo phương pháp trên ta có  $\phi(A, 1) = 5$ . Với  $k = 2$  có  $\phi(D, 2) = \max(11, 12) = 12$ . Điều đó có nghĩa là con đường từ A đến D đi qua C và ACD là biến được chọn với  $k=2$ . Một cách tương tự, với  $k=4$  có 2 con đường được chọn là ACDEF và AGHJ. Tuy nhiên, với  $k=5$  thì đoạn JB bị loại và chỉ tồn tại con đường duy nhất với cự ly dài là 28. Như vậy biến được xác định là ADEFFB.



a) Đồ thị liên thông biến diễn biến



b) Quá trình dò biến theo quy hoạch động

Hình 5.13. Dò biến theo quy hoạch động,

Trên đồ thị hình 5.13b, những đường nét đứt đoạn biểu thị cung bị loại; đường nét liền có mũi tên biểu thị đường đi hay biến của ảnh.

### 5.4.2. Thuật toán

Theo phân tích trên, thuật toán dò biên theo qui hoạch động được mô tả một cách hình thức như sau:

- Xuất phát từ điểm ban đầu  $X_A$ , ta xác định con đường gồm các đỉnh thoả 5.20 để đạt đến đỉnh  $X_B$ .
- Việc dò tìm này giống như phép duyệt đồ thị theo kề sau: tại điểm hiện thời  $x_i$  (lúc xuất phát là  $X_A$ ), ta xác định các điểm kế tiếp, dựa vào 8 lân cận theo hướng của điểm hiện tại và điểm trước đó. Nếu hướng đó hợp lệ, ta chọn nó như một đỉnh trên con đường cần tìm. Ngược lại, để tìm đỉnh có khả năng ta sẽ chọn 1 trong 8 lân cận của đỉnh đang xét với chi phí lớn nhất (tiêu chí max). Có thể có nhiều đỉnh thoả mãn và như vậy ta cần lưu trữ các đỉnh này coi như các *đỉnh có khả năng*. Nếu không có đỉnh nào được chọn trong 8 lân cận đó, ta phải quay lui và chọn đỉnh khác đã lưu trữ.
- Quá trình được lặp lại cho đến đỉnh được xem xét hiện tại chính là đích  $X_B$ .

Để cài đặt giải thuật này, cần sử dụng các cấu trúc dữ liệu và các thủ tục sau:

- Stack winerstack: Lưu trữ các đỉnh trên con đường tối ưu từ đỉnh xuất phát  $X$  đến đỉnh đích  $X_B$ .
- Stack Tempstack: Lưu trữ các đỉnh *có khả năng* để xem xét lúc quay lui một khi thất bại tạm thời.
- Véc-tơ Index: gồm 9 phần tử. Mỗi phần tử của véc-tơ gồm 3 trường: hàng, cột và giá. Giá là tiêu chí xét đỉnh thoả 5.20; hàng và cột xét theo hướng.
- ImagIn : ảnh vào.
- $X_A, X_B, \text{Last}$ : có cấu trúc như một phần tử của véc-tơ Index.
- **Find\_Direc (last,current)**: Hàm cung cấp giá trị về hướng của điểm đang xét(current) và điểm trước của nó(last). Các giá trị có thể từ 0 đến 8. Hướng được đánh số như sau:

0	1	2
3	last	5
6	7	current

Hàm trả về giá trị -1 nếu không thuộc hướng trên.

- Thủ tục **Find\_Successor(ImagIn, X, Last, Index)**: Tìm các điểm tiếp theo có thể và đặt vào véc-tơ Index. Điểm tiếp theo này phải nằm trong giới hạn của ảnh, khác với điểm hiện thời (current) và điểm trước đó (last) và cưỡng độ mức xám phải lớn hơn ngưỡng T. Khi điểm ảnh được chọn, trường giá sẽ tăng lên một lượng; bảng mức xám của điểm ảnh đó.

- Thủ tục **Init\_Index(Index)**: Đặt lại giá trị cho mỗi phần tử của véc-tơ Index bằng -1 mỗi khi khởi tạo một đợt tìm mới.

- Thủ tục **Sort\_Pixel (Index)**: Sắp xếp lại véc-tơ index theo trường giá tăng dần để tìm các đỉnh có khả năng làm cực đại harm φ.

Giải thuật dò biên theo qui hoạch động được viết như sau:

**Edge\_Dynamic( X<sub>A</sub>, X<sub>B</sub>, ImagIn, T, Index, Last, WinerStack)**

/\* X<sub>A</sub>, X<sub>B</sub>, T, ImagIn: các tham số vào.

WinerStack: tham số ra. Chứa các đỉnh biên trên đường từ X<sub>A</sub> đến X<sub>B</sub> \*/

1. If X<sub>A</sub> = X<sub>B</sub> then KetThuc;

2. Khởi tạo Ban đầu

Last = X<sub>A</sub>;

Push(WinerStack,X<sub>A</sub>);

ImagIn[XA] = 0 ; [Đánh dấu để không xét lại]

Init\_Index(Index);

Find\_Successor(ImagIn,X,T,Index,Last);

Sort\_Pixel(Index);

If Index[8] = -1 then {điểm hiện tại không có điểm tiếp theo - đã xét hết}

Begin Xi <- tiếp theo trong Index

    Init\_Index(Index);

end

3. Vòng lặp chính

While True do

Begin

```

Push(WinerStack,Xi);
ImagIn[Xi] = 0;
If Xi = Xn then EXIT; { Kết thúc}
Find_Successor(ImagIn,Xi,T,Index,Last);
d = Find_Direc(last,Xi);
Last = Xi;
if (d > 0) & (ImagIn[Index[d]] là thoát Then Xi = Index[d]);
Sort_Pixel(Index);
/* xác định đỉnh có khả năng */
For i=1 to 8 do
    If Index[i].gia <> -1 then Push(TempStack,Index[i]);
If Index[8] <> -1 & ImagIn[Index[8]] = 0
    then Xi=Index[8];
Else
    While (TempStack not Empty) & (ImagIn[Xi] = 0) do
        Begin Xi = Pop(TempStack);
        If TempStack is Empty then QuayLai;
        While (WinerStack not Empty and Xi <> Xi) do
            Xi = POP(WinerStack);
        end
    Init_Index(Index);
end

```

#### 4. KetThuc.

Chú ý rằng giải thuật này sẽ xét điểm kế tiếp nằm trên đường nối điểm đang xét với điểm trước nó trước tiên. Nếu không thoả, tất cả các điểm kế tiếp sẽ được xét bằng cách sắp xếp theo trường giá tăng dần (thủ tục Find\_Succesor). Điểm được chọn là điểm có giá cao nhất (Index[8].gia). Trong trường hợp ngược lại, giải thuật quay lui để tìm đường khác nhau vào các đỉnh đã lưu trong TempStack. Giải thuật kết thúc và thành công khi điểm xét là điểm đích X<sub>n</sub>. Giải thuật thất bại khi TempStack trở thành rỗng.

## 5.5. MỘT SỐ PHƯƠNG PHÁP KHÁC

Ngoài các phương pháp đã trình bày trên, trong việc xử lý ảnh người ta cũng áp dụng một số phương pháp khác tân tiến: cách tiếp cận bởi mô hình mặt, cách tiếp cận tối ưu hóa.

Cách tiếp cận theo mô hình mặt dựa vào việc thực hiện xấp xỉ đa thức trên ảnh gốc hay ảnh đã thực hiện phép lọc Laplace. Cách tiếp cận tối ưu nhằm xác định một hàm (một bộ lọc), làm giảm phương sai  $\sigma^2$  hoặc giảm một số điểm cực trị cục bộ. Dưới đây sẽ trình bày một cách tóm lược các phương pháp đó.

### 5.5.1. Tiếp cận theo mô hình mặt

Ý tưởng cách tiếp cận này là tại lân cận điểm cắt không (điểm biên), ảnh sau khi lọc Laplace có thể được xấp xỉ bởi một đa thức bậc 3 theo hàng và cột. Đa thức thường được dùng là đa thức Trebuchép với lân cận 3x3. Các đa thức này được định nghĩa như sau:

$$P_0(x,y) = 1 \quad P_1(x,y) = x \quad P_2(x,y) = y \quad P_3(x,y) = x^2 - \frac{2}{3}$$

$$P_4(x,y) = xy \quad P_5(x,y) = y^2 - \frac{2}{3} \quad P_6(x,y) = xP_3(x,y) \quad P_7(x,y) = yP_3(x,y)$$

$$\text{và } P_8(x,y) = P_3(x,y)P_3(y,x)$$

Với mỗi điểm cắt không phát hiện tại  $P(x,y)$  trong ảnh đã được lọc bởi toán tử Laplace -Gaus-Huertas và Medioni tính xấp xỉ theo công thức:

$$I_{L-G}(x,y) = \sum_{n=0}^{N-1} a_n P_n(x,y)$$

Vấn đề là xác định các hệ số  $a_i$ ,  $i=1, 2, \dots, N-1$ . Nếu  $W$  là cửa sổ lọc tại điểm cắt không và  $x, y, i, j$  ở trong cửa sổ, các hệ số  $a_i$  có thể được tính như một tổ hợp tuyến tính:

$$a_i = \frac{\sum_{x=i}^N \sum_{y=j}^M P_n(x,y) I_{L-G}(x,y)}{\sum_{i=1}^N \sum_{j=1}^M P_n^2(i,j)}$$

$I_{L-G}(x,y)$  ký hiệu ảnh đã được lọc bởi toán tử Laplace-Gauss. Các hệ số này có thể nhận được bởi chập ảnh  $I_{L-G}(x,y)$  với các nhân chập như trung bình có trọng số hay một số nhân chập khác.

Các bước cài đặt phương pháp này có thể mô tả như sau:

- Chập ảnh gốc kích thước  $N \times M$  với toán tử Laplace-Gaus kích thước  $M^2$ , ảnh thu được gọi là  $I_{L-G}$ .

2. Trích chọn các điểm cắt không của ảnh  $I_{\text{L}G}$ . ảnh kết quả ký hiệu là  $I_{\text{ROI}}$ .
3. Với mỗi điểm cắt không trong  $I_{\text{ROI}}$ , thực hiện một xấp xỉ trong lán canh  $3 \times 3$  để suy ra các điểm cắt không theo cách giải tích.
4. Tạo một ảnh mới của các điểm cắt không kích thước  $nXnY$  mà các đường bao được xác định với độ phân giải  $n$  nào đó.

### 5.5.2. Tiếp cận tối ưu hóa

Ý tưởng của cách tiếp cận này là định vị đúng vị trí bằng cách cực tiểu hóa phương sai  $\sigma^2$  của vị trí các điểm cắt không hoặc hạn chế số điểm cực trị bộ để chỉ tạo ra một đường bao. Canny đã đề xuất 3 ràng buộc tương ứng với 3 điều kiện:

$$\Sigma = \frac{A \int_{-\infty}^{\infty} h(x) dx}{n_0 \sqrt{\int_{-\infty}^{\infty} h^2(x) dx}}$$

$$\Lambda = \frac{A |h'(0)|}{n_0 \sqrt{\int_{-\infty}^{\infty} h^2(x) dx}}$$

$$x_{\text{min}} = 2\pi \sqrt{\frac{\int_{-\infty}^{\infty} h'^2(x) dx}{\int_{-\infty}^{\infty} h'''^2(x) dx}}$$

Ràng buộc  $\Sigma$  nhằm tìm một hàm  $h(x)$  phản đối xứng sao cho tỉ số giữa tín hiệu và nhiễu là cực đại. Ràng buộc thứ hai  $\Lambda$  nhằm cực tiểu hóa phương sai. Còn ràng buộc thứ ba nhằm hạn chế điểm cực trị bộ với mục đích cung cấp chỉ một đường bao.

Nhìn chung đây là một vấn đề phức tạp, chúng tôi chỉ dùng lại ở mục đích giới thiệu qua cho độc giả.

**Bài tập chương 5****Câu 1**

- 1) Xác định các mặt nạ theo các hướng còn lại ứng với mặt nạ  $M_1$ ,  $M_2$  và  $M_3$ .
- 2) Hãy viết biểu thức tính đạo hàm bậc hai một cách tường minh theo mặt nạ  $H_1$ ,  $H_2$ .
- 3) Cho ảnh số I:

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

- a) Hãy tính  $G = |G_x| + |G_y|$  với  
 $G_x = H_x \otimes I$  và  $G_y = H_y \otimes I$ ;  $H_x$ ,  $H_y$  là nhân chập Prewitt.

- b) Hãy tính Laplace của ảnh:  $I_L = H_L \otimes I$

**Câu 2**

- 1) Viết thủ tục phát hiện biên ảnh dùng toán tử Robert.
- 2) Viết thủ tục phát hiện biên ảnh dùng toán tử Sobel.
- 3) Viết thủ tục phát hiện biên ảnh dùng toán tử Prewitt..
- 4) Viết một thủ tục tính Laplace của ảnh đã cho với kiểu mặt nạ tự chọn như tham số ( $H_1$ ,  $H_2$ ,  $H_3$ ).
- 5) Viết một thủ tục thực hiện việc dò biên theo thuật toán Canny.
- 6) Xây dựng thủ tục dò biên theo qui hoạch động dựa vào thuật toán đã cho. Xây dựng các hàm và thủ tục con .

# 6

## PHÂN VÙNG ẢNH (SEGMENTATION)

### 6.1. TỔNG QUAN VỀ PHÂN VÙNG ẢNH

Phân vùng ảnh là bước then chốt trong xử lý ảnh. Giai đoạn này nhằm phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là cùng mức xám, cùng màu hay cùng độ nhám, v.v. Trước hết cũng cần làm rõ khái niệm "vùng ảnh" (segment) và đặc điểm vật lý của vùng.

Vùng ảnh là một chi tiết, một thực thể trong toàn cảnh. Nó là một tập hợp các điểm có cùng hoặc gần cùng một tính chất nào đấy: mức xám, mức màu, độ nhám, . . . Vùng ảnh là một trong hai thuộc tính quan trọng của ảnh. Nói đến vùng ảnh là nói đến tính chất bề mặt. Đường bao quanh một vùng ảnh (boundary) là biên ảnh. Các điểm trong một vùng ảnh có độ biến thiên giá trị mức xám tương đối đồng đều hay tính kết cấu (texture) tương đồng.

Dựa vào đặc tính vật lý của vùng ảnh, người ta có nhiều kỹ thuật phân vùng. Nếu phân vùng dựa trên các miền liên thông, ta gọi là kỹ thuật phân vùng dựa theo miền đồng nhất hay miền kẽ. Nếu ta phân vùng dựa vào biên gọi là kỹ thuật phân vùng biên. Ngoài ra, còn có các kỹ thuật khác như phân vùng dựa vào biên độ, phân vùng theo kết cấu.

Như đã trình bày trong chương trước, mục đích của phân tích ảnh là để có một miêu tả tổng hợp về nhiều phần tử khác nhau cấu tạo nên ảnh thô (brut image). Vì lượng thông tin chứa trong ảnh là rất lớn, trong khi đó đa số ứng dụng chỉ cần một số thông tin đặc trưng nào đó, do vậy cần có một quá trình để giảm lượng thông tin khổng lồ ấy. Quá trình này bao gồm phân vùng ảnh và trích chọn đặc tính chủ yếu. Các kỹ thuật dùng cho quá trình này sẽ được đề cập chi tiết dưới đây.

### 6.2. PHÂN VÙNG ẢNH DỰA THEO NGUỒNG BIÊN ĐỘ

Đặc tính đơn giản nhất và có thể hữu ích nhất của ảnh đó là biên độ của các tinh

chất vật lý của ảnh như: độ phản xạ, độ truyền sáng, màu sắc hoặc đáp ứng đa phổ. Thí dụ, trong ảnh X-quang, biên độ mức xám biểu diễn đặc tính bão hòa của các phần hấp thụ của cơ thể và làm cho ta có khả năng phân biệt xương với các phần mềm, tế bào lành với các tế bào bị nhiễm bệnh, v.v.

Như vậy, có thể dùng ngưỡng biên độ để phân vùng khi mà biên độ đủ lớn đặc trưng cho ảnh. Thí dụ biên độ trong bộ cảm biến ảnh hồng ngoại có thể phân ảnh vùng nhiệt độ thấp hay vùng có nhiệt độ cao hơn. Kỹ thuật phân ngưỡng theo biên độ rất có ích đối với ảnh nhị phân như văn bản in, đồ họa, ảnh màu hay ảnh X-quang.

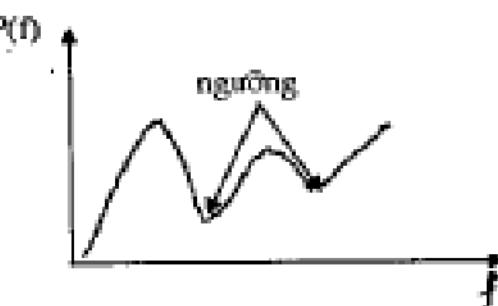
Việc chọn ngưỡng trong kỹ thuật này là bước rất quan trọng. Người ta thường tiến hành theo các bước chung sau:

- Xem xét lược đồ xám của ảnh để xác định các đỉnh và các khe. Nếu ảnh có dạng rắn luộn (nhiều đỉnh và khe), các khe có thể sử dụng để chọn ngưỡng.
- Chọn ngưỡng t sao cho một phần xác định trước  $\tau$  của toàn bộ số mẫu là thấp hơn t.
- Điều chỉnh ngưỡng dựa trên xem xét lược đồ xám của các điểm lân cận.
- Chọn ngưỡng như xem xét lược đồ xám của những điểm thoả tiêu chuẩn chọn. Thí dụ, với ảnh có độ tương phản thấp, lược đồ của những điểm có biên độ Laplace  $g(m,n)$  lớn hơn giá trị t định trước (sao cho từ 5% đến 10% số điểm ảnh với gradient lớn nhất sẽ coi như biên) sẽ cho phép xác định các đặc tính ảnh lưỡng cực tốt hơn ảnh gốc.
- Khi có một mô hình phân lớp xác suất, việc xác định ngưỡng dựa vào tiêu chuẩn nhằm cực tiểu xác suất của sai số hoặc một số tính chất khác theo luật Bayes.

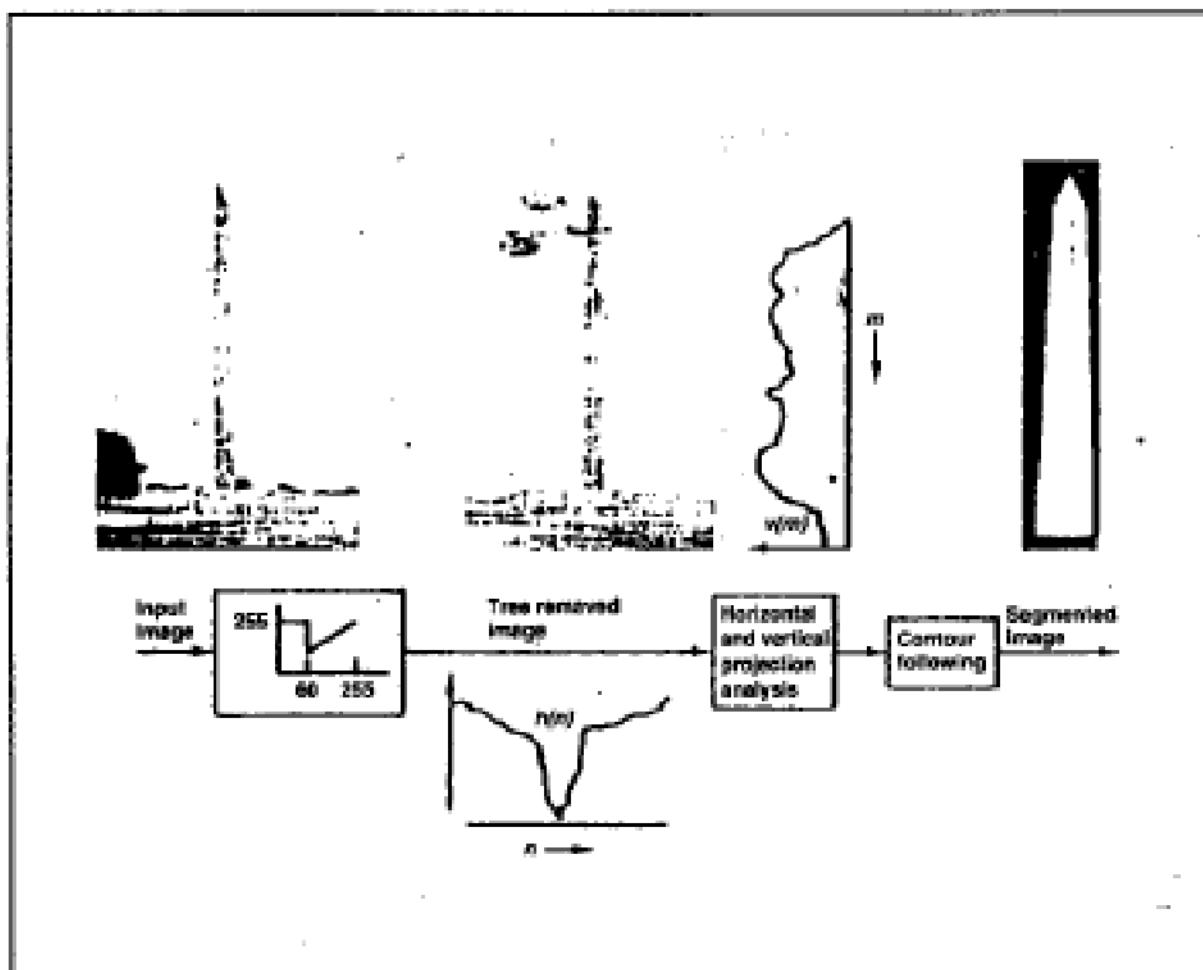
Thí dụ, ta muốn phân đoạn tượng dải Washington từ cảnh trong hình 6-1 dưới đây. Trước tiên, những điểm có cường độ thấp sẽ được phân ngưỡng để cô lập những vùng rất tối (đó là những cây trong ảnh) để loại cây ra khỏi ảnh. Tiếp theo ta xác định một hình chữ nhật bao quanh tượng dải nhờ phân ngưỡng theo hình chiếu đứng và chiếu bằng, được định nghĩa như sau:

$$h(n) = \sum_m u(m,n) / \sum_m 1$$

$$v(m) = \sum_n u(m,n) / \sum_n 1$$



với  $h(n)$ : hình chiếu bằng;  $v(m)$ : hình chiếu đứng.



Hình 6.1. Phân vùng sử dụng hình chiếu đứng và bằng.

Đường bao quanh đường biên đối tượng nằm bên trong hình chữ nhật cho ta ảnh được phân đoạn. Quá trình nhị phân hóa ảnh cũng chính là việc phân theo ngưỡng, tức là:

$$\text{Out}(x,y) = \begin{cases} 1 & \text{In}(x,y) > T \\ 0 & \text{In}(x,y) < T \end{cases} \quad T \text{ là ngưỡng cần xác định}$$

Trong một số trường hợp, ta cần biến đổi lược đồ xám theo công thức:

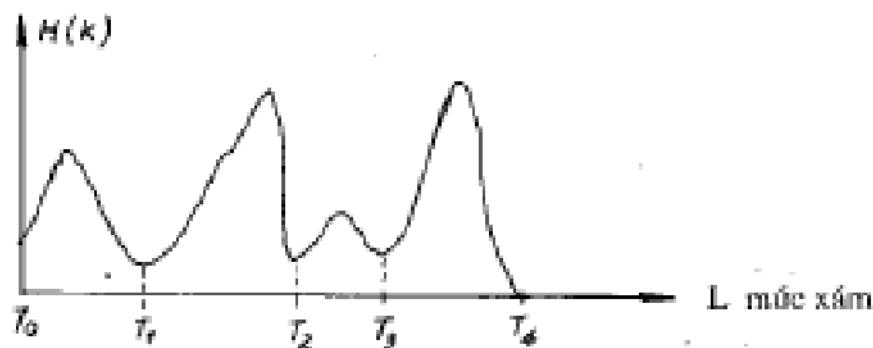
$$H(i) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} t(e(k,l)\delta(f(k,l) - i)) \quad (6.1)$$

với:  $e(k,l)$  là ảnh đầu ra của việc phát hiện biên;

$\delta(i)$  là hàm Delta;

$$t(e(k,l)) \text{ tính bởi: } t(e(k,l)) = \frac{1}{1 + |e(k,l)|}$$

Để hiểu rõ hơn nguyên tắc phân vùng dựa vào ngưỡng biến độ, ta xét thí dụ sau:



Hình 6.2. Lược đồ hình rán luron và cách chọn ngưỡng.

Giả sử ảnh có lược đồ xám như hình 6.2, ta chọn các ngưỡng như hình vẽ:  $T_0 = L_{min}$ , ...,  $T_4 = L_{max}$ . Ta có 5 ngưỡng và phân ảnh thành 4 vùng, ký hiệu  $C_k$  là vùng thứ k của ảnh,  $k=1, 2, 3, 4$ . Cách phân vùng theo nguyên tắc:

$$P(m,n) \in C_k \text{ nếu } T_{k-1} \leq P(m,n) < T_k, k=1, 2, 3, 4$$

Sau khi phân xong, nếu ảnh rõ nét thì việc phân vùng coi như kết thúc. Trường hợp ngược lại, cần điều chỉnh ngưỡng. Một trong các phương pháp quen biết là phương pháp ISODATA (Interactive Self Organizing Data Analysis) [9] cho phép điều chỉnh ngưỡng theo từng bước cho đến khi ta có một phân vùng như ý.

### 6.3. PHÂN VÙNG THEO MIỀN ĐỒNG NHẤT

Kỹ thuật phân vùng ảnh thành các miền đồng nhất dựa vào các tính chất quan trọng nào đó của miền. Việc lựa chọn các tính chất của miền sẽ xác định tiêu chuẩn phân vùng. Ở đây cũng cần phải xác định rõ tính đồng nhất của một miền của ảnh vì đó là điểm chủ yếu xác định tính hiệu quả của việc phân vùng. Các tiêu chuẩn hay được dùng là sự thuần nhất về mức xám, màu sắc đối với ảnh màu, kết cấu sợi và chuyển động.

Thí dụ, với ảnh hàng không, việc phân vùng theo màu cho phép phân biệt thảm thực vật: cánh đồng màu xanh hay vàng, rừng xanh thẳm, đường màu xám, mái nhà màu đe, v.v.

Đối với ảnh chuyển động, người ta tiến hành trả 2 ảnh quan sát được tại hai thời điểm

khác nhau. Trong trường hợp này, phần ảnh không thay đổi sẽ nhận giá trị không, những phần thay đổi sẽ nhận giá trị dương hay âm tương ứng với thay đổi hay dịch chuyển. Như vậy việc trừ ảnh thực ra là một xấp xỉ của đạo hàm theo thời gian của ảnh. Thực vậy, giả sử  $I(t)$  và  $I(t+\tau)$  là 2 ảnh quan sát ở thời điểm  $t$  và  $t+\tau$ . Nếu thời gian quan sát  $\tau$  là nhỏ, ta sẽ nhận được xấp xỉ của đạo hàm một cách trực tiếp:

$$\frac{\Delta I}{\Delta t} = \frac{I(t + \tau) - I(t)}{\tau}$$

Với cách tính này, ta có thể biết được vận tốc dịch chuyển của ảnh.

Cũng nhờ kỹ thuật trừ ảnh ta có thể xác định sự xuất hiện của những đối tượng mới (tín hiệu dương) hay sự biến mất của các đối tượng trong ảnh trước (tín hiệu âm).

Tính kết cấu (texture) là đặc tính rất quan trọng trong phân vùng ảnh. Nhờ nó, ta có thể phân biệt thảm cỏ với một mặt nhuộm màu xanh lá cây. Tính kết cấu đặc trưng cho kiểu dạng xuất hiện lặp trên bề mặt nào đó của đối tượng. Có 2 kiểu lặp: lặp có tính chu kỳ và lặp ngẫu nhiên. Lặp ngẫu nhiên thường gặp trong tự nhiên như cát, thảm cỏ; còn lặp có tính chu kỳ là lặp nhân tạo. Thí dụ như ảnh tạo bởi kỹ thuật phân nguội bằng ma trận Dithering trong chương 2 cho ảnh có kết cấu sợi.

Người ta có thể dùng logic vị từ để làm tiêu chuẩn đánh giá việc phân vùng. Giả sử ảnh  $X$  phải phân thành  $n$  vùng khác nhau, ký hiệu:  $Z_1, Z_2, \dots, Z_n$  và logic vị từ có dạng  $P(Z)$ . Việc phân vùng như vậy phải thoả các tính chất sau:

$$X = \bigcup_{i=1}^n Z_i \text{ với } i = 1, 2, \dots, n \quad (6.2)$$

$$Z_i \cap Z_j = \emptyset \quad (6.3)$$

$$P(Z_i) = \text{true} \text{ với } i = 1, 2, \dots, n \quad (6.4)$$

và  $P(Z_i \cup Z_j) = \text{false}$  (6.5)

Kết quả của việc phân vùng ảnh phụ thuộc vào dạng của vị từ  $P$  và các đặc tính biểu diễn bởi véc-tơ đặc tính. Thường vị từ  $P$  có dạng  $P(Z, X, t)$ , với  $X$  là véc-tơ đặc tính;  $t$  là nguội. Trường hợp đơn giản nhất, véc-tơ đặc tính chỉ chứa giá trị mức  $I(k, l)$  của ảnh và nguội chỉ đơn thuần là giá trị  $T$ .

$$P(Z) : I(k, l) < T.$$

Với ảnh màu, vectơ đặc tính X có thể là thành phần 3 màu R, G, B và  $I_R(k,l)$ ,  $I_G(k,l)$ ,  $I_B(k,l)$  là các thành phần tương ứng. Lúc đó luật phân ngưỡng có dạng:

$$P(Z,X,t) : I_R(k,l) < T_R \quad \text{và} \quad I_G(k,l) < T_G \quad \text{và} \quad I_B(k,l) < T_B \quad (6.6)$$

Có 3 cách tiếp cận chủ yếu trong phân vùng ảnh theo miền đồng nhất và độc lập với tiêu chuẩn chọn lựa tính đồng nhất:

- Phương pháp phân tách - cây tử phân (split - quad trees).
- Phương pháp hợp (merge).
- Phương pháp tách-hợp( split-merge).

Mức độ hiệu quả của các phương pháp là phụ thuộc vào việc chọn tiêu chuẩn đánh giá độ thuần nhất. Trên thực tế người ta hay sử dụng trung bình số học  $m_i$  và độ lệch chuẩn  $\sigma_i$  cho vùng  $Z_i$  có  $n_i$  điểm:

$$m_i = \frac{1}{n_i} \sum_{(k,l) \in Z_i} I(k,l) \quad (6.7)$$

$$\sigma_i = \sqrt{\frac{1}{n_i} \sum_{(k,l) \in Z_i} (I(k,l) - m_i)^2} \quad (6.8)$$

với  $n_i$  là số điểm của vùng  $i$ .

Hai vùng  $Z_i$  và  $Z_j$  có thể hợp nhất nếu:  $|m_i - m_j| < k\sigma_i$ , vùng  $Z_i$  được coi là thuần nhất nếu  $\sigma_i < T$  ( $T$  là ngưỡng).

### 6.3.1. Phương pháp tách cây tử phân

Về nguyên tắc, phương pháp này kiểm tra tính hợp thức của tiêu chuẩn một cách tổng thể trên miền lớn của ảnh. Nếu tiêu chuẩn được thỏa, việc phân đoạn coi như kết thúc. Trong trường hợp ngược lại, ta chia miền đang xét thành 4 miền nhỏ hơn. Với mỗi miền nhỏ, ta lại áp dụng một cách đệ quy phương pháp trên cho đến khi tất cả các miền đều thỏa. Phương pháp này có thể mô tả bởi thuật toán sau (lưu ý rằng tham số ở đây là miền đang xét):

Procedure PhanDown (Mien)

Begin

if miền đang xét không thỏa Then

Begin

Chia miền đang xét thành 4 miền : Z1, Z2, Z3 và Z4

```

For i = 1 to 4 do      PhanDoan(Zi)
End
Else      exit
End

```

Tiêu chuẩn xét miền đồng nhất ở đây có thể dựa vào mức xám. Có thể dựa vào độ lệch chuẩn (6-8) hay độ chênh giữa giá trị mức xám lớn nhất và giá trị mức xám nhỏ nhất. Giá trị Max và Min là giá trị mức xám lớn nhất và nhỏ nhất trong miền đang xét. Nếu:

$$| \text{Max} - \text{Min} | < T \text{ (ngưỡng)}$$

ta coi miền đang xét là miền đồng nhất, trong trường hợp ngược lại, miền đang xét là không thuần nhất và sẽ được chia làm 4 phần.

Thuật toán kiểm tra tiêu chuẩn dựa vào độ chênh max, min được viết:

```

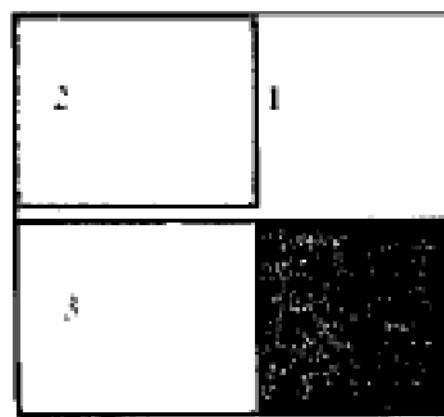
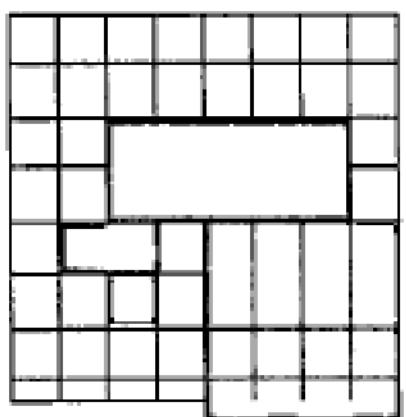
Function Examin_Criteria(I,N1,M1,N2,M2,T)
/* Giá thiết là ảnh có tối đa 255 mức xám.
(N1,M1),(N2,M2) là tọa độ điểm đầu và điểm cuối của miền, T là ngưỡng. */
Begin
1. Max =0; Min = 255;
2. For i = N1 to N2 do
    if I[i,j] < Min then Min = I[i,j];
    if I[i,j] > Max then Max = I[i,j];
3. If ABS( max - Min ) < T Then Examin_Criteria = 0
    Else Examin_Criteria = 1;
End

```

Nếu hàm trả về giá trị 0, có nghĩa vùng đang xét là đồng nhất, trường hợp ngược lại nghĩa là mức không đồng nhất. Trong giải thuật trên, khi miền là đồng nhất cần tính lại giá trị trung bình và cập nhật lại ảnh đầu ra. Giá trị trung bình được tính bởi:

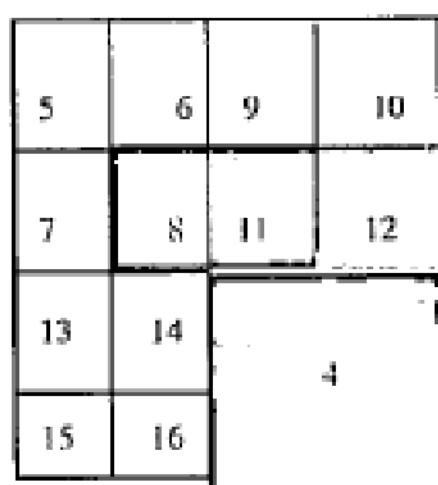
$$\text{Tổng giá trị mức xám} / \text{tổng số điểm trong vùng.}$$

Thuật toán này tạo nên một cây mà mỗi nút cha có 4 nút con ở mọi mức trừ mức ngoài cùng. Vì thế cây này có tên là cây tứ phân (quad tree). Cây này cho ta hình ảnh rõ nét về cấu trúc phân cấp của các vùng tương ứng với tiêu chuẩn.

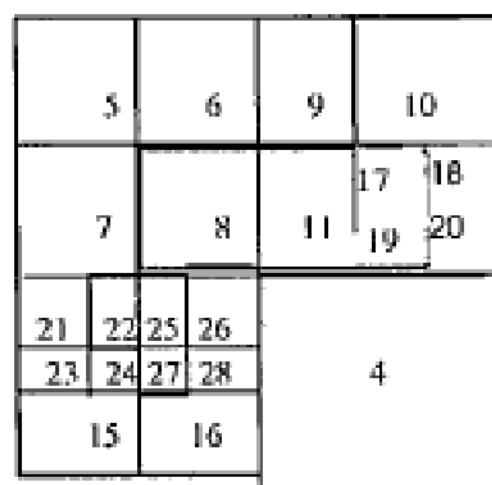


a) Ảnh gốc

b) Phân mức 1



a) Phân mức 2



b) Phân mức 3

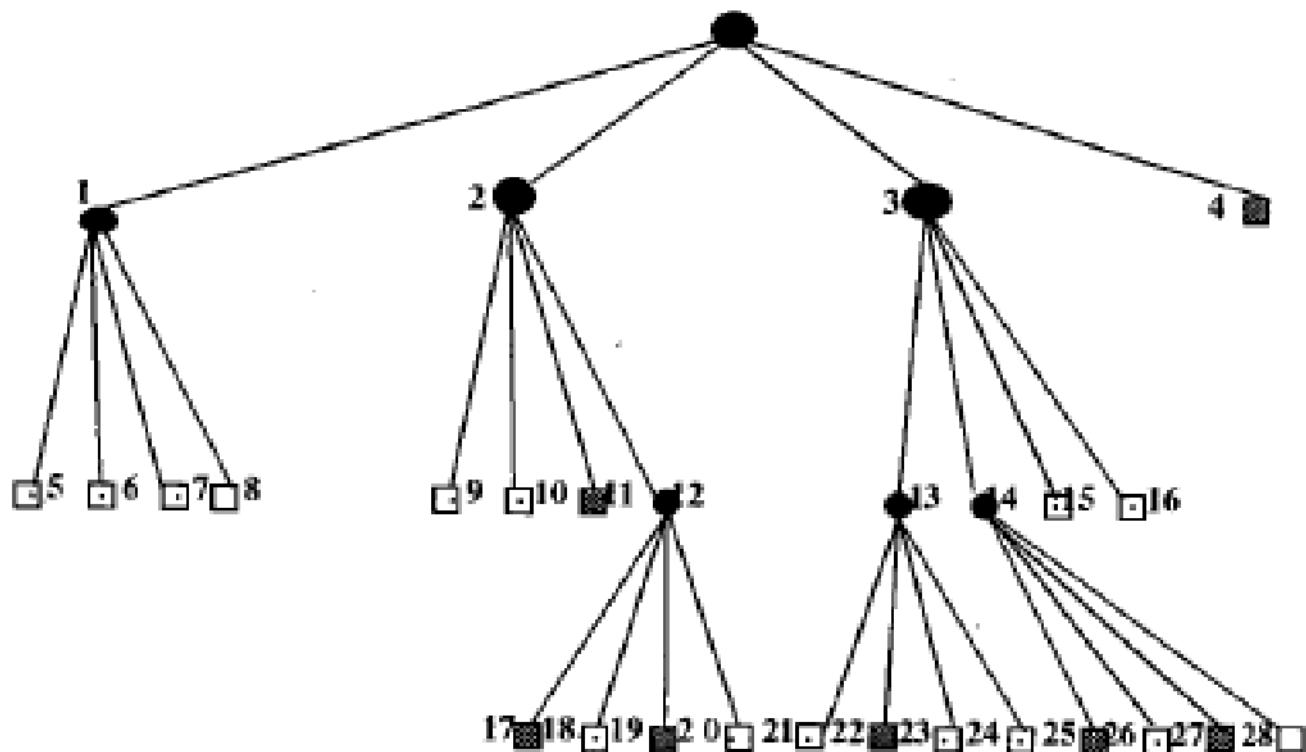
Hình 6.3. Phân vùng bởi tách.

Mỗi vùng thỏa tiêu chuẩn sẽ tạo nên một nút lá; nếu không nó sẽ tạo nên một nút trung và có 4 nút con tương ứng với việc chia làm 4 vùng. Ta cứ tiếp tục như vậy cho đến khi phân xong. Các nút lá của cây biểu diễn số vùng đã phân theo tiêu chuẩn (hình 6.3).

Tiêu chuẩn phân vùng ở đây là màu sắc. Nếu mọi điểm của vùng là màu trắng sẽ tạo nên một nút lá trắng và tương tự như vậy với nút lá đen. Nút màu ghi có nghĩa là vùng không thuần nhất và phải tiếp tục chia.

### 6.3.2. Phương pháp cục bộ hay phân vùng bởi hợp

Ý tưởng của phương pháp này là xem xét ảnh từ các miền nhỏ nhất rồi hợp chúng lại nếu thỏa tiêu chuẩn để được một miền đồng nhất lớn hơn. Ta lại tiếp tục với các miền thu



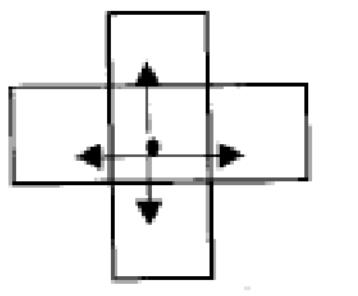
Hình 6.4. Cây tử phân tương ứng.

được cho đến khi không thể hợp được nữa. Số miền còn lại cho ta kết quả phân đoạn. Như vậy, miền nhỏ nhất của bước xuất phát là điểm ảnh.

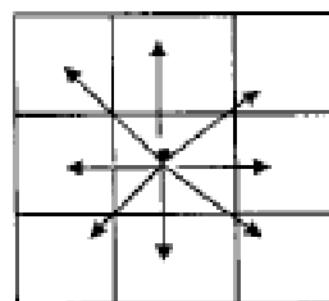
Phương pháp này hoàn toàn ngược với phương pháp tách. Song điều quan trọng ở đây là nguyên lý hợp 2 vùng. Việc hợp 2 vùng được thực hiện theo nguyên tắc sau:

- Hai vùng phải đáp ứng tiêu chuẩn, thí dụ như cùng màu hay cùng mức xám.
- Chúng phải kế cận nhau.

Ở đây cũng nên làm rõ khái niệm kế cận. Trong xử lý ảnh, người ta dùng khái niệm liên thông để xác định kế cận. Có hai khái niệm về liên thông mà chúng ta đã nói ở trên là 4 liên thông và 8 liên thông. Với 4 liên thông một điểm ảnh  $I(x,y)$  sẽ có 4 kế cận theo 2 hướng x và y; trong khi đó với 8 liên thông, điểm ảnh  $I(x,y)$  sẽ có 4 liên thông theo 2 hướng x và y và 4 liên thông khác theo hướng chéo  $45^\circ$ . Hình 6.5 dưới đây cho ta khái niệm về 4 và 8 liên thông.



a) 4-liên thông



b) 8-liên thông

Hình 6.5. Khái niệm 4 và 8 liên thông.

Dựa theo nguyên lý của phương pháp hợp, ta có 2 thuật toán:

- Thuật toán tô màu (Blob coloring);
- Thuật toán đẽ quy cục bộ.

#### 6.3.2.1. Thuật toán tô màu

Thuật toán này sử dụng khái niệm 4-liên thông. Người ta dùng một cửa sổ di chuyển trên ảnh để sánh với tiêu chuẩn hợp. Thuật toán có thể tóm tắt như sau:

For each point I(x,y) do

(\* Kiểm tra màu của các lân cận \*)

Begin

If Criteria(x,y) = Criteria(x-1,y) Then M(x,y) = M(x-1,y)

Else

If Criteria(x,y) = Criteria(x,y-1) then M(x,y) = M(x,y-1)

else M(x,y)= NewM;

End

(\* Hợp lại nếu các lân cận cùng màu \*)

if (Criteria(x,y) = Criteria(x-1,y)) and (Criteria(x,y) = Criteria(x,y-1))

then Merge 2 areas by giving the same color

End

Trong thuật toán trên, chỉ tiêu hợp là màu đồng nhất. Thực tế chúng tôi rằng thuật toán này tỏ ra khá hiệu quả khi dùng camera một hàng vì có thể phân đoạn theo từng hàng.

#### 6.3.2.2. Thuật toán đẽ quy cục bộ

Thuật toán này sử dụng phương pháp tìm kiếm trong một cây để làm tăng kích thước vùng. Trước tiên, người ta tìm kiếm các lân cận để tăng kích thước tối đa của vùng rồi sau đó mới quan tâm đến các vùng khác và cũng áp dụng thuật toán trên. Thuật toán này có sử

dùng một thủ tục để quy GiaTang để thực hiện việc tăng kích thước một vùng một cách dễ quy. Thuật toán được mô tả như sau:

*a) Thủ tục chính*

Procedure DQCB

Begin

For each point I(x,y) do

If I(x,y) < 0 Then Begin Save (I(x,y))

GiaTang(x,y)

NSeg <- NSeg + 1

End;

End

*b) Thủ tục GiaTang*

I(x,y) < 0 (\* đặt I(x,y) là không để không phải xét lại \*)

For  $\forall$  points kề cận của I(x,y) do

Begin

If (I(x,y) < 0) and (Criteria(Pixel) = Criteria\_Bd)

Then GiaTang(Pixel);

End;

Nhìn chung phương pháp này nhanh và dễ cài đặt. Việc viết chương trình cho thủ tục này coi như bài tập.

### 6.3.3. Phương pháp tổng hợp

Hai phương pháp vừa xét ở trên có một số nhược điểm. Phương pháp tách sẽ tạo nên một cấu trúc phân cấp và thiết lập mối quan hệ giữa các vùng. Tuy nhiên nó thực hiện việc chia quá chi tiết. Như trong thí dụ hình 6.3, vùng số 3 lẽ ra chỉ cần phân thành 2 vùng con, song thuật toán này lại chia đến 10 vùng con.

Phương pháp hợp cho phép làm giảm số miền liên thông xuống tối thiểu, nhưng cấu trúc hàng ngang dàn trải, không cho ta thấy rõ mối liên hệ giữa các miền.

Chính vì những nhược điểm này, người ta nghĩ đến phối hợp cả 2 phương pháp. Trước tiên, dùng phương pháp tách để tạo nên cây từ phân, phân đoạn theo hướng từ gốc đến lá. Tiếp theo, tiến hành duyệt cây theo chiều ngược lại và hợp các vùng có cùng tiêu chuẩn.

Với phương pháp này ta thu được một miêu tả cấu trúc của ảnh với các miền liên thông có kích thước tối đa.

Giải thuật tách hợp gồm một số bước chính sau:

1. Kiểm tra tiêu chuẩn đồng nhất.

a) Nếu không thỏa và số điểm trong một vùng nhiều hơn 1 điểm, tách làm 4 vùng (trên, dưới, phải và trái) bằng cách gọi đệ quy. Nếu kết quả tách xong và không tách được nữa chuyển sang bước 2.

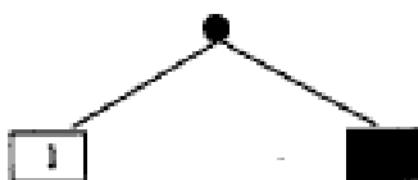
b) Nếu tiêu chuẩn đồng nhất là thỏa thì tiến hành hợp vùng và cập nhật lại giá trị trung bình vùng cho vùng.

2) Hợp vùng

Cần kiểm tra 4 lần cận như đã nêu trên. Có thể có nhiều vùng thỏa mãn. Khi đó ta chọn vùng tối ưu nhất rồi tiến hành hợp.

	1	
4	Hiện thời	3
	2	

Thí dụ với cây trong hình 6.4, áp dụng phương pháp tách hợp ta thu được một cây chỉ có 2 vùng hay nói cách khác là phân đoạn ảnh gốc thành 2 miền đồng nhất (hình 6.6).



Hình 6.6. Cây thu được theo phương pháp tách hợp.

#### 6.4. PHÂN VÙNG DỰA THEO ĐƯỜNG BIÊN

Như đã trình bày trong chương 5, biên là một trong những đặc trưng quan trọng của ảnh. Cũng vì thế mà trong nhiều ứng dụng, người ta sử dụng cách phân đoạn dựa theo biên. Việc phân đoạn ảnh dựa vào biên được tiến hành qua một số bước:

- Phát hiện biên và làm mờ biên.
- Làm mảnh biên.
- Nhị phân hóa đường biên.
- Mô tả biên.

Việc phát hiện biên đã mô tả trong chương trước. Trong phần dưới đây sẽ trình bày một số bước còn lại.

#### 6.4.1. Làm mảnh biên

Làm giảm hay mảnh biên thực chất là làm mờ biên với độ rộng chỉ 1 pixel. Như trong chương 5, kỹ thuật Laplace dùng trong phát hiện biên cho kết quả trực tiếp biên ảnh với độ rộng 1 pixel. Song với nhiều kỹ thuật khác thì không hoàn toàn như vậy.

Khi thực hiện đạo hàm một ảnh, ta thu được những điểm cực trị cục bộ. Theo kỹ thuật gradient, những điểm cực trị cục bộ có thể coi như biên. Do vậy cần tách biệt những điểm cực trị đó để xác định chính xác biên ảnh và để giảm độ rộng biên ảnh. Một phương pháp hay dùng trong kỹ thuật làm mảnh biên chữ là phương pháp "*Loại bỏ các điểm không cực đại*".

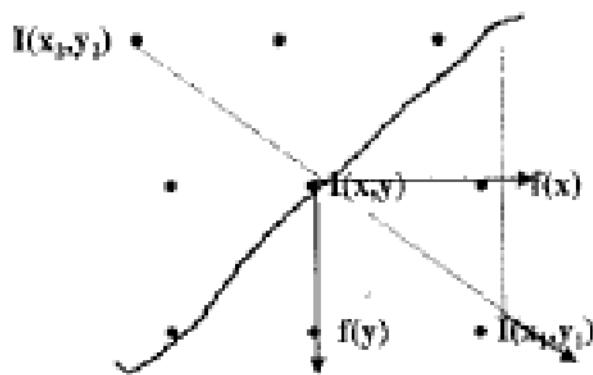
Giả sử ảnh  $I(x,y)$  gồm gradient hướng và gradient biên độ (còn gọi là là bản đồ biên độ và bản đồ hướng). Với mỗi điểm ảnh  $I(x,y)$ , ta xác định các điểm lân cận của nó theo hướng gradient. Gọi các điểm đó là  $I(x_1,y_1)$  và  $I(x_2,y_2)$ . Các điểm này được minh họa trên hình 6.7.

Nếu  $I(x,y)$  lớn hơn cả  $I(x_1,y_1)$  và  $I(x_2,y_2)$ , giá trị của  $I(x,y)$  sẽ được bảo toàn. Nếu không nó sẽ được đặt là 0 và coi như bị loại bỏ. Thuật toán này được sử dụng trong phương pháp Canny nói ở chương trước.

Ngoài thuật toán kể trên còn nhiều kỹ thuật làm mảnh biên để xác định khung của đối tượng (ảnh hay kí tự) như kỹ thuật mảnh biên chữ do Sherman đề xuất sau đó được Fraser cải tiến cho ảnh nhị phân. Kỹ thuật này có thể mô tả tóm tắt như sau:

Tại mỗi vị trí cùa số, phần tử trung tâm sẽ được xoá (đổi thành trắng) nếu nó thoả mãn một trong hai điều kiện sau:

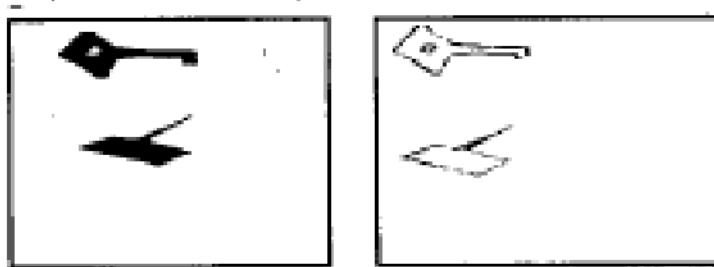
- Nó là điểm đen duy nhất kết nối với 2 điểm đen không kế nhau.
- Nó là điểm đen có duy nhất một lân cận cũng là điểm đen ngoại trừ không tồn tại một chuyển đổi nào tại phần tử trước nó.



Hình 6.7. Lần cận gradient của ảnh.

Cụ thể là điểm đó phải đồng thời thoả các điều kiện sau:

$$\left\{ \begin{array}{l} 2 \leq NZ(P_1) \leq 6 \\ \text{và } ZO(P_1) = 1 \\ \text{và } P_2^* P_4^* P_8 = 0 \text{ hay } ZO(P_2) \neq 1 \\ \text{và } P_2^* P_4^* P_8 = 0 \text{ hay } ZO(P_4) \neq 1 \end{array} \right.$$



a) Ảnh gốc

b) Ảnh sau khi làm mờ biến

Hình 6.8. Ảnh làm mờ bởi loại bỏ điểm cực đại.

với : -  $ZO(P_1)$  là số phần tử 0 chuyển sang phần tử khác 0 trong tập có thứ tự  $P_2, P_4, P_6, P_8$ .  
 -  $NZ(P_1)$  số hàng xóm (lần cận) khác 0 của  $P_1$ .

Hình 6.9 dưới đây mô tả một số tính huống áp dụng và không áp dụng được điều kiện này và thí dụ về ảnh sau khi đã làm mờ.

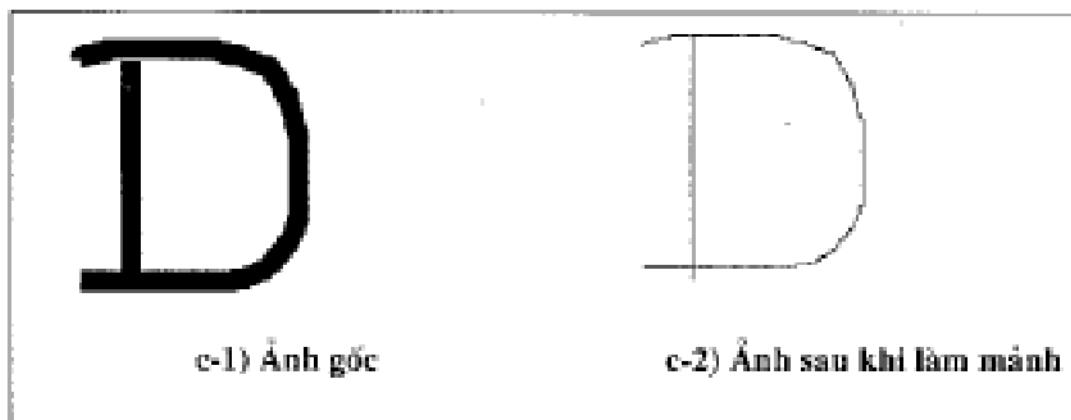
$P_3$	$P_2$	$P_1$
$P_4$	$P_5$	$P_6$
$P_5$	$P_6$	$P_7$

1	1	0
0	$P_1$	0
0	0	0

1	1	0
0	$P_1$	1
0	0	0

1	0	0
1	$P_1$	0
0	0	0

- a) Điểm ảnh  $P_1$  và 8 lân cận  
 b-1) xoá  $P_1$  sẽ tạo thành 2 miền  
 b-2) xoá  $P_1$  sẽ gây ngắn đoạn  
 b-3)  $2 < NZ(P1) < 6$



Hình 6.9. Làm mờ ảnh.

#### 6.4.2. Nhị phân hóa đường biên

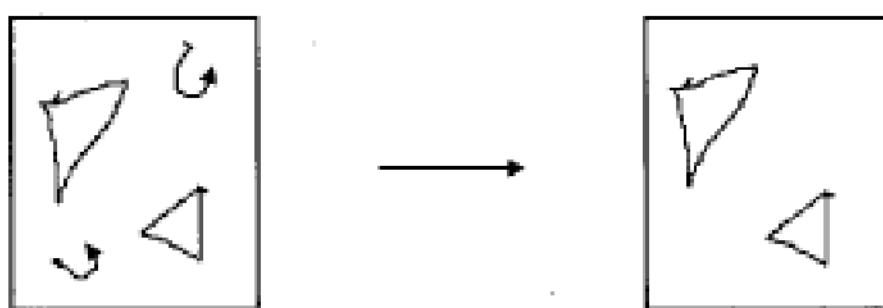
Nhị phân hóa đường biên là giai đoạn then chốt trong quá trình trích chọn vì nó xác định đường bao nào thực sự cần và đường bao nào có thể loại bỏ. Nói chung, người ta thường nhị phân hóa đường biên theo cách thức làm giảm nhiễu hoặc tránh hiện tượng kéo sợi trên ảnh. Điều này cũng giải thích tại sao phân đoạn dựa theo biên có hiệu quả khi ảnh có độ tương phản tốt. Trong trường hợp ngược lại, có thể sẽ bị mất một phần đường bao hay đường bao có chấn, không khép kín, v.v., do đó sẽ bất lợi cho biểu diễn sau này. Một phương pháp hay được dùng là chọn ngưỡng thích nghi. Với cách chọn này, ngưỡng sẽ phụ thuộc vào hướng của gradient nhằm làm giảm sự xoắn của biên độ. Lúc đầu người ta định ra một ngưỡng nào đó và sau đó sử dụng hệ số sinh thích nghi thông qua lời giải toán tử đạo hàm theo hướng tìm được để tinh chỉnh.

#### 6.4.3. Miêu tả đường biên

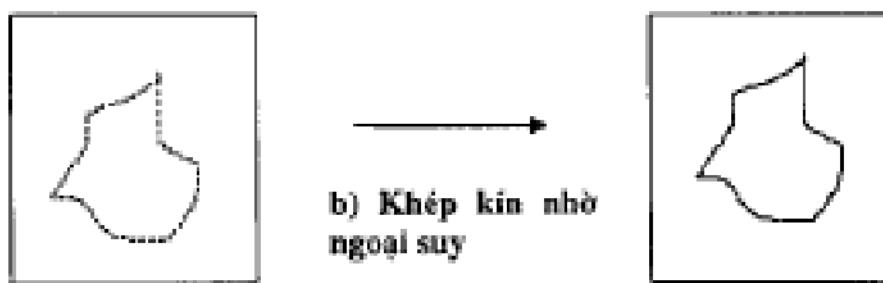
Khi đã có bản đồ biên của ảnh, ta cần phải biểu diễn nó dưới dạng thích hợp phục vụ cho việc phân tích và làm giảm lượng thông tin dùng để miêu tả đối tượng. Nguyên tắc chủ yếu là tách biệt mỗi biên và gán cho nó một mã. Đầu từ thời điểm này, chúng ta không quan tâm tới cách biểu diễn quen thuộc của ảnh bởi ma trận số mà ta quan tâm đến cách

miêu tả bởi một cấu trúc thích hợp và có động.

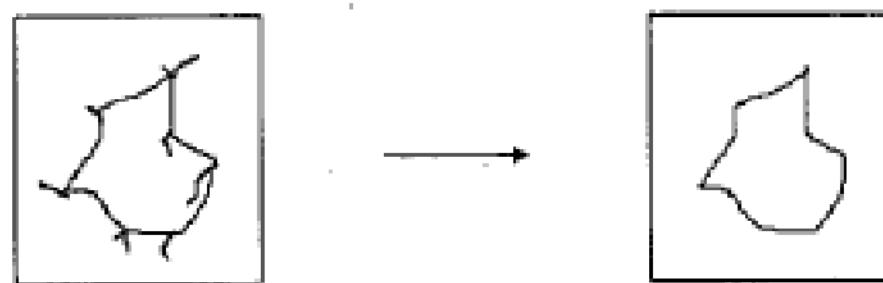
Quá trình miêu tả biến đổi tượng là khá rộng vì có nhiều phương án khác nhau và mỗi phương án có liên quan mật thiết với các đặc thù của từng ứng dụng. Việc tách biệt đường bao có thể được tăng cường thêm các điều kiện nhằm loại bỏ các đường bao không khép kín hoặc khép kín nhờ phương pháp ngoại suy, hay bỏ đi các chân rết bám theo các đường bao kín.



a) Loại bỏ đường bao hở



b) Khép kín nhờ ngoại suy



c) Loại bỏ các chân rết

Hình 6.10. Làm rõ biến.

Việc mã hoá đường bao có thể thực hiện theo nhiều cách khác nhau. Có thể dùng biểu diễn chính xác đường bao hay xấp xỉ nhờ nội suy. Thông thường các cấu trúc cơ sở mã hoá đường bao gồm 4 loại: điểm, đoạn thẳng, cung và đường cong.

Tuy nhiên, cũng cần lưu ý là luôn có xung đột giữa độ phức tạp tính toán và khả năng biểu diễn ảnh của cách mã hoá thông tin. Biểu diễn đường bao bởi các điểm nói chung không phức tạp song lại rất nghèo nàn về cấu trúc và không có động (vì có bao nhiêu điểm tạo nên biên ta phải mô tả bấy nhiêu). Trong khi đó, biểu diễn bằng đường cong đa thức bậc cao làm tăng độ phức tạp tính toán, song bù lại cấu trúc dữ liệu lại rất có động. Một số phương pháp mã hoá đường bao hay dùng sẽ được trình bày dưới đây.

#### *6.4.3.1 Mã hoá theo tọa độ Décác*

Kiểu mã hoá này khá đơn giản. Đường bao của ảnh được biểu diễn bởi một danh sách các điểm ảnh tạo nên đường bao. Gọi C là đường bao của ảnh và C(i,j) là các điểm. Rõ ràng cách mã hoá này bộc lộ nhược điểm là không giấu được lượng thông tin. Tuy nhiên, việc tính toán lại khá nhanh và có thể cung cấp những phương tiện phục vụ cho việc trích chọn các đặc trưng hình học của ảnh.

Việc mã hoá được thực hiện theo phương pháp "*Đi theo đường bao*" (Contour Following), sử dụng kỹ thuật tìm kiếm thông tin theo chiều sâu trên cây. Nếu áp dụng một cách đơn thuần kỹ thuật này, có thể ta sẽ thu được đường bao thô theo nghĩa có điểm có thể xuất hiện 2 lần. Người ta thường phối hợp với việc kiểm tra 8 liên thông để làm mịn biên.

Thuật toán Contour Following có xem xét 8 liên thông có thể mô tả như sau:

For each point I(x,y) do

Begin

if I(x,y) ∈ C then

    Root <- I(x,y)

    KQ <- CountFoll(root,0)

    if KQ then Dem <- Dem+1

Function ContFoll(Pic, Depth)

Begin

If

End

If (Depth <> 0) and (pic = root then Control = True

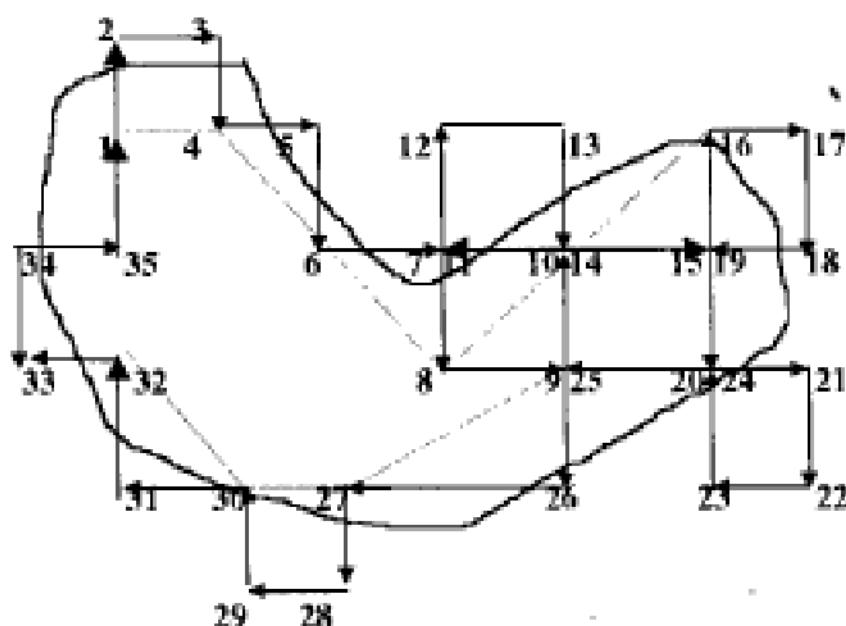
else if 8 pixels = 0 then Control = False (\* .vì đây là đường chân rất cần loại \*)

else Begin . push Stack all the pixels <>0

```

    . Reset all these pixel
    . Repeat
        For each pixel in stack do
            KQ <- ContFolk(i, depth+1)
        Until KQ = OK or Stack empty
    End
End

```

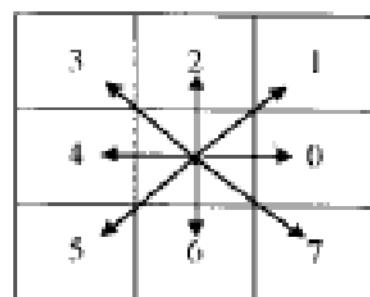


Hình 6.11. Thuật toán "đi theo đường bao".

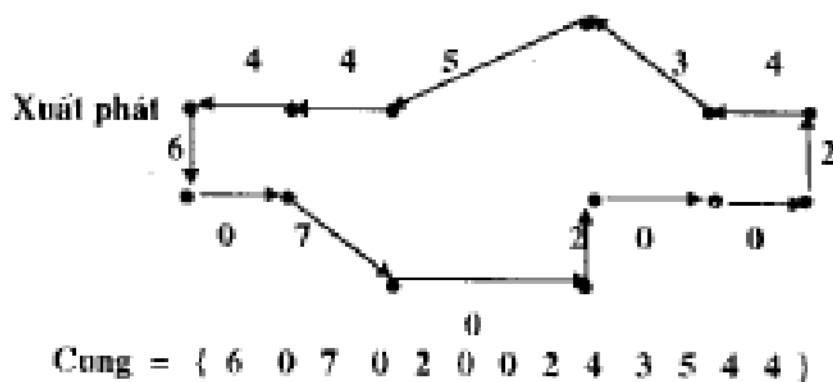
Hình 6.11 trên cho ta danh sách các điểm ảnh trên đường bao được duyệt theo thứ tự từ 1 đến 35.

#### 6.4.3.2. Mã hoá Freeman

Phương pháp mã hoá này cũng là một cách biểu diễn chính xác các điểm đường bao bằng việc sử dụng vị trí tương đối của điểm trên đường bao với điểm trước. Nguyên tắc của việc mã hoá này thể hiện trên hình 6-12a. Người ta dùng một mặt nạ để xác định mã của mỗi điểm trong 8 liên thông so với điểm ở tâm. Từ một điểm đã cho trên đường bao, người ta mã hoá đường bao bằng cách đi theo nó (hình 6-12b). Tuy nhiên, người ta thích mã hoá theo góc giữa các cung (6-12c). Trong trường hợp này, mã góc sẽ được tính bởi:  $goc_i = \text{MOD}_{16}[\text{arc}_i - \text{arc}_{i-1} + 11] - 3]$ . Hình 6-12d là mã của 6-12b theo góc và gọi là mã đường bao.



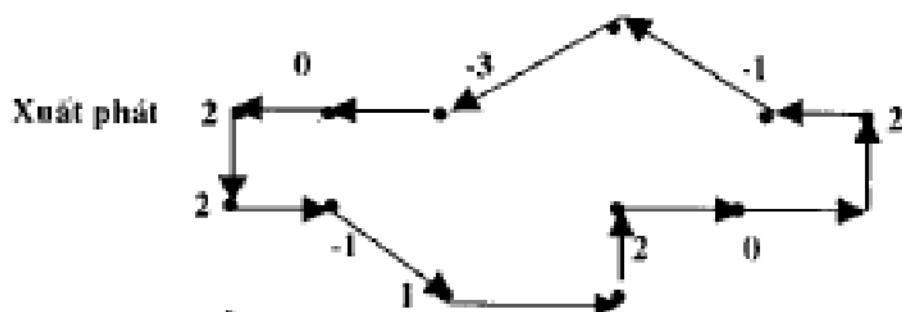
a) 8 liên thông và mã hướng tương ứng.



b) Đồ thị biểu diễn đường bao có trọng số.



c) Mã hóa góc.



Góc = { 2 2 -1 -1 2 -2 0 2 2 -1 2 -1 0 }

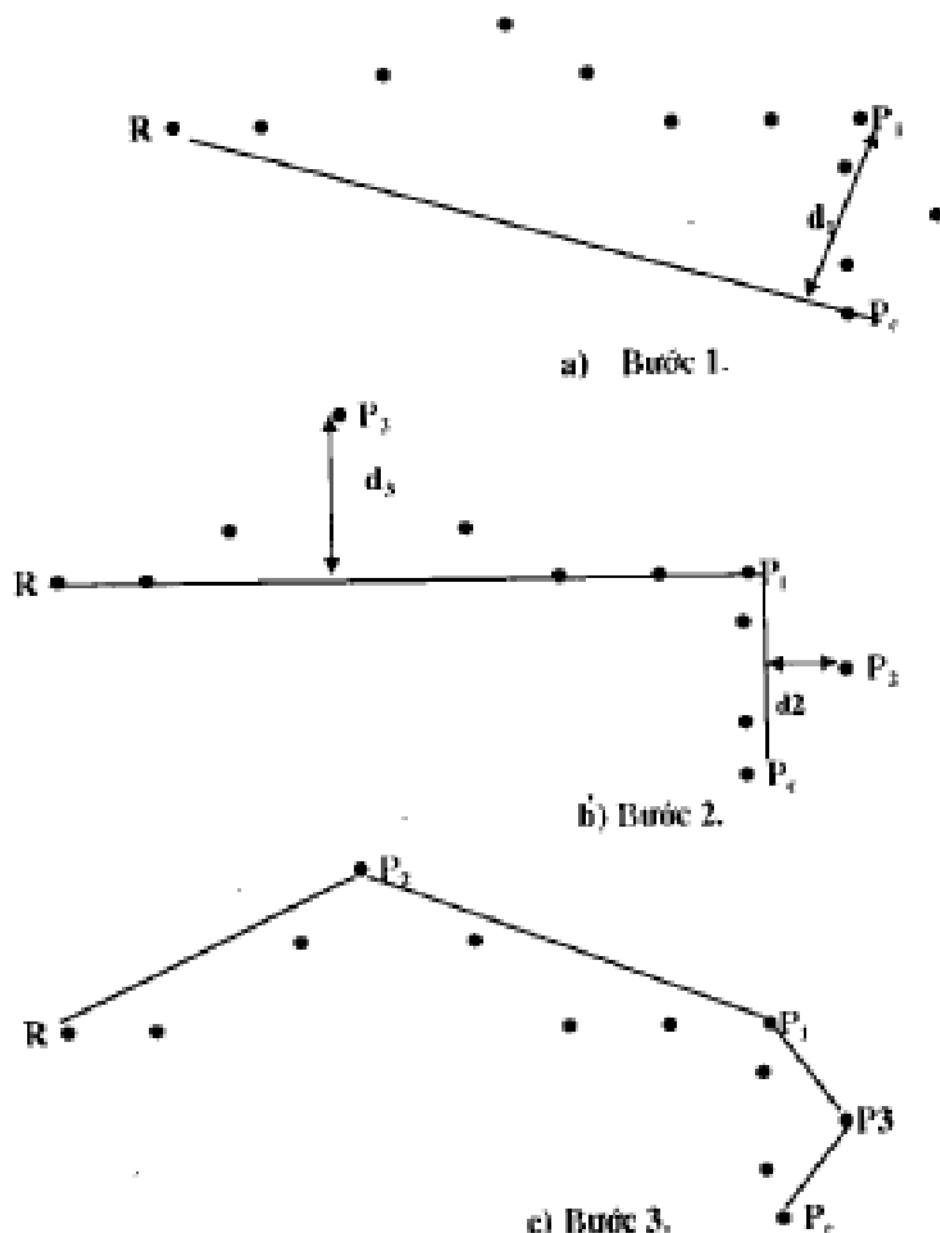
d) Mã hóa theo góc.

Hình 6.12. Mã hóa Freeman.

#### 6.4.3.3. Xấp xỉ bởi đoạn thẳng

Ngược với hai cách mã hóa ở trên, kỹ thuật mã hóa bởi đoạn thẳng không cho phép khôi phục tất cả các thông tin chưa dựng trong đường bao nhưng lại có thể xấp xỉ nó bởi đoạn thẳng với độ chính xác phụ thuộc vào người dùng. Nếu độ chính xác càng thấp, thông tin mô tả càng cồng kềnh.

Có thể mô tả tóm tắt như sau: tiến hành lấy mẫu đường bao theo nhiều giao đoạn với bước nhảy là  $P$ . Nối điểm xuất phát  $R$  với điểm đang xét  $P_i$  bởi một đoạn thẳng. Sau đó tính toạ độ của  $P_j$ , một điểm nằm giữa  $R$  và  $P_i$  sao cho khoảng cách từ  $P_j$  đến đoạn thẳng là cực đại. Gọi khoảng cách này là  $d_s$ . Nếu  $d_s$  lớn hơn một ngưỡng cho trước (độ chính xác của xấp xỉ) người ta phân đoạn  $RP_i$  thành hai đoạn  $RP_j$  và  $P_jP_i$ , và tiếp tục thực hiện lấy mẫu với từng đoạn cho tới khi đoạn thẳng tìm được là "rõ giàn" với đường bao.



Hình 6.13. Xấp xỉ đường bao bởi đường gấp khúc.

Cùng trong phương pháp xếp xỉ bởi đoạn thẳng, có một cách tiếp cận hoàn toàn với khác với phương pháp trên, đó là phép biến đổi Hough. Cách tiếp cận đó như sau:

Giả sử  $(x', y')$  là một điểm trong ảnh. Mọi điểm qua  $(x', y')$  phải thoả phương trình:

$$y' = mx' + c \quad \text{với } m, c \text{ là các tham số} \quad (6.9)$$

Viết lại phương trình trên ta có:

$$c = -mx' + y'$$

Như vậy, mọi đường thẳng qua  $(x', y')$  tương ứng với một điểm trong không gian tham  $(c, m)$ . Xét hai điểm  $(x_1, y_1)$  và  $(x_2, y_2)$  nằm trên  $m$  cùng một đường thẳng:

Với mỗi điểm ảnh, mọi đường thẳng qua  $m$  được biểu diễn bởi một điểm trong  $(c, m)$ . Thí dụ:

$$P(x_1, y_1): c = -mx_1 + y_1$$

$$P(x_2, y_2): c = -mx_2 + y_2$$

Nhưng một đường thẳng duy nhất trong không gian  $(x, y)$  qua 2 điểm  $(x_1, y_1)$  và  $(x_2, y_2)$  được biểu diễn bởi giao của 2 đường trong không gian tham số  $(c, m)$ . Điểm giao cho giá trị của  $c$  và  $m$  trong phương trình  $y = mx + c$  (hình 6.14a, b).

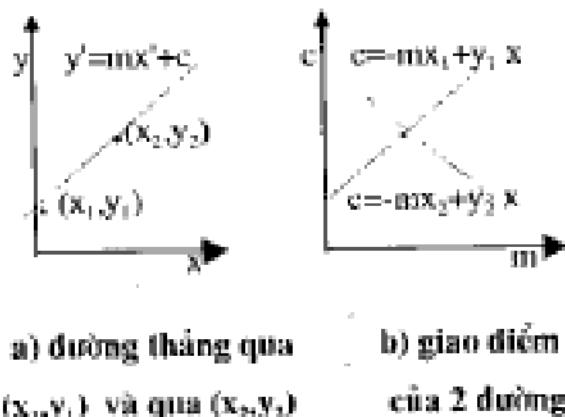
Để áp dụng kỹ thuật này, không gian tham số  $(c, m)$  cần phải được lượng hoá và như vậy ta cần dùng một ma trận tham số  $P(c, m)$  với:

$$c_1 \leq c \leq c_K \text{ và } m_1 \leq m \leq m_L; K \text{ là số điểm chia của } C \text{ và } N \text{ là số điểm ảnh.}$$

Cách tiến hành như sau:

1. Khởi tạo bảng tham số  $P(c, m)$ : các phần tử của bảng này được gán là 0.
2. Với mỗi điểm  $(x_i, y_i) \in$  biên ảnh,  $P(c, m) = P(c, m) + 1$  cho các điểm thoả phương trình 6.9.
3. Lặp lại quá trình trên cho đến khi toàn bộ ảnh được quét.

Kết thúc các bước này, mỗi phần tử của ma trận  $P(c, m)$  chứa số điểm biên thoả 6.9. Nếu số điểm này vượt quá một ngưỡng  $T$  nào đấy thì một đường thẳng dạng  $y' = mx' + c$  được khởi tạo. Cần chú rằng, trong biến đổi Hough, các điểm nằm trên cùng một đường thẳng không nhất thiết là liên tục. Đây là một tính chất quan trọng.

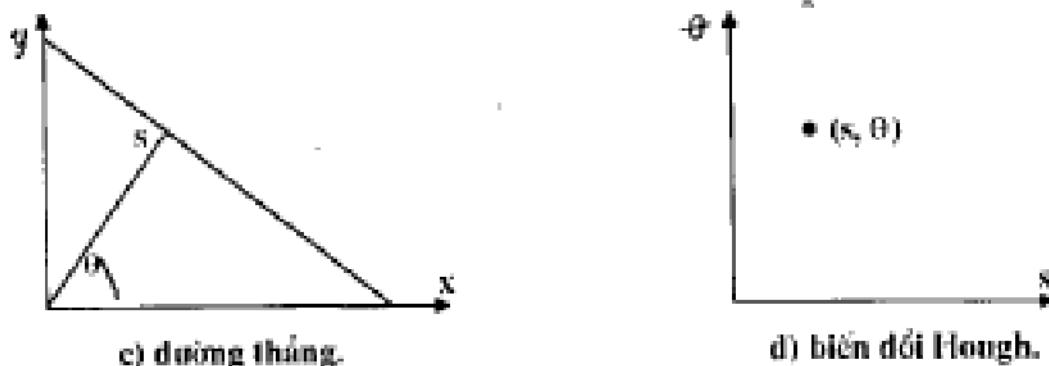


Hình 6.14.

Tuy nhiên, cần xem xét phương pháp này có áp dụng trong thực tế được không? Điều đó có nghĩa là cần tính độ phức tạp tính toán của phương pháp. Để dễ dàng thấy rằng độ phức tạp tính toán là  $O(KL)$  với  $K$  và  $L$  có nghĩa như trên. Độ phức tạp này là chấp nhận được. Song cũng dễ thấy, nếu biểu diễn bởi đường thẳng thì khi biểu diễn các đường đứng (vertical straight line) thì c có xu hướng tiến ra vô cùng. Một cách khắc phục là dùng hệ toạ độ cực  $(s, \theta)$ .

Trong biến đổi Hough, một đường thẳng trong một mặt phẳng với khoảng cách  $s$  và hướng  $\theta$  có thể biểu diễn bởi:  $s = x \cos\theta + y \sin\theta$  (hình 6.14 c và d).

Đường thẳng này có thể coi như một điểm trong mặt phẳng  $(s, \theta)$  như trong hình 6.14d. Kỹ thuật này sử dụng tất cả các điểm của đường bao và biến đổi gradient hướng. Biến đổi Hough tạo nên một ảnh xạ các điểm của đường thẳng với một điểm trong mặt phẳng.



c) đường thẳng.

d) biến đổi Hough.

Hình 6.14c,d. Biến đổi Hough trong hệ toạ độ cực.

Giả sử các điểm của đường bao là  $(x_i, y_i)$ ,  $i=1, 2, \dots, N$ . Với một số giá trị đã chọn của  $s$  và  $\theta$ , ánh xạ cặp  $(x_i, y_i)$  thành điểm  $(s, \theta)$  có nghĩa là với 1 điểm ta đi tìm phương trình đường thẳng đi qua điểm này và có hướng vuông góc với gradient hướng. Ta tính được  $C(s, \theta)$ . Ta thu được tập biểu diễn bởi quan hệ sau:

$$C(s_i, \theta_i) = C(s_i, \theta_i) + 1 \text{ nếu } x_i \cos\theta + y_i \sin\theta = s_i \text{ với } \theta = \theta_i; \quad (6.10)$$

Cực trị cực bộ  $C(s, \theta)$  cho các đường thẳng khác nhau đi qua điểm biến này. Lưu ý rằng việc tìm kiếm trong không gian 2 chiều có thể biến đổi sang không gian một chiều nếu gradient hướng  $\theta$ , ở mỗi điểm biến là đã biết.

Miền biến thiên của các tham số:

$$-\sqrt{N_1^2 + N_2^2} \leq s \leq \sqrt{M_1^2 + M_2^2}$$

$$-\pi/2 \leq \theta \leq \pi/2$$

#### 6.4.3.4. Xấp xỉ đa thức

Nếu ta dùng phương pháp xấp xỉ đường cong (biên thường cong) bằng đường thẳng như trên, sai số sẽ lớn. Để giảm sai số người ta nghĩ đến dùng xấp xỉ bởi đường cong đa thức. Cũng giống như xấp xỉ đường thẳng, ta cũng phải lấy mẫu đường bao để tính các hệ số của đa thức. Các đa thức thường được dùng là loại spline.

Giả sử đường bao được biểu diễn theo tham số t. Các tọa độ tương ứng của một điểm trên đường bao là  $x(t), y(t)$ . Việc biểu diễn bởi đường cong đa thức có thể viết:

$$x(t) = \sum_{j=0}^n p_j B_{j,k}(t) \quad (6.11)$$

với  $p_i$  là các điểm kiểm tra và  $B_{j,k}$ ,  $i=1, 2, \dots, n$ ;  $k=1, 2, \dots$  là các đường cong chuẩn bậc k. Các đa thức  $B_{j,k}$  được tính 1 cách hồi quy như sau:

$$\begin{cases} B_{j,k}(t) = \frac{(1-t_i)B_{j,k-1}(t) + (t_{i+k-1}-t_i)B_{j+1,k-1}(t)}{t_{i+k-1}-t_i} & \text{với } k=2, 3, \dots \\ \text{và } B_{j,1}(t) = 1 \text{ nếu } t_i \leq t \leq t_{i+1} \text{ và } B_{j,1}(t) = 0 \text{ nếu không.} \end{cases} \quad (6.12)$$

Các  $t_i$ ,  $i=0, 1, \dots$  gọi là các mẫu. Các định liên kết với các mẫu kí hiệu là  $s$ , và được định nghĩa như trung bình cục bộ của chuỗi k-1 mẫu:

$$s_i = (t_{i+1} + t_{i+2} + \dots + t_{i+k})/(k-1) \quad k > 2 \text{ và } 0 \leq i \leq n \quad (6.13)$$

Các  $t_i, s_i$  là các giá trị đặc biệt của tham số định. Thị dụ với  $k=3$  ta có đường bậc 2,  $k=4$  ta có dạng lặp phương.

Nếu việc phân đoạn là đều, có nghĩa là  $t_{i+1} - t_i = \Delta t$  với mọi i thì  $B_{j,k}(t)$  là đường spline đều và ta có

$$B_{j,k}(t) = B_{j,1}(t-i), i = k-1, \dots, n-k+1 \quad (6.14)$$

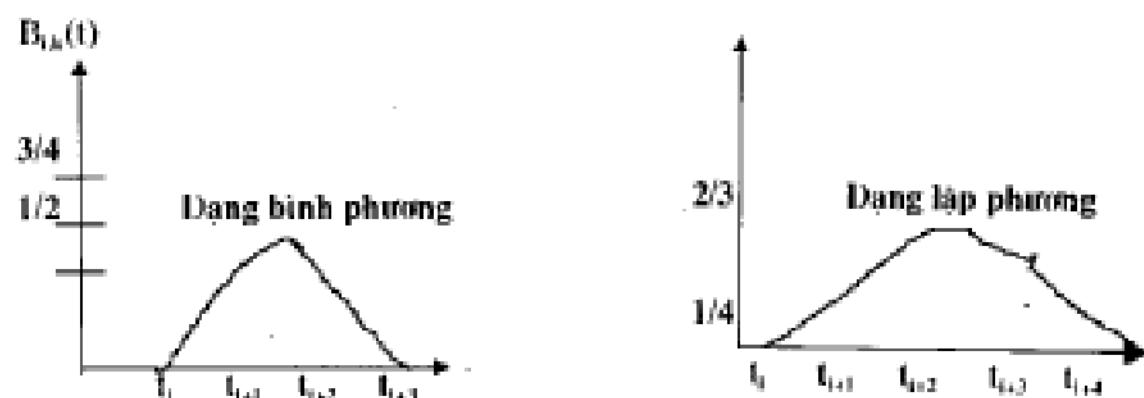
Nếu  $\Delta t = 1$  và đường spline là mở và đều, các  $t_i$  sẽ được chọn bởi:

$$t_i = \begin{cases} 0 & i < k \\ i-k+1 & k \leq i \leq n \\ n-k+2 & i > n \end{cases} \quad (6.15)$$

Nếu đường spline có chu kỳ đều, các  $t_i$  sẽ được chọn bởi:

$$t_i = i \bmod(n+1) \quad (6.16)$$

$$\text{và } B_{j,k}(t) = B_{j,1}(t-i) \bmod (n+1) \quad (6.17)$$



Hình 6.15. Một số dạng đường spline.

#### 6.4.4. Nối lỏng thuật toán phân vùng (Relaxation in Algorithms in region analysis)

Các phương pháp trên gọi là phương pháp túi định (determinist) theo nghĩa mỗi điểm ảnh chỉ thuộc một vùng. Tuy nhiên, thực tế có điểm nằm trên đường giúp分辨 và như vậy nó thuộc nhiều vùng. Sẽ có ích nếu ta xây dựng một véc-tơ  $P_k$  cho mỗi điểm  $x_k$ . Véc-tơ này chứa xác suất  $p_k(i)$  để thuộc  $R_i$ ,  $i=1, 2, \dots, N$ :

$$P_k = [p_k(1), p_k(2), \dots, p_k(N)] \quad (6.18)$$

Véc-tơ này có thể được sử dụng để tạo nên sự phân vùng túi định: điểm ảnh  $x_k$  sẽ được gán cho vùng  $R_i$  có xác suất  $p_k(i)$  lớn nhất. Xác suất  $p_k(i)$  cũng được gọi là trọng số tin cậy (confidence weight) và phải thoả mãn:

$$0 \leq p_k(i) \leq 1 \quad (6.19)$$

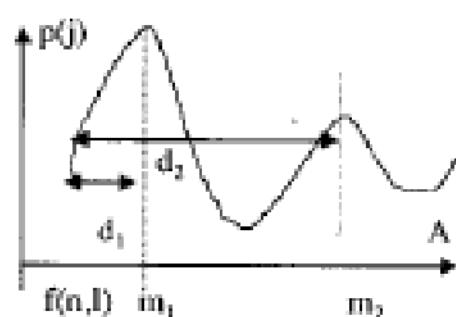
$$\sum_{i=1}^N p_k(i) = 1 \quad (6.20)$$

Có nhiều cách để xây dựng ước lượng ban đầu cho véc-tơ  $P_k$  này. Lược đồ xám là một công cụ thích hợp. Giả sử rằng  $m_i$ ,  $i=1, 2, \dots, N$  là cường độ xám trung bình của mỗi vùng tương ứng với đỉnh của lược đồ xám và  $f(x_k) = f(n, l)$ , cường độ xám của điểm tại  $x_k = (n, l)$ . Ước lượng ban đầu của các trọng số tin cậy  $p_k^{(0)}(i)$  với  $i=1, 2, \dots, N$  được tính như sau:

$$p_k^{(0)}(i) = \frac{1/|f(n, l) - m_i|}{\sum_{i=1}^N 1/|f(n, l) - m_i|} \quad i=1, \dots, N \quad (6.21)$$

Xét  $d_i = |f(n,i) - m_i|$  là khoảng cách từ điểm có cường độ sáng  $f(n,i)$  đến miền  $R_i$ , có trung bình  $m_i$ ,  $i=1, \dots, N$ .

Trên thực tế, rất có thể có 2 điểm ảnh kề cận thuộc về 2 miền khác nhau  $R_i, R_j$ .



Các miền như vậy gọi là miền tương hợp. Các miền không tương hợp là các miền không hy vọng tìm thấy các vị trí ảnh lân cận.

Để miêu tả sự tương hợp giữa 2 miền  $R_i$  và  $R_j$ , người ta đưa ra các khái niệm hàm tương hợp:  $r(i,j)$  với  $-1 \leq r(i,j) \leq 1$  và được định nghĩa:

$$r(i,j) : \begin{cases} < 0 & R_i, R_j \text{ tương hợp} \\ = 0 & R_i, R_j \text{ độc lập} \\ > 0 & R_i, R_j \text{ không tương hợp} \end{cases} \quad (6.22)$$

Các hàm tương hợp có thể xác định trước hoặc được ước lượng nhờ phân vùng ban đầu. Khi đã có hàm tương hợp, ta có thể sử dụng để cập nhật lại véc-tơ tin cậy  $p_k$ .

Thường các miền không tương hợp có xu hướng cạnh tranh các điểm lân cận, trong khi đó các miền tương hợp lại có xu hướng hợp lại. Sự cạnh tranh và hợp nhất diễn ra liên tục cho đến khi đạt được một trạng thái ổn định. Mỗi điểm  $x_k$  nhận được một độ tin cậy thêm từ các điểm  $x_i$  trong lân cận và trọng số  $p_i^{(n)}(j)$  của điểm  $x_i$  tại bước  $n$  được tính:

$$\Delta P_k^{(n)}(i) = \sum_i d_{ik} \sum_{j=1}^N r_{ij}(i,j) P_i^{(n)}(j) \quad (6.23)$$

Việc lấy tổng được thực hiện trên tất cả các trọng số tin cậy  $p(j)$  của các điểm  $x_i$  nằm trên lân cận với  $x_k$ .

Xác suất  $P_k^{(n+1)}(i)$  được tính:

$$P_k^{(n+1)}(i) = \frac{P_k^{(n)}(i)(1 + \Delta P_k^{(n)}(i))}{\sum_{i=1}^N P_i^{(n)}(i)(1 + \Delta P_i^{(n)}(i))} \quad (6.24)$$

Quá trình lặp sẽ dừng khi tiêu chuẩn hội tụ đạt được. Khi thuật toán kết thúc sẽ tạo nên một vùng đồng nhất liên thông khá lớn nhờ loại bỏ được các vùng nhiễu giả tạo bên trong. Cách phân vùng tái định có thể dễ dàng thực hiện bằng cách chọn  $p_k(i)$  lớn nhất:  $x_k$

thuộc vùng R<sub>i</sub>. Mức độ hiệu quả của kỹ thuật này phụ thuộc vào việc chọn hàm tương hợp. S.Peleg và A.Rosenthal [PER78] đã nêu ra một cách chọn sau:

$$r_h(i,j) = \ln \frac{N^2 \sum_{k=1}^{N^2} P_k^{(i,j)}(l) P_l^{(i,j)}(j)}{\sum_{k=1}^{N^2} P_k^{(i,j)}(i) \sum_{l=1}^{N^2} P_l^{(i,j)}(j)} \quad (6.25)$$

với N<sup>2</sup> là số điểm ảnh của ảnh gốc và P<sub>k</sub><sup>(i,j)</sup>(l) là xác suất khởi tạo ban đầu. Cần chú ý rằng giá trị r<sub>h</sub>(i,j) phải thuộc vào khoảng đóng [-1,1].

## 6.5. PHÂN ĐOẠN DỰA THEO KẾT CẤU BẾ MẶT (TEXTURE)

Như đã giới thiệu trong phần 6.3, kết cấu quan sát thấy trong các mẫu trên bế mặt các đối tượng như gỗ, hạt, cát, quần áo, v.v. Kết cấu là thuật ngữ phản ánh sự lặp lại của các phần tử sợi (texel) cơ bản. Sự lặp lại này có thể ngẫu nhiên hay có tính chu kỳ hoặc gần như có chu kỳ. Một texel chứa rất nhiều điểm ảnh. Trong phân tích ảnh, kết cấu được phân làm 2 loại chính: loại thống kê và loại cấu trúc.

### 6.5.1. Tiếp cận thống kê

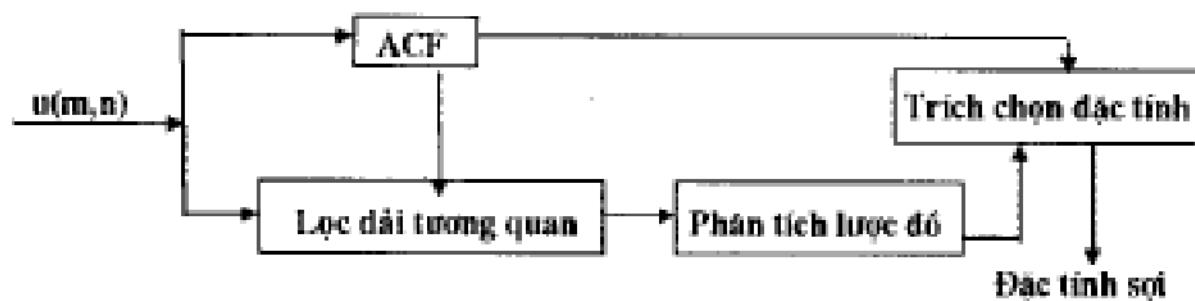
Tính kết cấu ngẫu nhiên rất phù hợp với các đặc trưng thống kê. Vì vậy, người ta có thể dùng các đặc trưng ngẫu nhiên để đo nó như: Hàm tự tương quan (AutoCorrelation Function-ACF), các biến đổi mặt dờ gờ, ma trận tương tranh, v.v.

Theo cách tiếp cận bằng hàm tự tương quan, độ thô của kết cấu sợi tỉ lệ với độ rộng của ACF, được biểu diễn bởi khoảng cách x<sub>0</sub>, y<sub>0</sub> sao cho r(x<sub>0</sub>, 0) = r(0, y<sub>0</sub>) = 1. Người ta cũng dùng cách đo nhánh của ACF nhờ hàm khối sinh moment:

$$M(k,l) = \sum_m \sum_n (m-\mu_1)^k (n-\mu_2)^l r(m,n) \quad (6.26)$$

với  $\mu_1 = \sum_m \sum_n m r(m,n)$  và  $\mu_2 = \sum_m \sum_n n r(m,n)$

Các đặc trưng của kết cấu sợi như độ thô, độ mịn hay hướng có thể ước lượng nhờ các biến đổi ảnh bằng kỹ thuật lọc tuyến tính (đã nêu trong chương 4). Một mô hình đơn giản trong trường ngẫu nhiên cho việc phân tích tính: kết cấu được mô tả như trong hình 6.16.



Hình 6.16. Phân tích kết cấu sợi bằng dài tương quan.

Trong mô hình này, trường kết cấu sợi trước tiên được giải chấp bởi bộ lọc lấy từ đầu ra của ACF. Như vậy, nếu  $u(m,n)$  là ACF thì:

$$u(m,n) \otimes u(m,n) = e(m,n) \quad (6.27)$$

Là một trường ngẫu nhiên không tương quan.

Lưu ý rằng bộ lọc là không duy nhất, có thể là nhân quả, bán nhân quả hay không nhân quả. Các ACF hay dùng như  $M(0,2)$ ,  $M(2,0)$ ,  $M(1,1)$ ,  $M(2,2)$ . Các đặc trưng của lược đồ bậc một của  $e(m,n)$  chẳng hạn như trung bình  $m_1$ , độ phân tán  $\sqrt{\mu_2}$  cũng hay được sử dụng.

Ngoài các đặc trưng trên, có thể đưa thêm một số khái niệm và định nghĩa các đại lượng dựa trên đó như: lược đồ hiệu mức xám(Histogram grey level difference), ma trận xuất hiện mức xám (grey level occurrence matrices).

- **Lược đồ hiệu mức xám** dùng để mô tả các thông tin mang tính không gian và được định nghĩa như sau:

Cho  $d = (d_1, d_2)$  là véc-tơ dịch chuyển giữa 2 điểm ảnh và  $g(d)$  là hiệu mức xám với khoảng cách  $d$ :

$$g(d) = |f(k,l) - f(k+d_1, l+d_2)| \quad (6.28)$$

với  $f(k,l)$  hàm cho giá trị mức xám tại toạ độ  $(k,l)$ . Gọi  $h_g(g,d)$  là lược đồ của hiệu mức xám với khoảng cách  $d$ . Như vậy với mỗi khoảng cách  $d$  ta có một lược đồ riêng.

Với một miền ảnh có kết cấu thô, lược đồ  $h_g(g,d)$  có khuynh hướng tập trung xung quanh  $g = 0$  với khoảng cách  $d$  nhỏ. Trái lại, với một miền ảnh có kết cấu mịn,  $h_g(g,d)$  sẽ phân nhánh dù với véc-tơ dịch chuyển  $d$  là khá nhỏ. Dựa trên lược đồ này, người ta định

nghĩa lại một số đại lượng:

$$\text{- Trung bình: } \mu_e = \sum_{k=1}^N g_k h_k(g_k, d) \quad (6.29)$$

$$\text{- Phương sai: } \sigma_e^2 = \sum_{k=1}^N (g_k - \mu_e)^2 h_k(g_k, d) \quad (6.30)$$

$$\text{- Độ tương phản: } c_d = \sum_{k=1}^N g_k^2 h_k(g_k, d) \quad (6.31)$$

Phương sai đo độ tản mát của hiệu mức xám tại một khoảng cách  $d$  nào đấy. Kết cấu xác định thường có khuynh hướng có phương sai  $\sigma_e$  tương đối nhỏ. Độ tương phản  $c_d$  chính là mômen của lược đồ  $h_g(g, d)$  xung quanh  $g=0$  và đo độ tương phản của hiệu mức xám.

Người ta cũng sử dụng entropy để đo độ đồng nhất của lược đồ  $h_g$ :

$$H_g = - \sum_{k=1}^N h_g(g_k, d) \ln(h_g(g_k, d)) \quad (6.32)$$

Ưu điểm cơ bản của lược đồ hiệu mức xám là tính toán đơn giản. Ngoài ra còn có khả năng cho ta tổ chức kết cấu không gian.

- Má trận xuất hiện liên hiệp mức xám:** gọi  $P(k, l, d)$  là xác suất liên hiệp của 2 điểm ảnh  $f_k$  và  $f_l$  với các mức xám  $k, l$  tương ứng cách nhau một khoảng cách  $d$ . Xác suất này dễ dàng tính được nhờ việc tính số lần xuất hiện  $n_{k,l}$  của cặp điểm ảnh  $(f_k, f_l)$  có mức xám  $k$  và  $l$  với khoảng cách  $d$ . Gọi  $n$  là tổng số cặp liên hiệp có thể với khoảng cách  $d$  trong ảnh. Các phần tử  $c_{k,l}$  của ma trận xuất hiện liên hiệp mức xám  $c_d$  được tính như sau:  $c_{d,l} = (c_{k,l})$

$$\text{và } c_{k,l} = P(k, l, d) = \frac{n_{k,l}}{n} \quad (6.33)$$

Má trận xuất hiện liên hiệp mức xám  $C_d$  là ma trận vuông  $N \times N$  phần tử ( $N$  là số mức xám của ảnh). Ma trận này chứa đựng các thông tin rất hữu ích về tổ chức kết cấu không gian. Nếu kết cấu tương đối thô thì các phần tử của ma trận tập trung xung quanh đường chéo chính. Ngược lại, nếu kết cấu bể mặt mịn, giá trị các phần tử của  $c_d$  sẽ phân rải tương đối rõ.

Dựa trên khái niệm này người ta định nghĩa về một số độ đo:

- Xác suất cục bộ:  $p_k = \max_{0 \leq j \leq N} C_{k,j}$  (6.34)

- Entropy :  $H_0 = - \sum_{k=1}^N \sum_{l=1}^N C_{k,l} \ln(C_{k,l})$  (6.35)

Để dễ thấy entropy cục bộ khi xác suất liên hiệp  $P(k,l,d)$  có phân phối đều.

- Mô men hiệu bậc m :  $I_m = \sum_{k=1}^N \sum_{l=1}^N |k - l|^m C_{k,l}$

$I_m$  cực tiểu khi các phần tử của ma trận C tập trung trên đường chéo chính vì khoảng cách  $|k-l|^m$  là rất nhỏ.  $I_m$  nhỏ có nghĩa là kết cấu khá thô. Người ta cũng còn đưa vào một số độ đo khác như hàm tự tương quan, phổ năng lượng [Joanitas].

Để áp dụng cách tiếp cận này, cần cài đặt các giải thuật tính các đại lượng đo trên. Các giải thuật này sẽ được đề cập đến trong các tài liệu chuyên sâu.

### 6.5.2. Cách tiếp cận cấu trúc

Kết cấu sợi có cấu trúc thuần nhất là những texels xác định, mà sự xuất hiện lặp đi lặp lại tuân theo một luật tái định hay ngẫu nhiên nào đấy. Một texel về thực tế là một nhóm các pixel có cùng một số tính chất bất biến lặp trên ảnh. Một texel cũng có thể định nghĩa theo mức xám, theo bề mặt hay tính đồng nhất đối với một số các tính chất như kích thước, hướng, lược đồ bậc hai (ma trận tương tranh).

Với các texel được phân bổ ngẫu nhiên, tính kết cấu sợi tương ứng của nó được coi là yếu (weak) ngược với qui luật phân bố tái định gọi là khoẻ (strong). Khi tính kết cấu sợi là yếu, luật phân bố có thể do bởi:

- Mật độ gờ;
- Các loại dài của các texel liên thông tối đa;
- Mật độ cục trị tương đối: số pixel trên một đơn vị diện tích có mức xám cục trị cục bộ địa phương so với các lân cận.

Ngoài 2 cách tiếp cận trên, người ta còn dùng cách tiếp cận khác bằng cách lấy tổ hợp 2 cách trên và gọi là kỹ thuật *mosaic*. Mô hình này biểu diễn các quá trình hình học ngẫu nhiên, thí dụ như khám ngẫu nhiên hay đều của một mội phẳng vào các đường cong lồi sẽ làm nổi lên tính kết cấu tế bào.

### 6.5.3. Phân đoạn theo tính kết cấu

Khi đối tượng xuất hiện trên một nền có tính kết cấu cao, việc phân đoạn dựa vào tính kết cấu trở nên khá quan trọng. Nguyên nhân là vì kết cấu sợi thường chứa mật độ cao các gờ (edge) và làm cho phân đoạn dựa vào biên trở nên kém hiệu quả, trừ phi ta loại tính kết cấu. Việc phân đoạn dựa vào miền đồng nhất cũng có thể áp dụng cho các đặc trưng kết cấu và có thể dùng để phân đoạn các miền có tính kết cấu.

Nhìn chung, việc phân loại và phân vùng dựa vào kết cấu là một vấn đề khó. Ở đây, phần trình bày trong tài liệu chỉ mang tính chất giới thiệu. Có thể giải quyết vấn đề này trong thực tế nếu ta biết trước các loại kết cấu (dựa vào quy luật hay các phân bố).

### Bài tập chương 6

1. Cho ảnh số sau:

I =	1	1	1	1	2	2	2	2
	1	1	2	1	2	2	2	2
	1	1	1	1	2	2	2	2
	1	1	1	1	2	3	2	2
	4	4	5	4	5	5	5	5
	4	4	4	4	5	5	6	5
	5	5	5	5	6	5	5	5
	5	5	5	5	5	5	5	5

Hãy thực hiện việc phân vùng ảnh trên dựa vào phương pháp tách cây từ phân. Tiêu chuẩn đồng nhất dựa vào độ lệch chuẩn (công thức 6-8).

Ghi ý: Ảnh ban đầu không thuần nhất, do vậy phải chọn ngưỡng T sao cho độ lệch chuẩn σ phải lớn hơn T. Ảnh sẽ được chia làm 4 vùng và cứ tiếp tục cho đến khi không chia được nữa. Ngưỡng T ở đây là ngưỡng tổng thể.

2. Trong phân đoạn dựa vào biên, người ta sử dụng thuật toán "Di theo đường biên" như đã mô tả trong chương. Kết quả phân đoạn là một tập văn bản miêu tả các đường bao và có cấu trúc như sau:

- Số hiệu segment xxx
- Toạ độ các điểm thuộc đường bao (x,y)
- xxx yyyy

- Chu vi:  $x_{xxx}$
- Trọng tâm:  $(xxx, yyy)$
- Min:  $(xxx, yyy)$
- Max:  $(xxx, yyy)$
- Số hiệu segment tiếp

Hãy viết một thủ tục thực hiện công việc trên. Lưu ý rằng thủ tục có sử dụng kỹ thuật kiểm tra 8 liên thông để khử các đường chán rết (bao giờ do nhiều) và hiện các điểm và các đường bao.

3. Trong phân đoạn dựa theo miền đồng nhất, người ta dựa vào tính tương tự giữa các miền. Giả sử tính tương tự (tiêu chuẩn xét) là mức xám và giá trị ngưỡng cho theo tham số trong khoảng  $[0, 255]$ . Mỗi miền ứng với một mức xám nào đó. Vì lý do bộ nhớ khi gọi đệ quy, việc phân đoạn ảnh thực hiện trên một ảnh được lấy mẫu  $64 \times 64$ . Hãy viết chương trình thực hiện thuật toán đệ quy cục bộ đã miêu tả trong tài liệu.

4. Viết giải thuật phân ảnh thành 2 vùng dựa vào ngưỡng.

5. Giả sử ảnh có 256 mức xám được phân thành n vùng đều nhau. Hãy viết giải thuật thực hiện việc phân vùng trên.

6. Dựa vào giải thuật thù trong tài liệu và giả sử thêm rằng:

- Ảnh sẽ được chia làm N vùng, mỗi vùng có  $N_p$  điểm.
- Lần đầu mỗi vùng chỉ có một điểm hạt nhân đã được cung cấp.
- Ngưỡng được cung cấp là T.
- Tiêu chuẩn hợp vùng: điểm đang xét chưa được phân vào vùng nào, nó là một trong các lân cận của điểm trước và  $|I(x+\Delta x, y+\Delta y) - \text{trung bình vùng}| < T$ . Với trung bình vùng tính như trong tài liệu.

Hãy viết chương trình thực hiện phân vùng có xét đến thao tác hợp nhất vùng.

7. Dựa vào giải thuật phân đoạn, viết một thủ tục phân vùng ảnh theo kỹ thuật tách cây từ phân có sử dụng hàm Examin\_Criteria để kiểm tra tiêu chuẩn đồng nhất.

Nếu cần hiểu thêm chi tiết, hãy tham khảo trong tài liệu[12].

# 7

## NHẬN DẠNG ẢNH PATTERN RECOGNITION

Như chỉ ra trong hình 1.1-a chương 1, nhận dạng ảnh là giai đoạn cuối cùng của các hệ thống xử lý ảnh. Nhận dạng ảnh dựa trên nền tảng lý thuyết nhận dạng (pattern recognition) nói chung và đã được đề cập trong nhiều sách về nhận dạng. Ở đây, ta không nhắc lại mà chỉ trình bày mang tính chất giới thiệu một số khái niệm cơ bản và các phương pháp thường được sử dụng trong kỹ thuật nhận dạng. Và cuối cùng sẽ đề cập đến một trường hợp cụ thể về nhận dạng đó là nhận dạng chữ viết, một vấn đề đã và đang được quan tâm nhiều.

Trong lý thuyết nhận dạng nói chung và nhận dạng ảnh nói riêng có 3 cách tiếp cận khác nhau:

- Nhận dạng dựa vào phân hoạch không gian.
- Nhận dạng cấu trúc.
- Nhận dạng dựa vào kỹ thuật mạng nơron.

Hai cách tiếp cận đầu là các kỹ thuật kinh điển. Các đối tượng ảnh quan sát và thu nhận được phải trải qua giai đoạn tiền xử lý nhằm tăng cường chất lượng, làm nổi các chi tiết (chương 4), tiếp theo là trích chọn và biểu diễn các đặc trưng (chương 5 và chương 6), và cuối cùng mới qua giai đoạn nhận dạng. Cách tiếp cận thứ ba hoàn toàn khác. Nó dựa vào cơ chế đoán nhận, lưu trữ và phân biệt đối tượng mô phỏng theo hoạt động của hệ thần kinh con người. Do cơ chế đặc biệt, các đối tượng thu nhận bởi thị giác người không cần qua giai đoạn cải thiện mà chuyển ngay sang giai đoạn tổng hợp, đối sánh với các mẫu đã lưu trữ để nhận dạng. Đây là cách tiếp cận có nhiều hóa học. Các cách tiếp cận trên sẽ trình bày chi tiết trong các phần dưới đây.

### 7.1. TỔNG QUAN VỀ NHẬN DẠNG

Nhận dạng là quá trình phân loại các đối tượng được biểu diễn theo một mô hình

nào đó và gán cho chúng vào một lớp (gán cho đối tượng một tên gọi) dựa theo những quy luật và các mẫu chuẩn. Quá trình nhận dạng dựa vào những mẫu học biết trước gọi là nhận dạng có thầy hay *học có thầy* (supervised learning); trong trường hợp ngược lại gọi là *học không có thầy* (non supervised learning). Chúng ta sẽ lần lượt giới thiệu các khái niệm này.

### 7.1.1. Không gian biểu diễn đối tượng, không gian diễn dịch

#### *Không gian biểu diễn đối tượng*

Các đối tượng khi quan sát hay thu thập được, thường được biểu diễn bởi tập các đặc trưng hay đặc tính. Như trong trường hợp xử lý ảnh, ảnh sau khi được tăng cường để nâng cao chất lượng, phân vùng và trích chọn đặc tính như đã trình bày trong các chương từ chương 4 đến chương 6, được biểu diễn bởi các đặc trưng như biên, miền đồng nhất, v.v. Người ta thường phân các đặc trưng này theo các loại như: đặc trưng topô, đặc trưng hình học và đặc trưng chức năng. Việc biểu diễn ảnh theo đặc trưng nào là phụ thuộc vào ứng dụng tiếp theo.

Ở đây ta đưa ra một cách hình thức việc biểu diễn các đối tượng. Giả sử đối tượng  $X$  (ảnh, chữ viết, dấu vân tay, v.v.) được biểu diễn bởi  $n$  thành phần ( $n$  đặc trưng):  $X = \{x_1, x_2, \dots, x_n\}$ ; mỗi  $x_i$  biểu diễn một đặc tính. Không gian biểu diễn đối tượng thường gọi tắt là *không gian đối tượng*  $\mathcal{X}$  được định nghĩa:

$$\mathcal{X} = \{X_1, X_2, \dots, X_n\}$$

trong đó mỗi  $X_i$  biểu diễn một đối tượng. Không gian này có thể là vô hạn. Để tiện xem xét chúng ta chỉ xét tập  $\mathcal{X}$  là hữu hạn.

#### *Không gian diễn dịch*

Không gian diễn dịch là tập các tên gọi của đối tượng. Kết thúc quá trình nhận dạng ta xác định được tên gọi cho các đối tượng trong tập không gian đối tượng hay nói là đã nhận dạng được đối tượng. Một cách hình thức gọi  $\Omega$  là tập tên đối tượng:

$$\Omega = \{w_1, w_2, \dots, w_k\} \text{ với } w_i, i = 1, 2, \dots, k \text{ là tên các đối tượng}$$

Quá trình nhận dạng đối tượng  $f$  là một ánh xạ  $f: \mathcal{X} \rightarrow \Omega$  với  $f$  là tập các quy luật để định một phần tử trong  $\mathcal{X}$  ứng với một phần tử trong  $\Omega$ . Nếu tập các quy luật và tập tên các đối tượng là biết trước như trong nhận dạng chữ viết (có 26 lớp từ A đến Z), người ta gọi là nhận dạng có thầy. Trường hợp thứ hai là nhận dạng không có thầy. Dường nhiên trong trường hợp này việc nhận dạng có khó khăn hơn.

### 7.1.2. Mô hình và bản chất của quá trình nhận dạng

#### 7.1.2.1. Mô hình

Việc chọn lựa một quá trình nhận dạng có liên quan mật thiết đến kiểu mô tả mà người ta sử dụng để đặc tả đối tượng. Trong nhận dạng, người ta phân chia làm 2 họ lớn:

- Họ mô tả theo tham số.
- Họ mô tả theo cấu trúc.

Cách mô tả được lựa chọn sẽ xác định *mô hình* của đối tượng. Như vậy, chúng ta sẽ có 2 loại mô hình: *mô hình theo tham số và mô hình cấu trúc*.

\* **Mô hình tham số** sử dụng một vectơ để đặc tả đối tượng. Mỗi phần tử của vectơ mô tả một đặc tính của đối tượng. Thí dụ như trong các đặc trưng chúc năng, người ta sử dụng các hàm cơ sở trực giao để biểu diễn. Và như vậy ảnh sẽ được biểu diễn bởi một chuỗi các hàm trực giao. Giá trị C là đường bao của ảnh và C(i,j) là điểm thứ i trên đường bao, i = 1, 2, ..., N (đường bao gồm N điểm).

Giả sử tiếp :

$$x_0 = \frac{1}{N} \sum_{i=1}^N x_i$$

$$y_0 = \frac{1}{N} \sum_{i=1}^N y_i$$

là tọa độ tâm điểm. Như vậy, moment trung tâm bậc p, q của đường bao là:

$$\mu_{pq} = \frac{1}{N} \sum_{i=1}^N (x_i - x_0)^p (y_i - y_0)^q \quad (7.1)$$

Vectơ tham số trong trường hợp này chính là các moment  $\mu_{ij}$  với  $i=1, 2, \dots, p$  và  $j=1, 2, \dots, q$ . Còn trong số các đặc trưng hình học, người ta hay sử dụng chu tuyến, đường bao, diện tích và tỉ lệ  $T = 4\pi S/p^2$ , với S là diện tích, p là chu tuyến.

Việc lựa chọn phương pháp biểu diễn sẽ làm đơn giản cách xây dựng. Tuy nhiên, việc lựa chọn đặc trưng nào là hoàn toàn phụ thuộc vào ứng dụng. Thí dụ, trong nhận dạng chữ (số trình bày sau), các tham số là các dấu hiệu:

- số điểm chạc ba, chạc tư,
- số điểm chu trình,

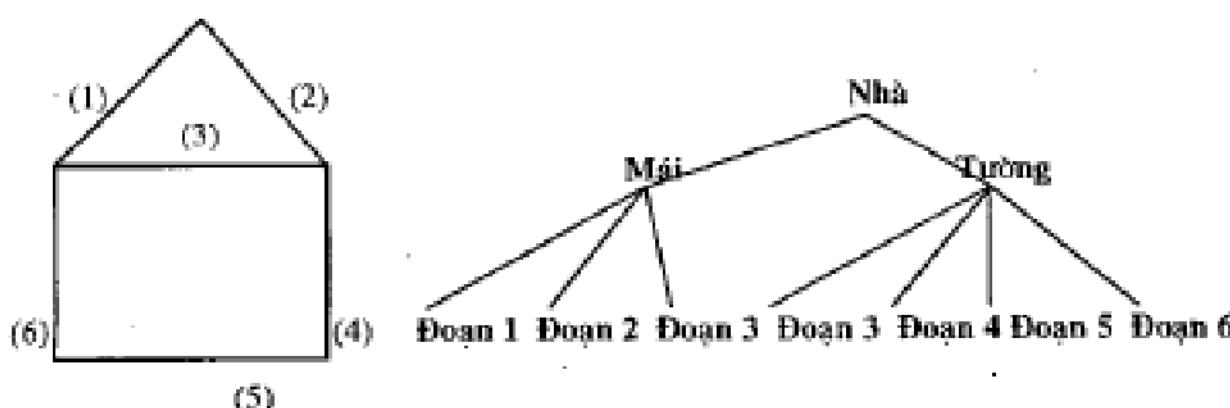
- số điểm ngoại,
- số điểm kết thúc.



• **Mô hình cấu trúc:** Cách tiếp cận của mô hình này dựa vào việc mô tả đối tượng nhờ một số khái niệm biểu thị các đối tượng cơ sở trong ngôn ngữ tự nhiên. Để mô tả đối tượng, người ta dùng một số dạng nguyên thuỷ như đoạn thẳng, cung, v.v. Chẳng hạn một hình chữ nhật được định nghĩa gồm 4 đoạn thẳng vuông góc với nhau từng đôi một. Trong mô hình này người ta sử dụng một bộ ký hiệu kết thúc  $V_e$ , một bộ ký hiệu không kết thúc gọi là  $V_n$ . Ngoài ra có dùng một tập các luật sản xuất để mô tả cách xây dựng các đối tượng phù hợp dựa trên các đối tượng đơn giản hơn hoặc đối tượng nguyên thuỷ (tập  $V_o$ ). Trong cách tiếp cận này, ta chấp nhận một khẳng định là: cấu trúc một dạng là kết quả của việc áp dụng luật sản xuất theo những nguyên tắc xác định bắt đầu từ một dạng gốc bắt đầu. Một cách bình thường, ta có thể coi mô hình này tương đương một văn phạm  $G = (V_o, V_n, P, S)$  với:

- $V_e$  là bộ ký hiệu kết thúc,
- $V_n$  là bộ ký hiệu không kết thúc,
- $P$  là luật sản xuất,
- $S$  là dạng (ký hiệu bắt đầu).

Thí dụ, đối tượng nhà gồm mái và tường, mái là một tam giác gồm 3 cạnh là 3 đoạn thẳng, tường là một hình chữ nhật gồm 4 cạnh vuông góc với nhau từng đôi một sẽ được mô tả thông qua cấu trúc mô tả dựa vào văn phạm sinh như chỉ ra trong hình 7.1 dưới đây.



Hình 7.1. Mô hình cấu trúc của một đối tượng nhà.

### 7.1.2.2. Bản chất của quá trình nhận dạng

Quá trình nhận dạng gồm 3 giai đoạn chính:

- Lựa chọn mô hình biểu diễn đối tượng.
- Lựa chọn luật ra quyết định (phương pháp nhận dạng) và suy diễn quá trình học.
- Học nhận dạng.

Khi mô hình biểu diễn đối tượng đã được xác định, có thể là định lượng (mô hình tham số) hay định tính (mô hình cấu trúc), quá trình nhận dạng chuyển sang giai đoạn học. Học là giai đoạn rất quan trọng. Thao tác học nhằm cải thiện, điều chỉnh việc phân hoạch tập đối tượng thành các lớp.

Việc nhận dạng chính là tìm ra quy luật và các thuật toán để có thể gán đối tượng vào một lớp hay nói một cách khác gán cho đối tượng một tên.

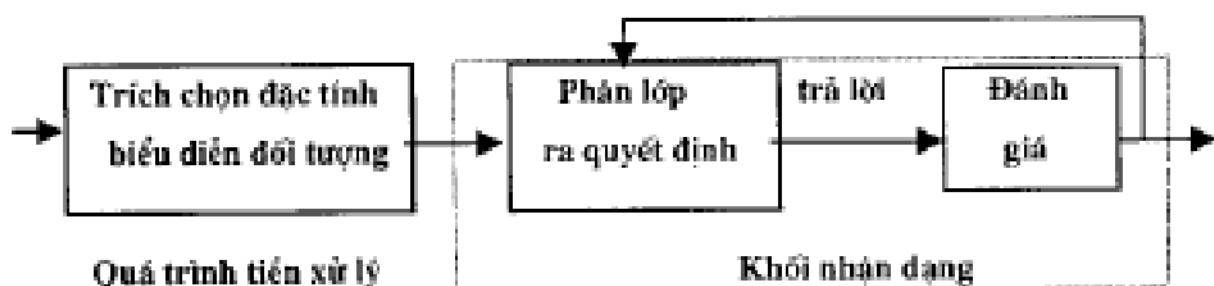
#### *Học có thầy (supervised learning)*

Kỹ thuật phân loại nhờ kiến thức biết trước gọi là học có thầy. Đặc điểm cơ bản của kỹ thuật này là người ta có một thư viện các mẫu chuẩn. Mẫu cần nhận dạng sẽ được đem sánh với mẫu chuẩn để xem nó thuộc loại nào. Thí dụ như trong một ảnh viễn thám, người ta muốn phân biệt một cánh đồng lúa, một cánh rừng hay một vùng đất hoang mà đã có các miêu tả về các đối tượng đó. Vấn đề chủ yếu là thiết kế một hệ thống để có thể đối sánh đối tượng trong ảnh với mẫu chuẩn và quyết định gán cho chúng vào một lớp. Việc đối sánh nhờ vào các thủ tục ra quyết định dựa trên một công cụ gọi là *hàm phân lớp* hay *hàm ra quyết định*. Hàm này sẽ được đề cập trong phần sau.

#### *Học không có thầy (unsupervised learning)*

Kỹ thuật học này phải tự định ra các lớp khác nhau và xác định các tham số đặc trưng cho từng lớp. Học không có thầy đương nhiên là khó khăn hơn. Một mặt, do số lớp không được biết trước, mặt khác những đặc trưng của các lớp cũng không biết trước. Kỹ thuật này nhằm tiến hành mọi cách gộp nhóm có thể và chọn lựa cách tốt nhất. Bắt đầu từ tập dữ liệu, nhiều thủ tục xử lý khác nhau nhằm phân lớp và nâng cấp dần để đạt được một phương án phân loại. Một số kỹ thuật tự học sẽ được trình bày trong phần 7.2.4.

Nhìn chung, dù là mô hình nào và kỹ thuật nhận dạng ra sao, một hệ thống nhận dạng có thể tóm tắt theo sơ đồ hình 7.2 sau:



Hình 7.2. Sơ đồ tổng quát một hệ nhận dạng.

## 7.2. NHẬN DẠNG DỰA TRÊN PHÂN HOẠCH KHÔNG GIAN

Trong kỹ thuật này, các đối tượng nhận dạng là các đối tượng định lượng. Mỗi đối tượng được biểu diễn bởi một vectơ nhiều chiều. Trước tiên, ta xem xét một số khái niệm như: phân hoạch không gian, hàm phân biệt, sau đó sẽ đi vào một số kỹ thuật cụ thể.

### 7.2.1. Phân hoạch không gian

Giả sử không gian đối tượng  $\Omega$  được định nghĩa:  $\Omega = \{X_i, i=1, 2, \dots, m\}$ ,  $X_i$  là một vectơ. Người ta nói  $\mathcal{P}$  là một phân hoạch của không gian  $\Omega$  thành các lớp  $C_i, C_i \subset \Omega$  nếu:

$$C_i \cap C_j = \emptyset \text{ với } i \neq j \text{ và } \cup C_i = \Omega$$

Nói chung, đây là trường hợp lý tưởng: tập  $\Omega$  tách được hoàn toàn. Trong thực tế, thường gặp không gian biểu diễn tách được từng phần. Như vậy phân loại là dựa vào việc xây dựng một ánh xạ  $f: \Omega \rightarrow \mathcal{P}$ . Công cụ xây dựng ánh xạ này là các hàm phân biệt (Discriminant functions).

### 7.2.2. Hàm phân lớp hay hàm ra quyết định

Để phân đối tượng vào các lớp, ta phải xác định số lớp và ranh giới giữa các lớp đó. Hàm phân lớp hay hàm phân biệt là một công cụ rất quan trọng. Gọi  $\{g_i\}$  là lớp các hàm phân lớp. Lớp hàm này được định nghĩa như sau:

nếu  $\forall i \neq k, g_k(X) > g_i(X)$  thì ta quyết định  $X \in$  lớp  $k$ .

Như vậy để phân biệt  $k$  lớp, ta cần  $k-1$  hàm phân biệt. Hàm phân biệt  $g$  của một lớp nào đó thường dùng là hàm tuyến tính, có nghĩa là:

$$g(X) = W_0 + W_1 X_1 + W_2 X_2 + \dots + W_k X_k$$

trong đó:

-  $W_i$  là các trọng số gần cho các thành phần  $X_i$ .

-  $W_0$  là trọng số để viết cho gọn.

Trong trường hợp g là tuyến tính, người ta nói là việc phân lớp là tuyến tính hay siêu phẳng (hyperplan).

Các hàm phân biệt thường được xây dựng dựa trên khái niệm khoảng cách hay dựa vào xác suất có điều kiện.

Lẽ tự nhiên, khoảng cách là một công cụ rất tốt để xác định xem đối tượng có "gần nhau" hay không. Nếu khoảng cách nhỏ hơn một ngưỡng  $\tau$  nào đó ta coi 2 đối tượng là giống nhau và gộp chúng vào một lớp. Ngược lại, nếu khoảng cách lớn hơn ngưỡng, có nghĩa là chúng khác nhau và ta tách thành 2 lớp.

Trong một số trường hợp, người ta dựa vào xác suất có điều kiện để phân lớp cho đối tượng. Lý thuyết xác suất có điều kiện được Bayes nghiên cứu khá kỹ và chúng ta có thể áp dụng lý thuyết này để phân biệt đối tượng.

Gọi:  $P(X/C_i)$  là xác suất để có  $X$  biết rằng có xuất hiện lớp  $C_i$

$P(C_i/X)$  là xác suất có điều kiện để  $X$  thuộc lớp  $C_i$ .

với  $X$  là đối tượng nhận dạng,  $C_i$  là các lớp đối tượng.

Quá trình học cho phép ta xác định  $P(X/C_i)$  và nhờ công thức Bayes về xác suất có điều kiện áp dụng trong điều kiện nhiễu biến, chúng ta sẽ tính được  $P(C_i/X)$  theo công thức:

$$P(C_i/X) = \frac{P(X/C_i)P(C_i)}{\sum_{j=1}^k P(X/C_j)P(C_j)} = \frac{P(X/C_i)P(C_i)}{P(X)} \quad (7.2)$$

Nếu  $P(C_i/X) > P(C_k/X)$  với  $\forall i \neq k$  thì  $X \in C_i$ . Tuỳ theo các phương pháp nhận dạng khác nhau, hàm phân biệt sẽ có các dạng khác nhau.

### 7.2.3. Nhận dạng thống kê

Nếu các đối tượng nhận dạng tuân theo luật phân bố Gauss, mà hàm mật độ xác suất cho bởi:

$$f(x) = \frac{1}{2\pi\sigma^2} \exp - \frac{(x-m)^2}{2\sigma^2}$$

người ta có dùng phương pháp ra quyết định dựa vào lý thuyết Bayes. Lý thuyết Bayes thuộc loại lý thuyết thống kê nên phương pháp nhận dạng dựa trên lý thuyết Bayes có tên là phương pháp thống kê.

### Quy tắc Bayes

- Cho không gian đối tượng  $\Omega = \{X_i, i=1, 2, \dots, L\}$ , với  $X_i = \{x_1, x_2, \dots, x_g\}$
- Cho không gian diễn dịch  $\Omega = \{C_1, C_2, \dots, C_r\}$ ,  $r$  là số lớp.

Quy tắc Bayes phát biểu như sau:

$$\epsilon: \Omega \rightarrow \Omega \text{ sao cho } X \in C_k \text{ nếu } P(C_k/X) > P(C_l/X) \forall l < k, l=1, 2, \dots, r.$$

Trường hợp lý tưởng là nhận dạng luôn đúng, có nghĩa là không có sai số. Thực tế, luôn tồn tại sai số  $\epsilon$  trong quá trình nhận dạng. Vấn đề ở đây là xây dựng quy tắc nhận dạng với sai số  $\epsilon$  là nhỏ nhất.

### Phương pháp ra quyết định với $\epsilon$ tối thiểu

Ta xác định  $X \in C_k$  nhờ xác suất  $P(C_k/X)$ . Vậy nếu có sai số, sai số sẽ được tính bởi  $1 - P(C_k/X)$ . Để đánh giá sai số trung bình, người ta xây dựng một ma trận  $L(r,r)$  giả thiết là có n lớp.

Ma trận  $L$  được định nghĩa như sau:

$$L_{k,j} = \begin{cases} 1_{k,j} > 0 \text{ nếu } k \neq j & (\text{tồn tại sai số}) \\ 1_{k,j} \leq 0 \text{ nếu } k = j & (\text{không có sai số}) \end{cases} \quad (7.3)$$

Như vậy, sai số trung bình của sự phân lớp sẽ là:

$$r_k(X) = \sum_{j=1}^r 1_{k,j} P(C_j/X) \quad (7.4)$$

Để sai số là nhỏ nhất ta cần có  $r_k$  là min. Từ công thức 7.2 và 7.4 ta có:

$$r_k(X) = \sum_{j=1}^r 1_{k,j} P(X/C_j) P(C_j) \quad (7.5)$$

Vậy, quy tắc ra quyết định dựa trên lý thuyết Bayes có tính đến sai số được phát biểu như sau:

$$X \in C_k \text{ nếu } p_k < p_j \text{ với } p < k, p=1, 2, \dots, r.$$

(7.6)

với  $p_k$  là  $r_k(X)$ .

Trường hợp đặc biệt với 2 lớp  $C_1$  và  $C_2$ , ta dễ dàng có:

$$X \in C_1 \text{ nếu } P(X/C_1) > \frac{1_{21} - 1_{12}}{1_{11} - 1_{21}} \frac{P(C_1)}{P(C_2)} P(X/C_2) \quad (7.7)$$

Giả sử thêm rằng xác suất phân bố là đều ( $P(C_1) = P(C_2)$ ), sai số là như nhau ta có:

$$X \in C_1 \text{ nếu } P(X/C_1) > P(X/C_2) \quad (7.8)$$

#### 7.2.4. Một số thuật toán nhận dạng tiêu biểu trong tự học

Thực tế có nhiều thuật toán nhận dạng học không có thầy. Ở đây, chúng ta xem xét 3 thuật toán hay được sử dụng: Thuật toán nhận dạng dựa vào khoảng cách lớn nhất, thuật toán K-trung bình (K mean) và thuật toán ISODATA. Chúng ta lần lượt xem xét các thuật toán này vì chúng có bước tiếp nối, cái tiến từ thuật toán này qua thuật toán khác.

##### 7.2.4.1. Thuật toán dựa vào khoảng cách lớn nhất

###### a) Nguyên tắc

Cho một tập gồm m đối tượng. Ta xác định khoảng cách giữa các đối tượng và khoảng cách lớn nhất ứng với phần tử xa nhất tạo nên lớp mới. Sự phân lớp được hình thành dần dần dựa vào việc xác định khoảng cách giữa các đối tượng và các lớp.

###### b) Thuật toán

###### Bước 1

- Chọn hạt nhân ban đầu: giả sử  $X_1 \in C_1$  gọi là lớp  $g_1$ . Gọi  $Z_1$  là phần tử trung tâm của  $g_1$ .

- Tính tất cả các khoảng cách  $D_{ij} = D(X_j, Z_1)$  với  $j = 1, 2, \dots, m$
- Tìm  $D_{k1} = \max_j D_{ij}$ .  $X_k$  là phần tử xa nhất của nhóm  $g_1$ . Như vậy  $X_k$  là phần tử trung tâm của lớp mới  $g_2$ , kí hiệu  $Z_2$ .
- Tính  $d_1 = D_{12} = D(Z_1, Z_2)$ .

###### Bước 2

- Tính các khoảng cách  $D_{j1}, D_{j2}$ .
- $D_{j1} = D(X_j, Z_1), D_{j2} = D(X_j, Z_2)$ . Đặt  $D_k^{(2)} = \max_j D_{j1}$

###### Nguyên tắc chọn

- Nếu  $D_k^{(2)} < \theta$  d, kết thúc thuật toán. Phân lớp xong.
- Nếu không, sẽ tạo nên nhóm thứ ba. Gọi  $X_k$  là phần tử trung tâm của  $g_3$ , kí hiệu  $Z_3$ .
- Tính  $d_2 = (D_{12} + D_{13} + D_{23})/3$

với  $\theta$  là ngưỡng cho trước và  $D_{13} = D(Z_1, Z_3), D_{23} = D(Z_2, Z_3)$ .

Quá trình cứ lặp lại như vậy cho đến khi phân xong. Kết quả là ta thu được các lớp với các đại diện là  $Z_1, Z_2, \dots, Z_m$ .

### 7.2.4.2. Thuật toán K trung bình (giả sử có K lớp)

#### a) Nguyên tắc

Khác với thuật toán trên, ta xét K phân tử đầu tiên trong không gian đối tượng, hay nói một cách khác ta cố định K lớp. Hàm để đánh giá là hàm khoảng cách Euclidean:

$$J_k = \sum_{x \in C_k} D(X, Z_k) = \sum_{j=1}^k D(X_j, Z_k) \quad (7.9)$$

$J_k$  là hàm chỉ tiêu với lớp  $C_k$ . Việc phân vùng cho k hạt nhân đầu tiên được tiến hành theo nguyên tắc khoảng cách cực tiểu. Ở đây, ta dùng phương pháp đạo hàm để tính cực tiểu.

Xét  $\frac{\partial J_k}{\partial Z_k} = 0$  với  $Z_k$  là biến. Ta dễ dàng có (7.9) min khi:

$$\sum_{i=1}^k (X_i - Z_k) = 0 \Rightarrow Z_k = \frac{1}{N_k} \sum_{i=1}^{N_k} X_i \quad (7.10)$$

Công thức 7.10 là giá trị trung bình của lớp  $C_k$  và điều này lý giải tên của phương pháp.

#### b) Thuật toán

- Chọn  $N_c$  phân tử (giả thiết có  $N_c$  lớp) của tập T. Gọi các phân tử trung tâm của các lớp đó là:  $X_1, X_2, \dots, X_{N_c}$  và ký hiệu là  $Z_1, Z_2, \dots, Z_{N_c}$ .

- Thực hiện phân lớp

$X \in C_k$  nếu  $D(X, Z_k) = \min_j D(X, Z_j), j=1, \dots, N_c$ . (1) là lần lặp thứ nhất.

Tính tất cả  $Z_k$  theo công thức 7.10.

Tiếp tục như vậy cho đến bước q.

$X \in C_k(q-1)$  nếu  $D(X, Z_k^{(q-1)}) = \min_j D(X, Z_j^{(q-1)})$ .

Nếu  $Z_k^{(q-1)} = Z_k^{(q)}$  thuật toán kết thúc, nếu không ta tiếp tục thực hiện phân lớp.

### 7.2.4.3. Thuật toán ISODATA

ISODATA là viết tắt của từ Interactive Self Organizing Data Analysis. Nó là thuật toán khá mềm dẻo, không cần cố định các lớp trước. Các bước của thuật toán được mô tả như sau:

- Lựa chọn một phân hoạch ban đầu dựa trên các tam bất kỳ. Thực nghiệm đã chứng minh kết quả nhận dạng không phụ thuộc vào phân hoạch ban đầu [2].
- Phân vùng bằng cách sắp các điểm vào tâm gần nhất dựa vào khoảng cách Euclidean.

- Tách đôi lớp ban đầu nếu khoảng cách lớn hơn ngưỡng  $t_1$ .
  - Xác định phân hoạch mới trên cơ sở các tâm vừa xác định lại và tiếp tục xác định tâm mới.
  - Tính tất cả các khoảng cách đến tâm mới.
  - Nhóm các vùng với tâm theo ngưỡng  $t_2$ .
- Lặp các thao tác trên cho đến khi thoả tiêu chuẩn phân hoạch.

### 7.3. NHẬN ĐẠNG THEO CẤU TRÚC

#### 7.3.1. Biểu diễn định tĩnh

Ngoài cách biểu diễn theo định lượng như đã mô tả ở trên, tồn tại nhiều kiểu đối tượng mang tính định tĩnh. Trong cách biểu diễn này, người ta quan tâm đến các dụng và mối quan hệ giữa chúng. Giả thiết rằng mỗi đối tượng được biểu diễn bởi một dãy ký tự. Các đặc tính biểu diễn bởi cùng một số ký tự. Phương pháp nhận dạng ở đây là nhận dạng logic, dựa vào hàm phân biệt là hàm Bool. Cách nhận dạng là nhận dạng các từ có cùng độ dài.

Giả sử hàm phân biệt cho mọi ký hiệu là  $g_a(x)$ ,  $g_b(x), \dots$ , tương ứng với các ký hiệu a, b, ... . Để dễ dàng hình dung, ta giả sử có từ "abc" được biểu diễn bởi một dãy ký tự  $X = [x_1, x_2, x_3, x_4]$ . Tính các hàm tương ứng với 4 ký tự và có:

$$g_a(x_1) + g_b(x_2) + g_c(x_3) + g_d(x_4)$$

Các phép cộng ở đây chỉ phép toán OR. Trên cơ sở tính giá trị cực đại của hàm phân biệt, ta quyết định X có thuộc lớp các từ "abc" hay không. Trong cách tiếp cận này, đối tượng tương đương với câu.

#### 7.3.2. Phương pháp ra quyết định dựa vào cấu trúc

##### 7.3.2.1. Một số khái niệm

Thủ tục phân loại và nhận dạng ở đây gồm 2 giai đoạn: Giai đoạn đầu là giai đoạn xác định các quy tắc xây dựng, tương đương với việc nghiên cứu một văn phạm trong một ngôn ngữ chính thống. Giai đoạn tiếp theo khi đã có văn phạm là xem xét tập các dạng có được sinh ra từ các dạng đó không? Nếu nó thuộc tập đó coi như ta đã phân loại xong. Tuy nhiên, văn phạm là một vấn đề lớn. Trong nhận dạng cấu trúc, ta mới chỉ sử dụng được một phần rất nhỏ mà thôi.

Như trên đã nói, mô hình cấu trúc tương đương một văn phạm G :  $G = [V_n, V_p, P, S]$ . Có rất nhiều kiểu văn phạm khác nhau từ chính tắc, phi ngữ cảnh,... Độc giả quan tâm xin

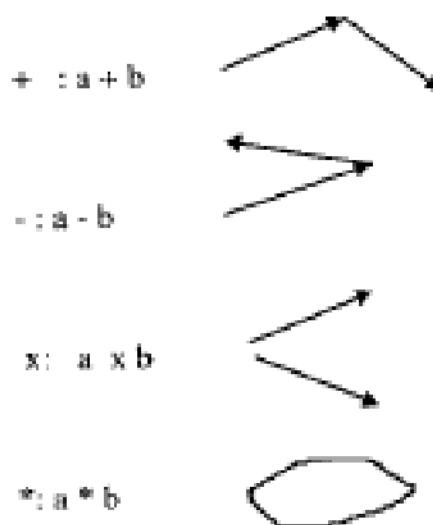
xem các tài liệu về lý thuyết ngôn ngữ hình thức hay ô tômat. Ở đây, xin giới thiệu một ngôn ngữ có thể được áp dụng trong nhận dạng cấu trúc; đó là ngôn ngữ PLD (Picture Language Description).

Ví dụ: Ngôn ngữ PLD

Trong ngôn ngữ này, các từ vựng là các vạch có hướng. Có 4 từ vựng cơ bản:



Các từ vựng trên các quan hệ được định nghĩa như sau:



Văn phạm sinh ra các mô tả trong ngôn ngữ được định nghĩa bởi:

$$G_A = \{V_n, V_T, P, S\}$$

với  $V_n = \{A, B, C, D, E\}$  và  $V_T = \{a, b, c, d\}$ . S là ký hiệu bắt đầu và P là tập luật sản xuất.

Ngôn ngữ này thường dùng nhận dạng các mạch điện.

### 7.3.2.2. Phương pháp nhận dạng

Các đối tượng cần nhận dạng theo phương pháp này được biểu diễn bởi một câu trong ngôn ngữ  $L(G)$ . Khi đó thao tác phân lớp chính là xem xét một đối tượng có thuộc văn phạm  $L(G)$  không? Nói cách khác nó có được sinh ra bởi các luật của văn phạm G không? Như vậy sự phân lớp là theo cách tiếp cận cấu trúc đòi hỏi phải xác định:

- Tập V, chung cho mọi đối tượng.
- Các quy tắc sinh P để sản sinh ra một câu và chống khác nhau đối với mỗi lớp.
- Quá trình học với các câu biểu diễn các đối tượng mẫu I nhằm xác định văn phạm G.
- Quá trình ra quyết định: xác định một đối tượng X được biểu diễn bởi một câu  $I_x$ . Nếu  $I_x$  nhận biết bởi ngôn ngữ  $L(G_x)$  thì ta nói rằng  $X \in C_x$ .

Nói cách khác, việc ra quyết định phân lớp là dựa vào phân tích cú  $G_x$  biểu diễn lớp  $C_x$  của pháp của văn phạm. Cũng như trong phân tích cú pháp - ngôn ngữ, có phân tích trên xuống, dưới lên, việc nhận dạng theo cấu trúc cũng có thể thực hiện theo cách tương tự.

Việc nhận dạng dựa theo cấu trúc là một ý tưởng và dấu sao cũng cần được nghiên cứu thêm.

#### 7.4. MẠNG NORON NHÂN TẠO VÀ NHẬN DẠNG THEO MẠNG NORON

Trước tiên, cần xem xét một số khái niệm cơ bản về bộ não cũng như cơ chế hoạt động của mạng noron sinh học. Tiếp theo, để tiện theo dõi, ở đây sẽ đề cập đến một ứng dụng của mạng noron trong nhận dạng chữ viết.

##### 7.4.1. Bộ não và noron sinh học

Các nhà nghiên cứu sinh học về bộ não cho ta thấy rằng các noron (tế bào thần kinh) là đơn vị cơ sở đảm nhiệm những chức năng xử lý nhất định trong hệ thần kinh, bao gồm não, tuỷ sống và các dây thần kinh. Mỗi noron có phần thân với nhân bên trong (gọi là soma), một dây thần kinh ra (gọi là sợi trực axon) và một hệ thống dạng cây các dây thần kinh vào (gọi là dendrite). Các dây thần kinh vào tạo thành một lưới dày đặc xung quanh thân tế bào, chiếm diện tích khoảng  $0,25 \text{ mm}^2$ , còn dây thần kinh ra tạo thành trực dài có thể từ 1 cm cho đến hàng mét. Đường kính của nhân tế bào thường chỉ là  $10^{-4}\text{m}$ . Trục dây thần kinh ra cũng có thể phân nhánh theo dạng cây để nối với các dây thần kinh vào hoặc trực tiếp với nhân tế bào các noron khác thông qua các khớp nối (gọi là synapse). Thông thường, mỗi noron có thể gồm vài chục cho tới hàng trăm ngàn khớp nối để nối với các noron khác. Người ta ước lượng rằng lưới các dây thần kinh ra cùng với các khớp nối bao phủ diện tích khoảng 90% bề mặt noron (hình 7-3).

Các tín hiệu truyền trong các dây thần kinh vào và dây thần kinh ra của các noron là

tin hiệu điện và được thực hiện thông qua các quá trình phản ứng và giải phóng các chất hữu cơ. Các chất này được phát ra từ các khớp nối dẫn tới các dây thần kinh vào sẽ làm tăng hay giảm điện thế của nhân tế bào. Khi điện thế này đạt tới một ngưỡng nào đó, sẽ tạo ra một xung điện dẫn tới trực dây thần kinh ra. Xung này được truyền theo trực, tới các nhánh sẽ khi chạm tới các khớp nối với các neuron khác sẽ giải phóng các chất truyền điện. Người ta chia làm hai loại khớp nối: khớp nối kích thích (excitatory) hoặc khớp nối ức chế (inhibitory).

Phát hiện quan trọng nhất trong ngành nghiên cứu về bộ não là các liên kết khớp thần kinh khá mềm dẻo, có thể biến động và chỉnh đổi theo thời gian tùy thuộc vào các dạng kích thích. Hơn nữa, các neuron có thể sản sinh các liên kết mới với các neuron khác và đôi khi, buổi các neuron có thể di trú từ vùng này sang vùng khác trong bộ não. Các nhà khoa học cho rằng đây chính là cơ sở quan trọng để giải thích cơ chế học của bộ não con người.

Phần lớn các quá trình xử lý thông tin đều xảy ra trên vỏ não. Toàn bộ vỏ não được bao phủ bởi mạng các tổ chức cơ sở có dạng hình thủng tròn với đường kính khoảng 0,5 mm, độ cao 4 mm. Mỗi đơn vị cơ sở này chứa khoảng 2000 neuron. Người ta chỉ ra rằng mỗi vùng não có những chức năng nhất định. Điều rất đáng ngạc nhiên chính là các neuron rất đơn giản trong cơ chế làm việc, nhưng mạng các neuron liên kết với nhau lại có khả năng tính toán, suy nghĩ, ghi nhớ và điều khiển. Có thể điểm qua những chức năng cơ bản của bộ não như sau:

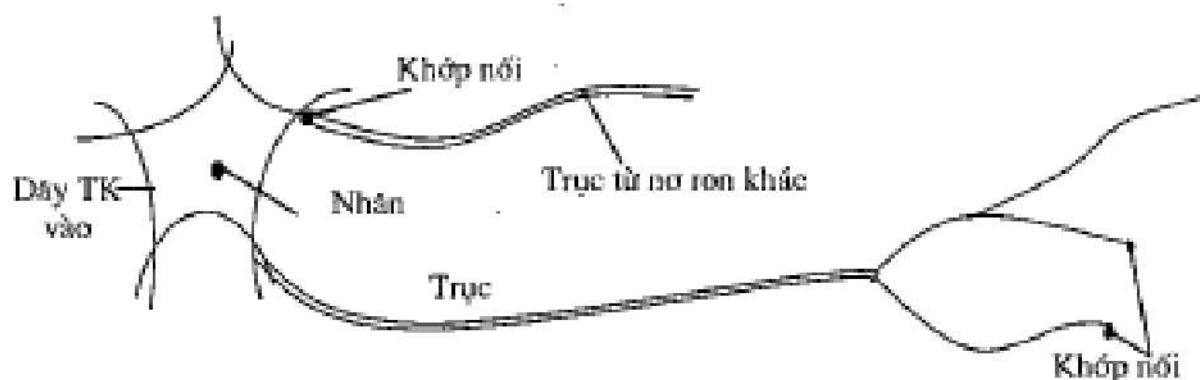
- Bộ nhớ được tổ chức theo các bộ thông tin và truy nhập theo nội dung (có thể truy xuất thông tin dựa theo giá trị các thuộc tính của đối tượng);
- Bộ não có khả năng tổng quát hoá, có thể truy xuất các tri thức hay các mối liên kết chung của các đối tượng tương ứng với một khái niệm chung nào đó;
- Bộ não có khả năng dung thứ lỗi theo nghĩa có thể điều chỉnh hoặc tiếp tục thực hiện ngay khi có những sai lệch do thông tin bị thiếu hoặc không chính xác. Ngoài ra, bộ não còn có thể phát hiện và phục hồi các thông tin bị mất dựa trên sự tương tự giữa các đối tượng;
- Bộ não có khả năng xuống cấp và thay thế dần dần. Khi có những trực trắc tại các vùng não (do bệnh, chấn thương) hoặc bắt gặp những thông tin hoàn toàn mới lạ, bộ não vẫn có thể tiếp tục làm việc;
- Bộ não có khả năng học.

## So sánh khả năng làm việc của bộ não và máy tính

	Máy tính	Bộ não người
Đơn vị tính toán	Bộ xử lý trung tâm với $10^5$ mạch logic cơ sở	Mạng $10^{11}$ neuron
Bộ nhớ	$10^9$ bit RAM	$10^{11}$ neuron
	$10^{10}$ bit bộ nhớ ngoài	với $10^{12}$ khớp nối thần kinh
Thời gian xử lý	$10^{-8}$ giây	$10^{-3}$ giây
Thông lượng	$10^9$ bit/giây	$10^{14}$ bit/giây
Cập nhật thông tin	$10^5$ bit/giây	$10^{14}$ neuron/giây

Dễ dàng thấy rằng bộ não con người có thể lưu giữ nhiều thông tin hơn các máy tính hiện đại; Tuy rằng điều này không phải đúng mãi mãi, bởi lẽ bộ não tiến hóa chậm, trong khi đó nhờ những tiến bộ trong công nghệ vi điện tử, bộ nhớ máy tính được nâng cấp rất nhanh. Hơn nữa, sự hạn kém về bộ nhớ trở nên hoàn toàn thứ yếu so với sự khác biệt về tốc độ tính toán và khả năng xử lý song song. Các bộ vi xử lý có thể tính  $10^8$  lệnh trong một giây, trong khi đó mạng neuron xử lý chậm hơn, cần khoảng vài miligiay để kích hoạt. Tuy nhiên, bộ não có thể kích hoạt hầu như cùng một lúc tại rất nhiều neuron và khớp nối, trong khi đó ngay cả máy tính hiện đại cũng chỉ có một số hạn chế các bộ vi xử lý song song. Nếu chạy một mạng neuron nhân tạo trên máy tính, phải tốn hàng trăm lệnh máy để kiểm tra một neuron có được kích hoạt hay không (tiêu phí khoảng  $10^{-3} \times 10^2$  giây/neuron). Do đó, dù bộ vi xử lý có thể tính toán nhanh hơn hàng triệu lần so với các neuron bộ não, nhưng xét tổng thể bộ não lại tính toán nhanh hơn hàng tỷ lần.

Cách tiếp cận mạng neuron nhân tạo có ý nghĩa thực tiễn rất lớn cho phép tạo ra các thiết bị có thể kết hợp khả năng song song cao của bộ não với tốc độ tính toán cao của máy tính. Tuy vậy, cần phải có một khoảng thời gian dài nữa để các mạng neuron nhân tạo có thể mô phỏng được các hành vi sáng tạo của bộ não con người. Chẳng hạn, bộ não có thể thực hiện một nhiệm vụ khá phức tạp như nhận ra khuôn mặt người quen sau không quá 1 giây, trong khi đó một máy tính tuần tự phải thực hiện hàng tỷ phép tính (khoảng 10 giây) để thực hiện cùng thao tác đó, nhưng với chất lượng kém hơn nhiều, đặc biệt trong trường hợp thông tin không chính xác, không đầy đủ.



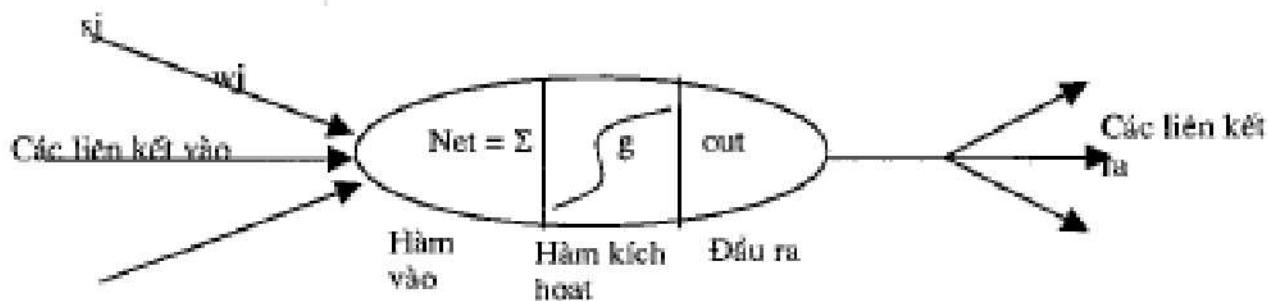
Hình 7.3 . Cấu tạo neuron sinh học.

#### 7.4.2. Mô hình mạng neuron nhân tạo

Mạng neuron nhân tạo (Artificial Neural Network) gọi tắt là MNR bao gồm các nút (đơn vị xử lý, neuron) được nối với nhau bởi các liên kết neuron. Mỗi liên kết kèm theo một trọng số nào đó, đặc trưng cho đặc tính kích hoạt/ ức chế giữa các neuron. Có thể xem các trọng số là phương tiện để lưu giữ thông tin dài hạn trong mạng neuron và nhiệm vụ của quá trình huấn luyện (học) mạng là cập nhật các trọng số khi có thêm các thông tin về các mẫu học, hay nói một cách khác, các trọng số được điều chỉnh sao cho đáng diệu vào ra của nó mô phỏng hoàn toàn phù hợp môi trường đang xem xét.

Trong mạng, một số neuron được nối với môi trường bên ngoài như các đầu ra, đầu vào.

##### 7.4.2.1. Mô hình neuron nhân tạo



Hình 7.4 . Mô hình neuron nhân tạo.

Mỗi neuron được nối với các neuron khác và nhận được các tín hiệu  $s_j$  từ chúng với các trọng số  $w_j$ . Tổng các thông tin vào có trọng số là:

$$\text{Net} = \sum w_j s_j$$

Người ta gọi đây là thành phần tuyến tính của neuron. Hàm kích hoạt g (còn gọi là hàm chuyển) đóng vai trò biến đổi từ Net sang tín hiệu đầu ra out.

$$\text{out} = g(\text{Net}).$$

Đây là thành phần phi tuyến của neuron. Có 3 dạng hàm kích hoạt thường được dùng trong thực tế:

$$\begin{aligned} \text{Hàm dạng bước} \quad \text{step}(x) &= \begin{cases} 1 \text{ nếu } x \geq 0 \\ 0 \text{ nếu } x < 0 \end{cases} \quad \text{hoặc} \quad \text{step}(x) = \begin{cases} 1 \text{ nếu } x \geq \theta \\ 0 \text{ nếu } x < \theta \end{cases} \\ \text{Hàm dấu} \quad \text{sign}(x) &= \begin{cases} 1 \text{ nếu } x \geq 0 \\ -1 \text{ nếu } x < 0 \end{cases} \quad \text{hoặc} \quad \text{sign}(x) = \begin{cases} 1 \text{ nếu } x \geq \theta \\ -1 \text{ nếu } x < \theta \end{cases} \end{aligned}$$

Hàm sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-\alpha(x+\beta)}}$$

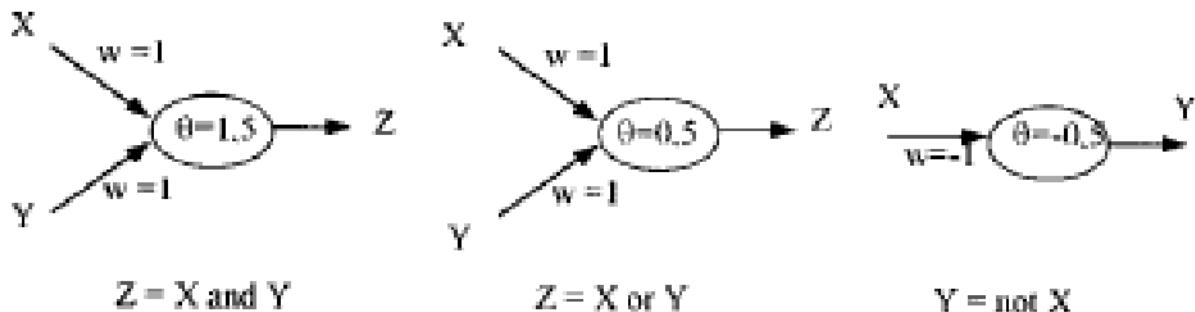
Ở đây ngưỡng  $\theta$  đóng vai trò làm tăng tính thích nghi và khả năng tính toán của mạng neuron. Sử dụng kỹ pháp véc-tơ,  $S = (s_1, \dots, s_n)$  véc-tơ tín hiệu vào,  $W = (w_1, \dots, w_n)$  véc-tơ trọng số, ta có

$$\text{out} = g(\text{Net}), \quad \text{Net} = SW.$$

Trường hợp xét ngưỡng  $\theta$ , ta dùng biểu diễn véc-tơ mới  $S = (s_1, \dots, s_n, 0)$ ,  $W = (w_1, \dots, w_n, 1)$

**Khả năng biểu diễn của neuron**

Bộ vi xử lý máy tính dựa trên tích hợp các mạch logic cơ sở. Có thể thấy rằng các neuron hoàn toàn mô phỏng khả năng tính toán của các mạch cơ sở AND, OR, NOT.



#### 7.4.2.2. Mạng neuron

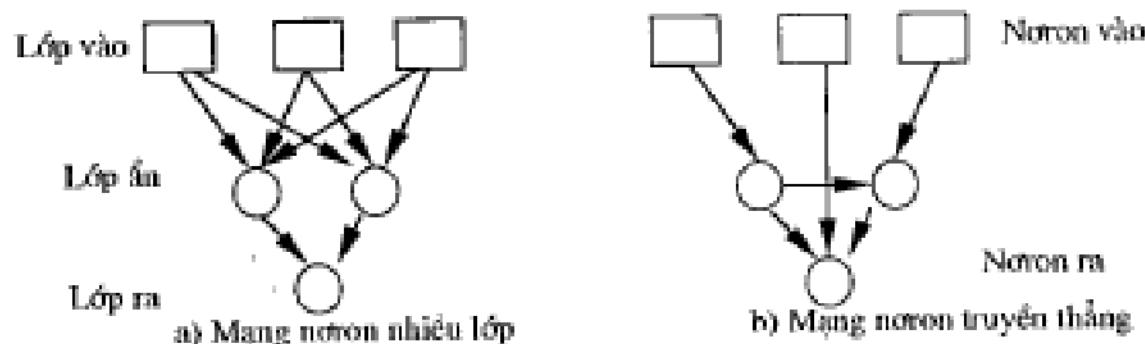
Mạng neuron là hệ thống bao gồm nhiều phần tử xử lý đơn giản (neuron) hoạt động song song. Tính năng của hệ thống này tuỳ thuộc vào cấu trúc của hệ, các trọng số liên kết

neuron và quá trình tính toán tại các neuron đơn lẻ. Mạng neuron có thể học từ dữ liệu mẫu và tổng quát hóa dựa trên các dữ liệu mẫu học. Trong mạng neuron, các neuron đơn nhận tín hiệu vào gọi là neuron vào và các neuron đưa thông tin ra gọi là neuron ra.

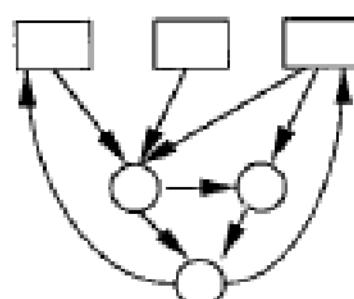
#### A. PHÂN LOẠI CÁC MẠNG NÓRON

**Theo kiểu liên kết neuron:** Ta có mạng neuron truyền thẳng (feed-forward Neural Network) và mạng neuron qui hồi (recurrent NN). Trong mạng neuron truyền thẳng, các liên kết neuron đi theo một hướng nhất định, không tạo thành đồ thị không có chu trình (Directed Acyclic Graph) với các đỉnh là các neuron, các cung là các liên kết giữa chúng. Ngược lại, các mạng qui hồi cho phép các liên kết neuron tạo thành chu trình. Vì các thông tin ra của các neuron được truyền lại cho các neuron đã góp phần kích hoạt chúng, nên mạng hồi qui còn có khả năng lưu giữ trạng thái trong của nó dưới dạng các ngưỡng kích hoạt ngoài các trọng số liên kết neuron.

**Theo số lớp:** Các neuron có thể tổ chức lại thành các lớp sao cho mỗi neuron của lớp này chỉ được nối với các neuron ở lớp tiếp theo, không cho phép các liên kết giữa các neuron trong cùng một lớp, hoặc từ neuron lớp dưới lên neuron lớp trên. Ở đây cũng không cho phép các liên kết neuron nhảy qua một lớp.



Hình 7.5 . Mạng neuron truyền thẳng và nhiều lớp



Hình 7.6. Mạng neuron hồi qui.

Để dễ dàng nhận thấy rằng các neuron trong cùng một lớp nhận được tín hiệu từ lớp trên cùng một lúc, do vậy về nguyên tắc chúng có thể xử lý song song. Thông thường, lớp neuron vào chỉ chịu trách nhiệm truyền đưa tín hiệu vào, không thực hiện một tính toán nào nên khi tính số lớp của mạng, người ta không tính lớp vào. Ví dụ, mạng neuron ở hình 7.5 có 2 lớp : một lớp ẩn và một lớp ra.

## B. HAI CÁCH NHÌN VỀ MẠNG NEURON

- Mạng neuron như một công cụ tính toán:

Giả sử mạng neuron NN có m neuron vào và n neuron ra, khi đó với mỗi vector các tín hiệu vào  $X = (x_1, \dots, x_m)$ , sau quá trình tính toán tại các neuron ẩn, ta nhận được kết quả ra  $Y = (y_1, \dots, y_n)$ . Theo nghĩa nào đó mạng neuron làm việc với tư cách một bảng tra, mà không cần biết dạng phụ thuộc hàm tường minh giữa Y và X. Khi đó ta viết :

$$Y = \text{Tính}(X, NN)$$

Cần lưu ý thêm rằng các neuron trên cùng một lớp có thể tính toán đồng thời, do vậy độ phức tạp tính toán nói chung sẽ phụ thuộc vào số lớp mạng.

Các thông số cấu trúc mạng neuron bao gồm:

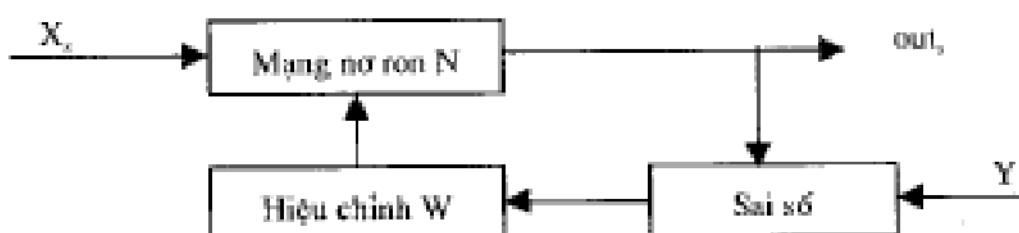
- + Số tín hiệu vào , số tín hiệu ra.
- + Số lớp neuron.
- + Số neuron trên mỗi lớp ẩn.
- + Số lượng liên kết của mỗi neuron (liên kết đầy đủ, liên kết bộ phận và liên kết ngẫu nhiên).
- + Các trọng số liên kết neuron.

- Mạng neuron như một hệ thống thích nghi có khả năng học (huấn luyện) để tinh chỉnh các trọng số liên kết cũng như cấu trúc của mình sao cho phù hợp với các mẫu học (samples). Người ta phân biệt ba loại kỹ thuật học: (i) học có quan sát (supervised learning) hay còn gọi là học có thầy; (ii) học không có giám sát (unsupervised learning) hay còn gọi là học không có thầy và (iii) học tăng cường. Trong học có giám sát, mạng được cung cấp một tập mẫu học  $(X_i, Y_i)$  theo nghĩa  $X_i$  là các tín hiệu vào, thì kết quả ra đúng của hệ phải là  $Y_i$ . Ở mỗi lần học, vector tín hiệu vào  $X_i$  được đưa vào mạng, sau đó so sánh sự sai khác giữa các kết quả ra đúng  $Y_i$  với kết quả tính toán out. Sai số này sẽ được dùng để hiệu chỉnh lại các trọng số liên kết trong mạng. Quá trình cứ tiếp tục cho đến khi thỏa mãn một tiêu chuẩn nào đó. Có hai cách sử dụng tập mẫu học: hoặc dùng các mẫu lần lượt, hết mẫu

này đến mẫu khác, hoặc sử dụng đồng thời tất cả các mẫu một lúc. Các mạng với cơ chế học không giám sát được gọi là các mạng tự tổ chức. Các kỹ thuật học trong mạng nơron có thể nhắm vào hiệu chỉnh các trọng số liên kết (gọi là học tham số) hoặc điều chỉnh, sửa đổi cấu trúc của mạng bao gồm số lớp, số nơron, kiểu và trọng số các liên kết (gọi là học cấu trúc). Cả hai mục đích học này có thể thực hiện đồng thời hoặc tách biệt.

**Học tham số:** Giả sử có k nơron trong mạng và mỗi nơron có đúng 1 liên kết vào với các nơron khác. Khi đó, ma trận trọng số liên kết W sẽ có kích thước  $k \times k$ . Các thủ tục học tham số nhằm mục đích tìm kiếm ma trận W sao cho

$$Y_i = \text{Tính} (X_i, W) \text{ đối với mọi mẫu học } S = (X_i, Y_i) \quad (1)$$



Hình 7.7. Học tham số có giám sát.

**Học cấu trúc:** Với học tham số ta giả định rằng mạng có một cấu trúc cố định. Việc học cấu trúc của mạng truyền thẳng gắn với yêu cầu tìm ra số lớp của mạng L và số nơron trên mỗi lớp  $n_j$ . Tuy nhiên, với các mạng hồi qui còn phải xác định thêm các tham số ngưỡng  $\theta$  của các nơron trong mạng. Một cách tổng quát phải xác định bộ tham số  $P = (L, n_1, \dots, n_p, \theta_1, \dots, \theta_L)$ .

ở đây  $k = \sum n_j$  sao cho

$$Y_i = \text{Tính} (X_i, P) \text{ đối với mọi mẫu học } S = (X_i, Y_i) \quad (2)$$

Về thực chất, việc điều chỉnh các vectơ tham số W trong (1) hay P trong (2) đều qui về bài toán tìm kiếm tối ưu trong không gian tham số. Do vậy, có thể áp dụng các cơ chế tìm kiếm kinh điển theo gradient hay các giải thuật di truyền, lập trình tiến hóa.

### C. KHẢ NĂNG TÍNH TOÁN VÀ BIỂU DIỄN PHỤ THUỘC DỮ LIỆU CỦA MẠNG NƠRON

Mạng nơron truyền thẳng chỉ đơn thuần tính toán các tín hiệu ra dựa trên các tín hiệu vào và các trọng số liên kết nơron đã xác định sẵn ở trong mạng. Do đó chúng không có trạng thái bên trong nào khác ngoài vectơ trọng số W. Đối với mạng hồi qui, trạng thái trong của mạng được lưu giữ tại các ngưỡng của các nơron. Điều này có nghĩa là quá trình

tính toán trên mạng truyền thông có lớp lang hơn trong mạng qui hồi. Nói chung, các mạng qui hồi có thể không ổn định, thậm chí rối loạn theo nghĩa, khi cho vector giá trị đầu vào  $X$  nào đó, mạng cần phải tính toán rất lâu, thậm chí có thể bị lặp vô hạn trước khi đưa ra được kết quả mong muốn. Quá trình học của mạng qui hồi cũng phức tạp hơn rất nhiều. Tuy vậy, các mạng qui hồi có thể cho phép mô phỏng các hệ thống tương đối phức tạp trong thực tế.

#### D. XÁC ĐỊNH CẤU TRÚC MẠNG TỐI ƯU.

Như đã nói ở trên, lựa chọn sai cấu trúc mạng có thể dẫn tới hoạt động mạng trở nên kém hiệu quả. Nếu ta chọn mạng quá nhỏ có thể chúng không biểu diễn được sự phụ thuộc dữ liệu mong muốn. Nếu chọn mạng quá lớn để có thể nhớ được tất cả các mẫu học dưới dạng bảng tra, nhưng hoàn toàn không thể tổng quát hóa được cho những tín hiệu vào chưa biết trước. Nói cách khác, cũng giống như trong các mô hình thống kê, các mạng neuron có thể đưa tới tình trạng quá thừa tham số.

Bài toán xác định cấu trúc mạng tốt có thể xem như bài toán tìm kiếm trong không gian tham số (xem phần học cấu trúc và học tham số). Một cách làm là sử dụng giải thuật di truyền. Tuy vậy, không gian tham số có thể rất lớn và để xác định một trạng thái  $W$  (hoặc  $P$ ) trong không gian đòi hỏi phải huấn luyện mạng, do vậy rất tốn thời gian. Có thể áp dụng tư tưởng tìm kiếm leo dốc (hill-climbing) nhằm sửa đổi một cách có lựa chọn, mang tính địa phương cấu trúc mạng hiện có. Có hai cách làm:

- + Hoặc bắt đầu với một mạng lớn, sau đó giảm nhỏ xuống;
- + Hoặc bắt đầu với một mạng nhỏ, sau đó tăng dần lên.

Một kỹ thuật khác có thể áp dụng gọi là "Tốn thương tối ưu" nhằm loại bỏ một số liên kết trọng số trong mạng dựa trên cách tiếp cận lý thuyết thông tin. Đơn giản nhất là các liên kết có trọng số bằng 0. Quá trình cứ tiếp tục như vậy. Thực nghiệm chỉ ra rằng, kỹ thuật này có thể loại trừ tới  $3/4$  các liên kết, do đó nâng cao đáng kể hiệu quả của mạng.

Ngoài việc loại trừ các liên kết neuron thừa, người ta có thể vẫn bỏ những neuron không đóng góp nhiều vào quá trình thực hiện của mạng.

Giải thuật "Lớp ngồi" là một biến thể của kỹ thuật tăng trưởng mạng xuất phát từ cấu hình ban đầu tương đối nhỏ. Ý tưởng ở đây là xác định một cấu hình mạng cho phép tính đúng các mẫu học đã biết. Sau đó, mỗi khi thêm dân mẫu học mới, mạng được phép thêm một số neuron cho phép đoán đúng kết quả học hiện tại và quá trình cứ tiếp tục như vậy.

### 7.4.3. Các mạng nơron một lớp

#### 7.4.3.1. Mạng Hopfield

Năm 1982 nhà vật lý người Mỹ J.J. Hopfield đã đề xuất mô hình mạng nơron một lớp NN cho phép tạo ảnh xạ dữ liệu từ tín hiệu vào sang tín hiệu ra theo kiểu tự kết hợp (auto-association) tức là nếu tín hiệu vào là  $X$  thuộc miền giá trị  $D$  nào đó thì kết quả ra  $Y$ :

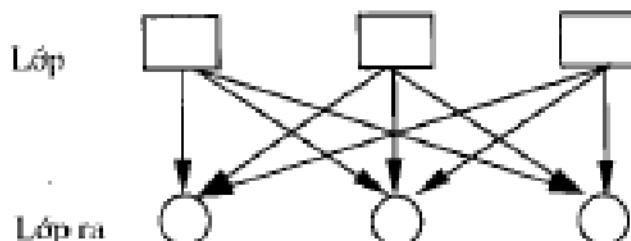
$$Y = \text{Tinh}(X, NN) \text{ cũng thuộc vào miền } D \text{ đó.}$$

Như vậy, một vector tín hiệu vào  $X$  bị thiếu thông tin hoặc biến dạng có thể được phục hồi dạng nguyên bản của mình.

Trong ứng dụng, mạng Hopfield đã mở rộng được khả năng tự kết hợp (hồi tưởng) của bộ não người, nhận ra người quen sau khi nhận thấy những nét quen thuộc trên khuôn mặt. Ngoài ra, với một số cải biến mạng Hopfield còn được dùng để giải quyết các bài toán tối ưu, bài toán xử lý dữ liệu trong điều khiển tự động.

## A. KIẾN TRÚC MẠNG

Mạng Hopfield có một lớp ra, với số nơron bằng số tín hiệu vào. Các liên kết nơron là đầy đủ.



Hình 7.8. Mạng Hopfield.

Nếu có  $m$  tín hiệu vào thì ma trận trọng số  $W$  sẽ có kích cỡ  $m \times m$ :  $W=(w_{ij})$  trong đó  $w_{ij}$  là trọng số liên kết nơron thứ  $j$  ở lớp vào sang nơron thứ  $i$  ở lớp ra (Các hàng tương ứng với nơron ra, các cột tương ứng với nơron vào).

Mạng nơron Hopfield yêu cầu các tín hiệu vào có giá trị luồng cực -1 và 1. Trường hợp dấu vào  $x$  nhị phân có thể dùng hàm biến đổi  $x' = 2x - 1$ .

Hàm kích hoạt được dùng tại các nơron là hàm dấu.

$$\text{out}_j = \text{sign}(\text{Net}_j) = \text{sign}\left(\sum_{i=1}^n w_{ij} x_i\right) \quad (3)$$

## B. HUẤN LUYỆN MẠNG

Mạng Hopfield HF học dựa trên nguyên tắc có giám sát. Giả sử có  $p$  mẫu học tương ứng với các vector tín hiệu vào  $X_s$ ,  $s=1, p$ . Mạng sẽ xác định bộ trọng số  $W$  sao cho

$$X_s = \text{Tính} (X_s, W) \text{ với mọi } s=1,p \quad (4)$$

Ta xây dựng ma trận trọng số  $W$  như sau:  $W = (w_{ij})$  với

$$w_{ij} = \begin{cases} \sum_{s=1}^p x_{sj} x_{si} & \text{Nếu } i \neq j \\ 0 & \text{Nếu } i=j \end{cases} \quad (5)$$

ở đây  $X_s = (x_{s1}, \dots, x_{sm})$ .

Một cách trực quan, trọng số liên kết  $w_{ij}$  sẽ tăng thêm một lượng là 1 (tương ứng với số hạng  $x_{sj} x_{si}$ ) nếu cả hai thành phần thứ  $i$  và thứ  $j$  của mẫu học  $X_s$  bằng nhau. Khi có mẫu học mới  $X_{p+1}$  ta chỉ cần xét các thành phần thứ  $i$  và thứ  $j$  của nó để cập nhật giá trị cho  $w_{ij}$  (6). Có thể chứng minh được với ma trận  $W$  được xác định như trong (5), ta sẽ có được (4). Nói cách khác, mạng đã "học thuộc" các ví dụ mẫu  $\{X_s\}$ .

### C. SỬ DỤNG MẠNG.

Giả sử đưa vào mạng vectơ tín hiệu  $\bar{X}$ .

Sử dụng mạng để tính đầu ra tương ứng với tín hiệu vào  $\bar{X}$  là quá trình lặp bao gồm các bước:

1. Ban đầu, đặt  $X^{(0)} = \bar{X}$ . Gọi  $Y^{(0)}$  là vectơ tín hiệu ra tương ứng với một lần cho  $X^{(0)}$  lan truyền trong mạng.

$$Y^{(0)} = \text{out}^{(0)} = \text{Tính} (HF, X^{(0)}).$$

Nếu  $Y^{(0)} \neq X^{(0)}$  thì tiếp tục bước lặp với  $t=t+1$  và  $X^{(t+1)} = Y^{(t)} = \text{out}^{(t)}$

2. Nếu  $Y^{(t)} = X^{(t)}$  thì dừng và khi đó  $X^{(t)}$  được coi là kết quả xử lý của mạng khi có tín hiệu vào  $\bar{X}$ .

Điểm chú ý quan trọng là ma trận  $W$  không thay đổi trong quá trình sử dụng mạng.

Một vài tình huống này sinh:

1) Mạng không hội tụ.

2) Mạng hội tụ và  $X^{(0)} = \bar{X}$

3) Mạng hội tụ và  $X^{(0)} = X_s$  với  $X_s$  là mẫu nào đó đã học.

4) Mạng hội tụ với  $X^{(0)} \neq X_s$  với mọi mẫu học  $X_s$

5) Mạng hội tụ với  $X^{(0)}$  nào đó như trong 2) 3) 4) nhưng là ảnh ngược (-1 thành 1, -1 thành -1).

6) Mạng có thể đưa ra luận phiến một vài mẫu học (hoặc ảnh ngược của chúng).

Trường hợp 2) có nghĩa rằng vectơ  $\bar{X}$  đã được đoán nhận đúng dựa trên mẫu học  $\{X_i\}$  hay nói cách khác,  $\bar{X}$  có thể suy ra từ mẫu học.

Trường hợp 3) chúng tỏ rằng mạng đã phục hồi dạng nguyên bản  $X$ , của  $\bar{X}$ .

Trường hợp 4) chỉ ra một vectơ mới, có thể xem là mẫu học và sẽ được dùng để cập nhật ma trận trọng số (xem (6)).

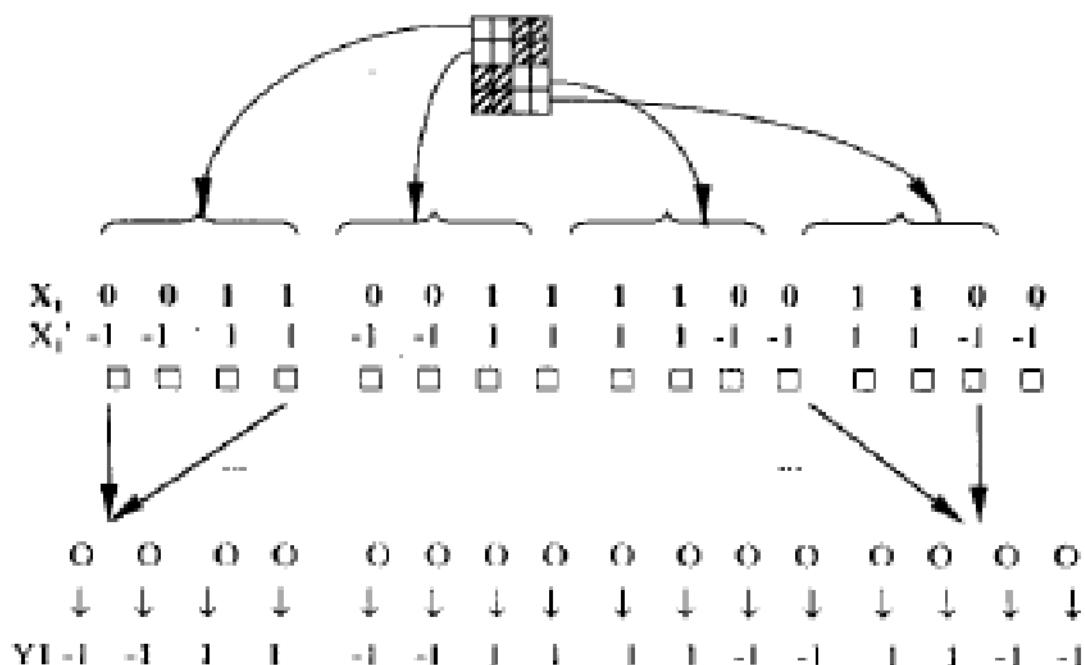
#### D. THỬ NGHIỆM MẠNG TRONG PHỤC HỒI ẢNH

Xét bài toán phục hồi ảnh đen trắng kích cỡ  $4 \times 4$ . Như vậy mỗi ảnh có 16 điểm ảnh. Ta thiết kế một mạng HF với 16 đầu vào và 16 neuron ra. Vecto đầu vào của mạng nhận được từ ma trận ảnh, lấy từng dòng một, sau khi đã biến đổi nhờ sử dụng hàm  $x' = 2x - 1$ .

Ban đầu ta có 4 mẫu

$X_1 = (0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0)$	$X_2 = (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0)$
$X_3 = (1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1)$	$X_4 = (1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1)$

Hình 7.9. Mẫu học.



Hình 7.10. Mạng Hopfield khôi phục ảnh.

Ma trận W được tính theo công thức (5) cho kết quả sau:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	2	0	0	2	0	-2	0	-2	-4	-2	0	0	2	4	4	1
2	0	2	2	0	2	0	2	-4	-2	0	2	-2	0	2	2	2
0	2	0	4	-2	0	2	4	-2	0	-2	0	0	2	0	0	3
0	2	4	0	-2	0	2	4	-2	0	-2	0	0	2	0	0	4
2	0	-2	-2	0	2	0	-2	0	-2	0	-2	-2	0	2	2	5
0	2	0	0	2	0	2	0	-2	0	2	0	-4	-2	0	0	6
-2	0	2	2	0	2	0	2	0	2	0	-2	-2	0	-2	-2	7
0	2	4	4	-2	0	2	0	-2	0	-2	0	0	2	0	0	8
-2	-4	-2	-2	0	-2	0	-2	0	2	0	-2	2	0	-2	-2	9
-4	-2	0	0	-2	0	2	0	2	0	2	0	0	-2	-4	-4	10
-2	0	-2	-2	0	2	0	-2	0	2	0	2	-2	-4	-2	-2	11
0	2	0	0	-2	0	-2	0	-2	0	2	0	0	-2	0	0	12
0	-2	0	0	-2	-4	-2	0	2	0	-2	0	0	2	0	0	13
2	0	2	2	0	-2	0	2	0	-2	-4	-2	2	0	2	2	14
4	2	0	0	2	0	-2	0	-2	-4	-2	0	0	2	0	4	15
4	2	0	0	2	0	-2	0	-2	-4	-2	0	0	2	4	0	16

Kết quả thử nghiệm với các ảnh có nhiễu tại 2,5,13 điểm ảnh (tương ứng với 13, 31 và 81%) được cho trên hình 7.11. Hơn nữa, với ảnh đầu vào có cùng số điểm ảnh biến dạng có thể dẫn tới những hành vi khác nhau (không hội tụ giống nhau, số vòng lặp khác nhau ...). Nếu có hơn 50% điểm ảnh biến dạng thì ảnh được tái tạo ở đầu ra là ám bản của ảnh gốc.

### E . KHẢ NĂNG NHỎ MẪU CỦA MẠNG HOPFIELD

Kết quả thực nghiệm chỉ ra rằng số neuron  $N^{out}$  nói chung gấp 7 lần số ảnh mẫu  $N^{inh}$  cần phải nhớ (đã khôi phục) trong mạng:

$$N^{out} = 7 \cdot N^{inh} \quad (7)$$

Từ công thức này rút ra hai điều:

Thứ nhất, độ phân giải  $r \times r$  của ảnh phụ thuộc vào cần phải nhớ bao nhiêu ảnh mẫu. Chẳng hạn, nếu cần nhớ 100 ảnh mẫu thì cần phải có 700 neuron, mỗi neuron tương ứng với một điểm ảnh. Do vậy,

$$r^2 = N^{out} = 7 \cdot N^{inh} = 700, \text{ do đó ảnh phải có độ phân giải } 27 \times 27.$$

	Mẫu X <sub>1</sub>	Mẫu X <sub>2</sub>	Mẫu X <sub>3</sub>	Mẫu X <sub>4</sub>
Ảnh gốc				
Ảnh nhiễu tại 2 điểm				
Ảnh kết quả số lần lặp				
Ảnh nhiễu tại 5 điểm				
Ảnh nhiễu tại 13 điểm				
	1	2	1	2
	4	4	5	3
	2	2	2	3

Hình 7.11. Thí nghiệm mạng với ảnh nhiễu.

Thứ hai, kích thước ma trận trọng số sẽ là  $m^2 = (N^{món})^2 = 49$  ( $N^{món}$ )<sup>2</sup>

Nếu cần nhớ 100 ảnh mẫu, cần phải lưu giữ 490.000 trọng số, mỗi trọng số cần 2 hoặc 4 byte; Do vậy, để lưu giữ thông tin về mạng cần phải mất cỡ 1Mbyte hoặc 2Mbyte. Đây chính là độ phức tạp của mạng Hopfield.

Để ước lượng chi phí thời gian, ta làm như sau:

Mỗi lần lặp để tính out<sup>(t)</sup> từ X<sup>(t)</sup> ta cần chi phí cỡ  $2 \cdot 10^6$  phép nhân và  $2 \cdot 10^6$  phép cộng. Một ước lượng thô là độ phức tạp thời gian của mạng Hopfield tăng theo luỹ thừa bậc 2 của kích cỡ bài toán.

Hệ thức (7) chỉ đúng khi các mẫu học phân bố ngẫu nhiên trong không gian mẫu. Nếu phân bố hoặc lựa chọn mẫu học tốt, có thể tăng khả năng nhớ mẫu của mạng từ 0,14 mẫu/l 1 neuron lên tới 0,25 mẫu / 1 neuron. Trong ví dụ đã xét, ta chỉ có 4 mẫu ( $N^{món} = 4$ ) dùng cho mạng với  $N^{món} = 4 \times 4 = 16$  neuron. Khả năng nhớ mẫu của nó là  $4/16 = 0,25$ .

## MỘT SỐ ĐIỂM LUU Ý VỀ MẠNG HOPFIELD

- + Mạng Hopfield cho phép tạo ảnh xạ tự kết hợp trên các tập dữ liệu.

- + Dữ liệu vào, ra có giá trị lưỡng cực.
- + Học có giám sát.
- + Mạng cho phép phục hồi dữ liệu.
- Khả năng nhớ mẫu phụ thuộc vào số nơron của mạng.

#### 7.4.3.2. Mạng kiểu bộ nhớ 2 chiều kết hợp thích nghi (Adaptive Bidirectional Associative Memory Neural Network)

Có chữ "hai chiều" trong tên gọi của mạng là vì có thể dùng mạng để biến đổi tín hiệu vào thành tín hiệu ra và ngược lại, nghĩa là:

$$Y = \text{Tinh}(X, W)$$

và  $X = \text{Tinh}(Y, W)$  ở đây  $W^T$  là ma trận chuyển vị của  $W$ . Chữ "thích nghi" có nghĩa là mạng có thể điều chỉnh ma trận trọng số cho phù hợp với đặc điểm của môi trường.

Theo một nghĩa nào đó, mạng ABAM có những nét giống mạng Hopfield:

- + Chúng cũng là mạng 1 lớp.
- + Tín hiệu vào có thể là nhị phân, hoặc lưỡng cực.
- + Việc xác định ma trận trọng số ban đầu giống nhau.

Điểm khác giữa 2 loại mạng chính là ở phạm vi bài toán có thể giải quyết và cách xác định các trọng số cho phù hợp với các bài toán đó:

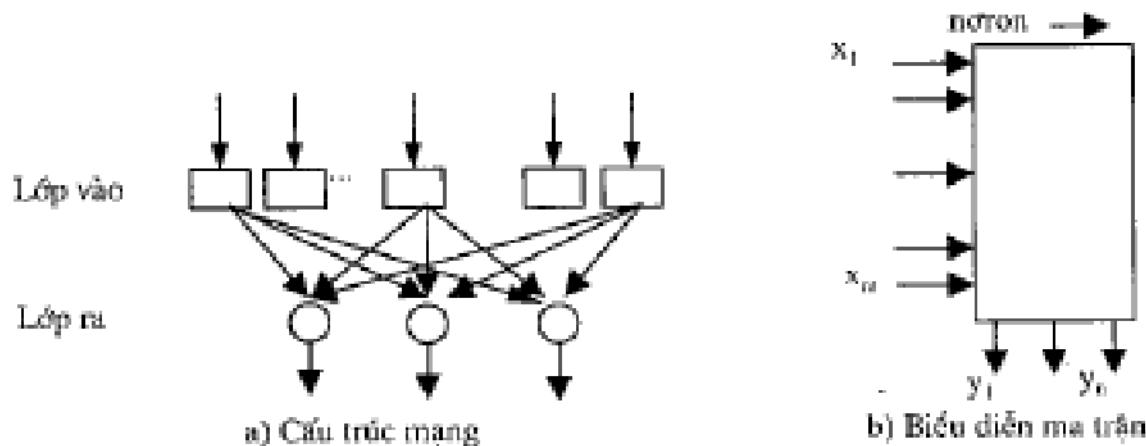
Mạng Hopfield được xác định đúng một lần và được dùng cho tất cả các bước tính toán. Kích thước của ảnh (số điểm ảnh trong mỗi mẫu) sẽ xác định số nơron và số trọng số liên kết, trong khi đó số mẫu học và hình dạng của chúng sẽ xác định giá trị các trọng số.

$$w_j^{(k)} = \sum_{i=1}^n x_i y_{ij} \quad (8)$$

Với mạng ABAM, ma trận trọng số không bắt buộc phải vuông. Thông thường, số nơron ra ít hơn nhiều số nơron vào. Ban đầu, ma trận trọng số được xác định dựa trên các tập mẫu  $\{(X_i, Y_i)\}$  giống như đối với mạng Hopfield nghĩa là:

Ở các bước tiếp theo trong quá trình học, ma trận trọng số  $W^{(k)}$  được thay đổi cho phù hợp sao cho tạo ra sự kết hợp thực sự 2 chiều giữa tín hiệu vào và tín hiệu ra trong tập mẫu học.

### A. KIẾN TRÚC MẠNG.



Hình 7.12 Mạng ABAM.

### B. HUẤN LUYỆN MẠNG.

Giả sử có tập mẫu học  $\{(X_i, Y_i)\}$ .

Sơ đồ quá trình học được thể hiện như sau:

Lặp  $\{(X_i, Y_i)\} \rightarrow$  ma trận  $W^{(0)}$

$$X_i W^{(0)T} \rightarrow Y_i^{(0)}, Y_i^{(0)} \rightarrow X_i^{(0)}$$

$\{(X_i^{(0)}, Y_i^{(0)})\} \rightarrow$  ma trận  $W^{(1)}$

$$X_i^{(0)} W^{(1)T} \rightarrow Y_i^{(1)}, Y_i^{(1)} \rightarrow X_i^{(1)}$$

cho đến khi  $\{(X_i^{(n)}, Y_i^{(n)})\} = \{(X_i, Y_i)\}$

+ Từ tập các mẫu học  $\{(X_i^{(n)}, Y_i^{(n)})\}$  xác định ma trận  $W^{(n)}$  theo công thức (8)

$$\text{Ban đầu } X_i^{(0)} = X_i \text{ và } Y_i^{(0)} = Y_i.$$

+ Sản sinh tập mẫu học mới  $\{(X_i^{(n+1)}, Y_i^{(n+1)})\}$  nhờ nhân ma trận  $W^{(n)}$  và chuyển vị của nó  $W^{(n)T}$  với các mẫu học gốc  $\{(X_i, Y_i)\}$ .

So sánh tập mẫu học mới và mẫu học gốc. Nếu trùng nhau thì dừng. Ngược lại, tiếp tục quá trình lặp.

Một số tình huống

- 1) Quá trình học không hội tụ.
- 2) Quá trình học hội tụ  $\{(X_i^{(n)}, Y_i^{(n)})\} = \{(X_i, Y_i)\}$ .

đồng thời  $X_i W^{0T} = Y_i$  và  $Y_i W^{0T} = X_i$  với mọi  $i$ .

3) Quá trình học dừng tại thời điểm t ứng với

$\{X_i^{(0)}\} = \{X_i\}$  và  $\{Y_i^{(0)}\} = \{Y_i\}$  nhưng có một mẫu sao cho  $X_i W^{0T} \neq Y_i$  hoặc  $Y_i W^{0T} \neq X_i$ .

4) Quá trình học dừng tại thời điểm t với

$\{(X_i^{(0)}, Y_i^{(0)})\} \subset \{(X_i, Y_i)\}$

5) Quá trình học dừng tại thời điểm t với

$\{(X_i^{(0)}, Y_i^{(0)})\}$  sao cho  $\{(X_i^{(0)}, Y_i^{(0)})\} = \{(X_i, Y_i)\}$ .

hoặc  $\{(X_i^{(0)}, Y_i^{(0)})\} = \{(X_i, Y_i)\}$ .

ở đây ta hiểu  $\overline{X}$  là vectơ đảo của  $\overline{X}$  (thay 1 thành -1 và ngược lại)

### C. SỬ DỤNG MẠNG

-Giả sử đã luyện tập mạng với tập mẫu  $\{(X_i, Y_i)\}$  và quá trình hội tụ đến ma trận trọng số  $W^{(0)}$  sao cho  $X_i W_i^{(0)T} = Y_i$  và  $Y_i W_i^{(0)T} = X_i$ .

Khi đó, ta có thể sử dụng như bảng tra 2 chiều:

- Biết tín hiệu vào  $X$  ta có thể xác định kết quả ra tương ứng  $Y = X W^{(0)T}$ .
- Biết tín hiệu ra  $Y$  ta có thể xác định tín hiệu vào tương ứng  $X = Y W^{(0)}$ .

Độ phức tạp bộ nhớ và độ phức tạp thời gian của mạng ABAM cỡ khoảng O(mn)

### D. THỬ NGHIỆM MẠNG TRONG NHẬN DẠNG, PHÂN LOẠI ẢNH

Ta xây dựng mạng BAM nhằm phân loại các ảnh cỡ 5x5. Giả sử có 5 mẫu học như trên hình vẽ 7.13. Ta摹 hoai các vectơ ra sao cho chúng trực giao với nhau. Để đơn giản ta viết  $Y$  dưới dạng nhị phân.

$X_1$	$X_2$	$X_3$	$X_4$	$X_5$
$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$

Hình 7.13. Mẫu học.

Ban đầu ma trận trọng số liên kết  $W(0)$  được cho trong hình 7.12. Ma trận này cũng

là ma trận trọng số nhận được sau khi học vì lẽ quá trình lặp không xảy ra. Nếu ta thay đổi một điểm ảnh (4% sai số) bất kỳ trên các ảnh vào mẫu, sẽ nhận được bảng kết quả sau:

Mẫu	Đích	Kết quả tính	Số ảnh vào	Kết quả tính	Số ảnh vào
1	10000	10000	(24 lần)	10010	(1 lần)
2	01000	01000	(25 lần)		
3	00100	00100	(25 lần)		
4	00010	00010	(20 lần)	10010	(5 lần)
5	00001	00001	(20 lần)	01001	(5 lần)

Hình 7.14. Kết quả tính toán đối với các ảnh mẫu vào bị nhiễu tại 1 điểm ảnh

$$W^T = \left( \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & \\ \begin{matrix} 3.0 & -1.0 & -1.0 & 3.0 & -1.0 & 1 \\ 3.0 & -1.0 & -1.0 & 3.0 & -1.0 & 2 \\ 3.0 & 3.0 & 3.0 & 3.0 & 3.0 & 3 \\ 1.0 & 1.0 & 1.0 & 1.0 & 5.0 & 4 \\ -1.0 & 3.0 & -1.0 & -1.0 & 3.0 & 5 \\ 3.0 & -1.0 & -1.0 & 3.0 & -1.0 & 6 \\ 5.0 & 1.0 & 1.0 & 1.0 & 1.0 & 7 \\ 3.0 & 3.0 & 3.0 & 3.0 & 3.0 & 8 \\ 1.0 & 1.5 & 1.0 & 1.0 & 1.0 & 9 \\ 1.0 & 1.0 & 1.0 & 1.0 & 5.0 & 10 \\ -1.0 & -1.0 & 3.0 & 3.0 & -1.0 & 11 \\ 1.0 & 1.0 & 1.0 & 5.0 & 1.0 & 12 \\ 1.0 & 1.0 & 1.0 & -3.0 & -3.0 & 13 \\ \end{matrix} \right) \quad \left( \begin{matrix} 1.0 & 1.0 & 5.0 & 1.0 & 1.0 & 14 \\ -1.0 & -1.0 & 3.0 & -1.0 & 3.0 & 15 \\ -1.0 & -1.0 & 3.0 & 3.0 & -1.0 & 16 \\ 3.0 & -1.0 & 3.0 & -1.0 & -1.0 & 17 \\ 1.0 & 1.0 & 5.0 & 1.0 & 1.0 & 18 \\ -1.0 & 3.0 & 3.0 & -1.0 & -1.0 & 19 \\ -3.0 & 1.0 & 1.0 & -3.0 & 1.0 & 20 \\ 3.0 & -1.0 & -1.0 & 3.0 & -1.0 & 21 \\ 1.0 & 1.0 & 1.0 & 5.0 & 1.0 & 22 \\ 3.0 & 3.0 & 3.0 & 3.0 & 3.0 & 23 \\ -1.0 & 3.0 & -1.0 & -1.0 & 3.0 & 24 \\ -1.0 & 3.0 & -1.0 & -1.0 & 3.0 & 25 \end{matrix} \right)$$

Hình 7.15. Ma trận trọng số liên kết nơron.

$X_1$	$X_2$	$X_3$	$X_4$
10001	11000	10110	10010
01001	01100	01010	
01101	00110		
00011			

Hình 7.17. Các ảnh ghép nhận được từ các ảnh vào mẫu.

Từ thí nghiệm trên, có thể rút ra một vài kết luận:

- Mạng có thể phục hồi ảnh mẫu nguyên bản ngay cả khi đầu vào bị biến dạng.
  - Mạng có thể tái tạo ảnh mẫu nguyên bản khi biết tín hiệu ra tương ứng.
  - Mạng có thể nhận ra các thành phần trong ảnh ghép.
  - Mạng có thể hiểu khái niệm ảnh âm bản (ảnh neutro).

Ảnh mẫu nguyên bản

$X_1$	$X_2$	$X_3$	$X_4$	$X_5$

Ảnh vào bị biến dạng


Kết quả tính toán

10010	0010	01001
-------	------	-------

Các mẫu ghép tương ứng

$X_{14}$	$X_{14}$	$X_{23}$

Hình 7.16. Các ảnh đầu vào nhiễu và kết quả phân dạng sai.

 Kết quả	 ảnh vào
 10110	 01101

Hình 7.18. Hai ảnh ghép không được nhận dạng đúng.

Như vậy, mạng có thể nhận dạng nhầm với 11 trường hợp đầu vào tương ứng với một ảnh nhiễu của  $X_1$ , 5 ảnh nhiễu của  $X_4$  và 5 ảnh nhiễu của  $X_5$  (hình 7.16). Ký hiệu  $X_0 = X_i$  or  $X_j$ . Ta có 10 ảnh ghép như trên hình 7.17. Sử dụng các vectơ đầu vào  $X_0$ , ta có kết quả rất thú vị là  $X_0$  sẽ kích hoạt đồng 2 neuron ra i, j gần với các ảnh thành phần  $X_i$  và  $X_j$  trừ 2 đầu vào  $X_{1,i}$  và  $X_{2,j}$  (hình 7.18).

#### Một số lưu ý về mạng ABAM

- Mạng có thể sử dụng theo 2 chiều,
- Cấu trúc mạng đơn giản, có thể dùng để phân loại, nhận dạng đối tượng,
- Dữ liệu vào: Nhị phân hoặc luồng cục,
- Cơ chế học có giám sát,
- Có thể phục hồi và nhận biết các đối tượng bị biến dạng.

### E. VAI TRÒ CỦA MẪU HỌC

Trên thực tế, quá trình học chỉ là nhằm tạo ra liên kết giữa các dữ liệu vào và dữ liệu ra trong mẫu học, điều ràng về bản chất chúng không có liên quan gì trực tiếp với nhau. Thực chất, mạng ABAM còn học được thêm hai khái niệm tổng quát, tuy rằng ta không chủ định huấn luyện cho chúng. Đó là :

- Phân tích ảnh ghép thành các thành phần;
- Lấy phần bù kết quả để đối sánh với ảnh âm bản.

Những điều này có thể đạt được khi ta chọn tương đối cẩn thận tập mẫu học. Quả thật khó có thể lựa chọn ngẫu nhiên các mẫu học để tạo ra mạng ABAM dù thông minh, có khả năng phân tích các ảnh vào ghép. Có thể thấy rằng kết quả xử lý của mạng ABAM chỉ là kết quả của việc lặp ráp từ các mẫu học. Một cách trực quan, ta có thể cho rằng các mẫu vào tương tự sẽ cho những kết quả tương tự. Điều này quả thật không đúng. Ví dụ  $X_1$  bị nhiễu 1 điểm ảnh sẽ cho kết quả là ảnh ghép  $X_{1,i}$ , trong khi đó nhiễu tại 1 điểm ảnh ở  $X_2$  không làm thay đổi kết quả đầu ra. Lý do chính ở đây là chúng ta lẫn lộn hai khái niệm "tương tự" và "đối xứng". Hơn nữa, ta nhận thấy 2 mẫu  $X_1$  và  $X_2$  cũng như  $X_4$  và  $X_5$  đối xứng nhau qua tâm. Các ảnh ghép  $X_{1,2}$  và  $X_{4,5}$  tự đối xứng với chính nó qua tâm. Ta nói chúng có chung tâm đối xứng. Tuy vậy,  $X_1$  không chung tâm đối xứng và không thể ghép với ảnh khác để có tính chất đó. Vì lẽ đó, sau khi học xong tập mẫu, chính  $X_1$  dù ngắn cần mạng có tính chất đối xứng đó. Như vậy có thể nói, để có hiệu ứng tương tự lên các mẫu vào bị biến dạng còn tuỳ thuộc vào sự tương tự của các mẫu học nguyên bản và các ảnh ghép của chúng.

## F. UỐC LƯỢNG ĐỘ PHÙ HỢP CỦA MẠNG ABAM

- Số neuron của mạng bằng số ảnh mẫu  $N^{max} = N^{in}$
- Ma trận trọng số có kích thước  $m \times N^{max}$ , ở đây  $m$  là số tín hiệu vào. Nếu mỗi ảnh có kích thước  $r \times r$  điểm ảnh, thì ma trận trọng số có kích thước  $m \times N^{max} = r^2 \times N^{in}$ .

Khác với mạng Hopfield, ở đây không có ràng buộc giữa độ phân giải  $r \times r$  của ảnh với số ảnh mẫu  $N^{in}$ . So sánh với kích thước của ma trận trọng số của mạng Hopfield, mạng ABAM nhỏ hơn cả 7 lần.

### 7.4.3.3. Mạng Kohonen

Cách xử lý thông tin trong các mạng ở trên thường chỉ quan tâm tới giá trị và dấu của các thông tin đầu vào, mà chưa quan tâm khai thác các mối liên hệ có tính chất cấu trúc trong lân cận của các vùng dữ liệu mẫu hay toàn thể không gian mẫu.

Chẳng hạn, với 2 thành phần: 1 tam giác, 1 hình chữ nhật,



ta có thể tạo thành hình ngôi nhà khi chúng được phân bố kế giáp với nhau theo một trật tự nhất định.

Tuovo Kohonen (1989) đã đề xuất một ý tưởng rất đáng chú ý về ánh xạ các đặc trưng topo tự tổ chức (theo nghĩa không cần có mẫu học) nhằm bảo toàn trật tự sắp xếp các mẫu trong không gian biểu diễn nhiều chiều sang một không gian mới các mảng neuron (một hoặc hai chiều). Trong mạng Kohonen, các vectơ tín hiệu vào gần nhau sẽ được ánh xạ sang các neuron trong mạng lân cận nhau.

## A. CẤU TRÚC MẠNG

Mạng Kohonen rất gần gũi với kiểu cấu trúc mạng neuron sinh học cả về cấu tạo lẫn cơ chế học. Mạng Kohonen thuộc vào nhóm mạng một lớp các neuron được phân bố trong mặt phẳng hai chiều theo kiểu lưới vuông, hay lưới lục giác (hình 7.19).

Phân bố này phải thỏa mãn yêu cầu: Mỗi neuron có cùng số neuron trong từng lớp láng giềng. Ý tưởng cơ bản của Kohonen là các dấu vào tương tự nhau sẽ kích hoạt các neuron gần nhau về khoảng không gian. Mỗi quan hệ tương tự (theo khoảng cách) có thể tổng quát hoá cho một lớp tương đối rộng các quan hệ tương tự giữa các tín hiệu đầu vào.

Một cách trực quan, có thể xem thuật giải huấn luyện mạng Kohonen nhằm biến đổi không gian tín hiệu vào sang mạng neuron giống như các thủ tục kiểu như "làm tròn" hay "tạo hình" dữ liệu.

Để đáp ứng yêu cầu các neuron có cùng số neuron lân cận trong mỗi lớp láng giềng, người ta thường dùng các phép cuộn chỉ số để đạt được hiệu ứng cái súng xe. Chẳng hạn, tọa độ  $(x_i, y_i)$  của các neuron thuộc lớp láng giềng thứ k của neuron có tọa độ  $(x, y)$  trong mảng neuron 2 chiều có kích thước pxq được cho trong thủ tục sau:

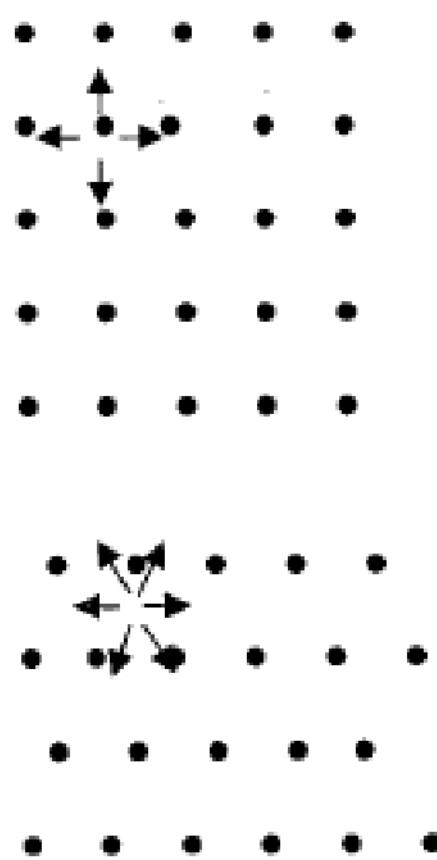
```

for      i:=-k to k do
for      j:=-k to k do
begin    xi,j:=mod(x+i+p-1,p) + 1;
          yi,j:=mod(y+j+q-1,q) + 1;
          if (i=k) or (j=k) then
              neuron (xi,j, yi,j) thuộc vào lớp láng giềng thứ k
          else
              neuron (xi,j, yi,j) thuộc vào lớp láng giềng thứ r
              r<k; r được xác định bởi max(xi,j,yi,j)
end;

```

Trường hợp lớp neuron Kohonen là một dây, cách cuộn tròn mảng neuron tạo thành một đường tròn.

Tất cả các neuron ở lớp kích hoạt có liên kết đầy đủ với lớp vào. Điểm quan trọng nhất trong mạng Kohonen là với một vecto tín hiệu vào, nó chỉ cho phép các phản hồi mang tính chất địa phương, nghĩa là đầu ra của mỗi neuron không được nối với tất cả các neuron khác mà chỉ với một số neuron lân cận. Sự phản hồi mang tính địa phương của những điều chỉnh (nếu có) tạo ra hiệu ứng là các neuron gần nhau về vị trí sẽ có hành vi tương tự khi có những tín hiệu giống nhau được đưa vào.



Hình 7.19: Luổi các neuron trong mặt phẳng hai chiều

## B. HUẤN LUYỆN MẠNG

Quá trình học được sử dụng trong mạng Kohonen dựa trên kỹ thuật cạnh tranh, không cần có tập mẫu học. Khác với trường hợp học có giám sát, các tín hiệu đầu ra có thể không biết được một cách chính xác.

Tại mỗi thời điểm chỉ có một neuron duy nhất  $C$  trong lớp kích hoạt được lựa chọn sau khi đã đưa vào mạng các tín hiệu  $X_i$ . Neuron này được chọn theo một trong hai nguyên tắc sau:

Nguyên tắc 1 Neuron c có tín hiệu ra cực đại

$$\text{out}_c \leftarrow \max_{j=1} (\text{out}_j) = \max \left( \sum_{i=1} \sum (x_{i,j} w_{ji}) \right) \quad (9)$$

Nguyên tắc 2 Vectơ trọng số của neuron c gần với tín hiệu vào nhất

$$\text{err}_c \leftarrow \min_{j=1} (\text{err}_j) = \min \left( \sum (x_{i,j} - w_{ji})^2 \right) \quad (10)$$

Sau khi xác định được neuron  $c$ , các trọng số  $w_{ij}$  được hiệu chỉnh nhằm làm cho đầu ra của nó lớn hơn hoặc gần hơn giá trị trọng số mong muốn. Do vậy, nếu tín hiệu vào  $x_i$  với trọng số  $w_{ij}$  tạo kết quả ra quá lớn thì phải giảm trọng số và ngược lại. Các trọng số của các neuron láng giềng  $j$  cũng phải được hiệu chỉnh giảm, tùy thuộc vào khoảng cách tính từ  $c$ . Ta đưa vào hàm tỷ lệ  $a(.) = a(d_{ij})$ , ở đây  $d_{ij}$  là khoảng cách topo giữa neuron trung tâm  $c$  và neuron  $j$  đang xét. Trên thực tế hàm  $a(.)$  có thể là hằng số, hàm tỷ lệ nghịch hoặc hàm có điểm uốn. Để đảm bảo yêu cầu, do có nhiều mẫu tham gia quá trình huấn luyện, ta đưa vào hệ số  $\eta(t)$ :

$$f = \eta(t) \cdot a(d_{ij}),$$

$$\eta(t) = (a_{\max} - a_{\min}) \frac{t_{\max} - t}{t_{\max} - 1} + a_{\min} \quad (11)$$

ở đây  $t$  là số đối tượng mẫu đã dùng để luyện mạng;

$t_{\max}$  là số mẫu tối đa;

$a_{\max}, a_{\min}$  tương ứng là giá trị cực đại, cực tiểu của hàm  $a(.)$ .

Tùy thuộc vào neuron trung tâm  $c$  được lựa chọn theo nguyên tắc 1 hoặc nguyên tắc 2 ta có cách hiệu chỉnh các trọng số  $w_{ij}$  tương ứng:

$$w_{ij} = w_{ij} + \eta(t) a(d_{ij}) (1 - x_{i,j} w_{ij}) \quad (12)$$

$$\text{hoặc } w_{ij} = w_{ij} + \eta(t) a(d_{ij}) (x_{i,j} - w_{ij}) \quad (13)$$

Sau đó, chuẩn hóa các trọng số sao cho:  $\sqrt{\sum_{i=1}^n w_{ij}^2} = 1$

Theo kinh nghiệm, cần phải tạo ra phân bố ngẫu nhiên các trọng số trong khoảng -0.1 đến 0.1 hoặc -1/m đến 1/m, ở đây m là số trọng số của mạng và chuẩn hóa dữ liệu vào, ra bằng -1 hoặc 1.

Tuy nhiên cũng phải chú ý một điều là việc lựa chọn tiêu chuẩn chuẩn hóa, định cỡ dữ liệu phụ thuộc rất nhiều vào bản chất bài toán.

### C. SỬ DỤNG MẠNG

Giả sử đã huấn luyện mạng để nhận được ma trận trọng số  $W$ . Khi đưa vào mạng một vector  $X$ , toàn bộ ma trận  $W$  lại được cập nhật theo các công thức (12) hoặc (13) tuỳ thuộc vào sử dụng nguyên tắc 1 hay nguyên tắc 2.

Như vậy, mạng Kohonen cho chúng ta biết được sự phân bố và quan hệ tương đối về mặt "địa lý" giữa các mẫu trong không gian biểu diễn.

### D. THÍ NGHIỆM MẠNG

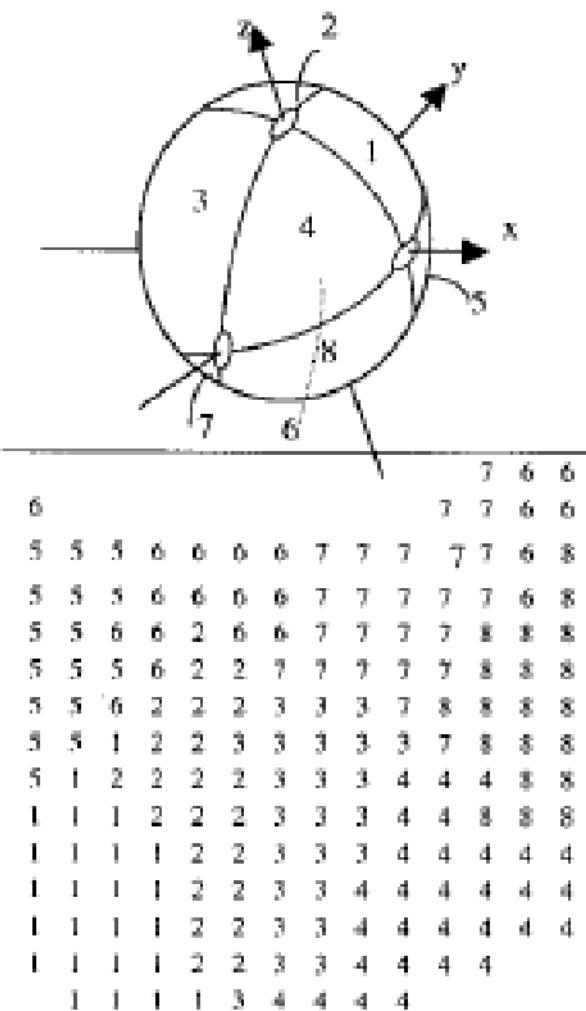
Ánh xạ từ không gian 3 chiều sang không gian 2 chiều.

Bài toán đặt ra là tạo ánh xạ từ một mặt cầu đơn vị 3 chiều với 2000 điểm phân bố ngẫu nhiên trong 8 mảng cầu sang một phẳng các neuron được phân bố trong lưới kích thước 15x15.

Mạng Kohonen được thiết kế có 3 đầu vào, tương ứng với 3 tọa độ và 225 neuron, phân bố thành lưới vuông 15x15. Mỗi neuron vào được nối dây dù với các neuron ra, do vậy tổng cộng có 675 trọng số. Ban đầu neuron trung tâm có 7 lớp láng giềng để đảm bảo rằng tất cả các vùng láng giềng kế giáp nhau. Giả sử, hiệu chỉnh cục đại tại neuron trung tâm  $a(0) = 0.3$  (xem công thức (11)) và tại lớp thứ 7 giá trị này chỉ là 0,5 % giá trị tại neuron trung tâm, do vậy bằng  $0.3 \times 0.005 = 0.0015$ . Giá trị có thể xem là rất nhỏ, do đó  $n(t) = \text{hàng số}$ . Trong quá trình luyện mạng, cứ 400 điểm mẫu được đưa vào để luyện mạng sẽ có một lớp láng giềng ở vòng ngoài bị co lại. Các neuron láng giềng càng xa sẽ càng ít bị hiệu chỉnh hơn. Trong thí nghiệm này ta sử dụng nguyên tắc 2 và công thức hiệu chỉnh (13), các giá trị trọng số ban đầu được lấy ngẫu nhiên trong khoảng [-0,1 - 0,1]. Kết quả huấn luyện mạng với 2000 mẫu được cho trong hình 7.20.

Để dễ dàng thấy rằng tất cả các quan hệ topo giữa các vùng trên mặt cầu được bảo toàn sau khi ánh xạ (hình 7.21).

No	Vùng kết		1	2	3	4	5	6	7	8
1	2 4 5	1	0	1	0	1	1	0	0	0
2	1 3 6	2	1	0	1	0	0	1	0	0
3	2 4 7	3	0	1	0	1	0	0	1	0
4	1 3 8	4	1	0	1	0	0	0	0	1
5	1 6 8	5	1	0	0	0	0	1	0	1
6	2 5 7	6	0	1	0	0	1	0	1	0
7	3 6 8	7	0	0	1	0	0	1	0	1
8	4 5 7	8	0	0	0	1	1	0	1	0



Hình 7.20. Quan hệ topo giữa các vùng.

Hình 7.21. Ánh xạ mặt cầu vào lưới ngron

Điểm thú vị là trên mảng có những vùng trống, nhằm tách rời điểm hội tụ của các vùng 1,2,3,4 ở cực bắc khỏi các vùng 5,6,7,8 ở bán cầu nam.

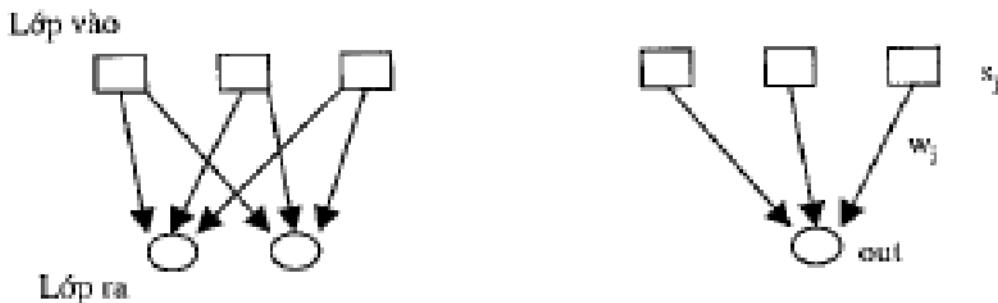
### *Một số lưu ý về mạng Kohonen*

- Mạng không chỉ quan tâm đến nội dung tín hiệu vào mà còn xem xét cấu trúc topo của các mẫu.
  - Mạng có thể biến đổi từ không gian nhiều chiều sang không gian ít chiều hơn.
  - Cơ chế học không có giám sát.
  - Các quan hệ topo được bảo toàn khi ánh xạ.

#### **7.4.3.4. Many Perceptrons**

#### A. CẤU TRÚC MẠNG

Mạng Perception do F Rosenblatt đề xuất năm 1960. Đây là mạng truyền thẳng một lớp có một hoặc nhiều đầu ra. Để đơn giản trong các trình bày ta giả sử mạng có một đầu ra.



$$\text{out} = \begin{cases} 1 & \text{nếu } \sum w_j s_j \geq 0 \\ 0 & \text{nếu ngược lại} \end{cases} \quad (14)$$

## B. HUẤN LUYỆN MẠNG

Mạng học dựa trên nguyên tắc có giám sát với tập mẫu  $(X_s, Y_s)$ .

Ý tưởng cơ bản của quá trình huấn luyện mạng là xác định bộ trọng số W sao cho  $\text{out}_s = \text{Tinh}(X_s, W) = Y_s$ , đối với mọi mẫu học s.

Ban đầu các trọng số được gán ngẫu nhiên trong khoảng [-0,5, 0,5]. Sau đó hiệu chỉnh các trọng số sao cho phù hợp với các mẫu học, làm giảm sai số giữa giá trị quan sát Y, với giá trị tính toán out<sub>s</sub>.

Các bước tiến hành như sau:

- Xác định ngẫu nhiên bộ trọng số trong [-0,5, 0,5].
- Với mỗi mẫu học  $(X_s, Y_s)$ ,  $X_s = (X_{s1}, \dots, X_{sn})$ , thực hiện các bước :

  - Tính giá trị out<sub>s</sub> theo công thức (14)
  - Xác định Err =  $Y_s - \text{out}_s$

Hiệu chỉnh các trọng số  $W_j = W_j + \alpha X_{sj} * Err$ , trong đó  $\alpha$  là hằng số học.

Luật học này có khác một chút so với nguyên bản do Rosenblatt đề nghị. Rosenblatt đã chứng minh được rằng quá trình học của mạng Perceptron sẽ hội tụ tới bộ trọng số W, biểu diễn đúng các mẫu học với điều kiện là các mẫu này biểu thị các điểm rời rạc của một hàm khả tách tuyến tính nào đó (hàm  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  được gọi là khả tách tuyến tính nếu các tập  $\{f^{-1}(x_i)\}$ , với  $x_i$  thuộc miền trị của f, có thể tách được với nhau bởi các siêu phẳng trong không gian  $\mathbb{R}^n$ ).

Năm 1969, Minsky và Papert đã chứng minh một cách chặt chẽ rằng lớp hàm phụ thuộc (giữa đầu ra vào đầu vào) có thể học bởi mạng Perceptron là lớp hàm khả tách tuyến tính.

Có thể thấy rằng sự hội tụ của quá trình học của mạng dựa trên nguyên lý tìm kiếm gradient trong không gian trọng số làm cực tiểu hàm sai số :

$$Err(W) = 1/2(Y_{\text{out}}(w))^2$$

Một điểm đáng chú ý là hệ số học  $\alpha$  phải được chọn không quá lớn để đảm bảo sự hội tụ của quá trình.

## C. SỬ DỤNG MẠNG VÀ KHẢ NĂNG BIỂU DIỄN CỦA MẠNG

Như đã chỉ ra trong phần nhập môn, các neuron riêng lẻ có thể biểu diễn các hàm logic sơ cấp như: AND, OR, NOT. Từ đó có thể hình dung rằng lớp các hàm có thể tính được nhờ mạng neuron Perceptron tương đối phong phú. Sau đây là một số hàm tiêu biểu:

- Hàm "đa số": Hàm này cho giá trị 1 nếu có hơn nửa số đối số có giá trị 1 và 0 trong trường hợp ngược lại. Ta xây dựng mạng Perceptron có n đầu vào nhận giá trị 0, 1 và một neuron ra với các trọng số liên kết  $w_j = 1$  và ngưỡng  $\theta = n/2$ .

- Hàm "thiểu số": Hàm này cho giá trị 1 nếu không quá nửa số đối số có giá trị 1 và 0 trong trường hợp ngược lại. Mạng Perceptron tương ứng được xây dựng với các trọng số liên kết  $w_j = -1$  và ngưỡng  $\theta = -n/2$ .

Tuy vậy, không thể dùng Perceptron để tính hàm  $X \oplus Y$ , bởi lẽ hàm này không khả tách tuyến tính, không tồn tại 1 đường thẳng phân chia 2 vùng  $\{(0,0), (1,1)\}$  và  $\{(0,1), (1,0)\}$ , tương ứng với các kết quả 0 và 1.

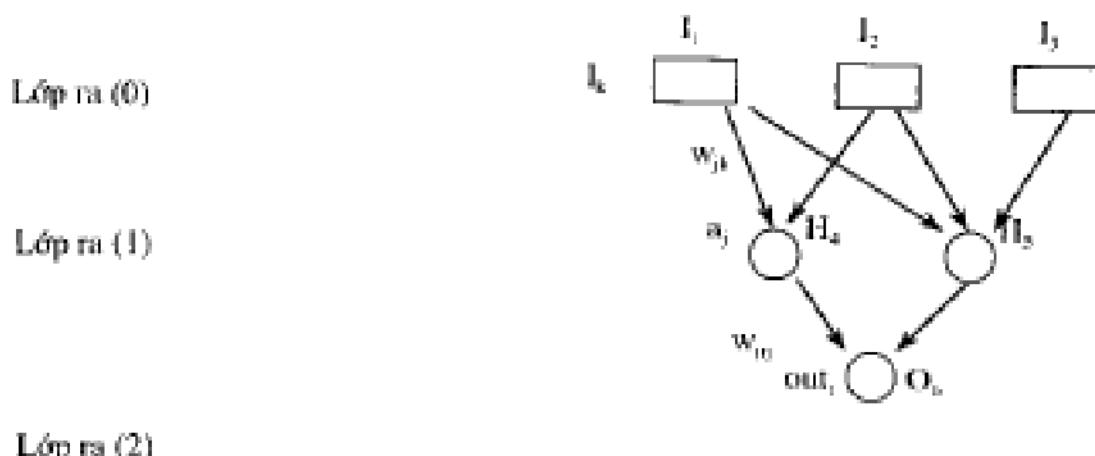
### 7.4 .4 Các mạng neuron nhiều lớp (Multi-layer Neural Network)

#### 7.4.4.1 Mạng neuron nhiều lớp lan truyền ngược sai số (Back-propagation Neural Network)

Rosenblatt và các tác giả khác cũng đã mô tả các mạng truyền thẳng nhiều lớp từ cuối những năm 50, nhưng họ chủ yếu chỉ nghiên cứu sâu về mạng Perceptron một lớp. Sở dĩ như vậy là do không tìm được cách thay đổi trọng số liên kết tại các lớp ẩn. Quả thật, ngay cả khi đã biết được sai số tại các đầu ra, người ta vẫn chưa hình dung được các sai số đó được phân bổ như thế nào tại các neuron ẩn. Trong cuốn sách về mạng Perceptron xuất bản 1969, Minsky và Papert đã chỉ ra rằng khó có thể tổng quát hoá luật học đổi với mạng một lớp sang mạng nhiều lớp. Có 2 lý giải chính cho vấn đề này. Thứ nhất, thuật giải học của mạng nhiều lớp có thể không hiệu quả, hoặc không hội tụ về điểm cực trị tổng thể trong không gian vector trọng số. Một khác, các nghiên cứu trong lý thuyết tính toán đã chỉ ra rằng trong trường hợp tối đa hóa hàm tổng quát từ mẫu học không phải lúc nào cũng

giải quyết được. Các nguyên tắc cơ bản trong luật học đối với mạng nhiều lớp đã được Bryson và Ho đề xuất từ năm 1969, nhưng phải tới giữa năm 1980 vấn đề này mới được quan tâm trở lại bởi công trình nghiên cứu của Rumelhart năm 1986. Một thống kê cho thấy 90% ứng dụng mạng nơron trong công nghệ hoá học sử dụng mô hình này.

### A. KIẾN TRÚC MẠNG



Hình 7.23. Mạng nơron 2 lớp.

Các nơron lớp thứ  $i$  được nối đầy đủ với các nơron lớp thứ  $i+1$ . Trong nhiều ứng dụng thực tế, để đơn giản, người ta thường sử dụng mạng có một lớp ẩn, số nơron trong lớp ẩn được xác định dựa trên kinh nghiệm, hoặc dựa trên các kỹ thuật tìm kiếm khác nhau (xem D, mục 1.2.2).

### B. HUẤN LUYỆN MẠNG

Quá trình huấn luyện mạng được trình bày ở đây là quá trình học có giám sát với tập mẫu  $\{(X_i, Y_i)\}$ . Thủ tục học có thể tóm lược như sau:

Mỗi khi đưa một mẫu  $X_i = (x_1, \dots, x_n)$  vào mạng, ta thực hiện các công việc sau:

- Lan truyền mẫu  $X_i$  qua mạng để có  $out_i = \text{Tinh}(X_i, NN)$ ,

- Tính sai số  $Err_i$  của mạng dựa trên sai lệch  $out_i - Y_i$ ,

- Hiệu chỉnh các trọng số liên kết nơron dẫn tới lớp ra  $W_{ij}$  từ nơron  $j$  tại lớp ẩn cuối cùng tới nơron  $i$  tại lớp ra:

$$w_{ij} = w_{ij} + \alpha \cdot a_j \cdot \delta_i \quad (15)$$

ở đây:  $\alpha$  là hệ số học,

$a_j$  là dấu ra của nơron  $j$ ,

$\delta_i$  là sai số mà nơron  $i$  ở lớp ra phải chịu trách nhiệm, được xác định theo

công thức:

$$\delta_i = \text{err}_i g'(\text{Net}_i) \quad (16)$$

với  $\text{err}_i$  là sai số thành phần thứ  $i$  trong  $\text{Err}_i$ ,  $\text{Net}_i$  là tổng thông tin vào có trọng số của nơron thứ  $i$  ( $\text{Net}_i = \sum w_{ij} a_j$ ) và  $g'(\cdot)$  là đạo hàm của hàm kích hoạt  $g$  được dùng trong các nơron.

- Hiệu chỉnh các trọng số liên kết nơron  $W_{jk}$  dẫn tới tất cả lớp ẩn từ nơron thứ  $k$  sang nơron  $j$  (các lớp ẩn được xét từ dưới lên):

Tính tổng sai số tại nơron  $j$  phải chịu trách nhiệm

$$\delta_j = g'(\text{Net}_j) \sum w_{ij} \delta_i \quad (17)$$

Hiệu chỉnh trọng số  $w_{jk}$

$$w_{jk} = w_{jk} + \alpha a_k \delta_j \quad (18)$$

(trường hợp xét liên kết từ nơron vào thứ  $k$  sang nơron  $j$  trên lớp ẩn thứ nhất, ta có  $a_k = I_k$ , chính là tín hiệu vào).

Chú ý :

a) Trường hợp xét hàm kích hoạt tại các nơron

$$g(x) = \frac{1}{1 + e^{-x}}$$

ta có hệ thức

$$g'(x) = g(x)(1 - g(x)).$$

b) Từ các công thức (15), (18) ta có thể viết lại:

$$w_{ij} = w_{ij} + \Delta w_{ij}, \quad w_{jk} = w_{jk} + \Delta w_{jk},$$

$$\text{với } \Delta w_{ij} = \alpha a_j \delta_i \text{ và } \Delta w_{jk} = \alpha a_k \delta_j$$

Trong các ứng dụng thực tế, người ta thường hiệu chỉnh  $\Delta w_{ij}$  theo nguyên tắc cố chủ ý đến thao tác trước đó. Do vậy:

$$\Delta w_{ij}^{(new)} = \alpha a_j \delta_i + \beta \Delta w_{ij}^{(old)}, \text{ ở đây } \beta \text{ là hệ số quán tính.}$$

Quá trình huấn luyện mạng cần chú ý tới các yếu tố sau:

- Các trọng số ban đầu  $w_{ij}$  được gán các giá trị ngẫu nhiên, nhỏ,
- Lựa chọn các hệ số học  $\alpha$  và hệ số quán tính  $\beta$  sao cho  $\alpha + \beta \approx 1$ , với  $\beta$  không lớn hơn  $\alpha$  quá nhiều,
- Các tín hiệu vào, ta nên được định cũ chỉ nằm trong khoảng [0,1]. Các nghiên cứu thực nghiệm chỉ ra rằng nên ở trong khoảng [0.2, 0.8].

### C. SỬ DỤNG MẠNG

Giả sử đã huấn luyện mạng như trên hình 7.23 với tập mẫu  $(X_i, Y_i)$  để được ma trận trọng số W. Quá trình lan truyền trong mạng một vectơ tín hiệu vào  $X = (x_1, x_2, x_3)$  được cho bởi:

$$\begin{aligned} \text{out} &= g(w_{01} a_1 + w_{02} a_2) = g(w_{01} g(w_{41} x_1 + w_{42} x_2 + w_{43} x_3) + \\ &\quad + w_{03} g(w_{51} x_1 + w_{52} x_2 + w_{53} x_3)) \\ &= F(X, W) \end{aligned}$$

#### *Khả năng tính toán của mạng nhiều lớp*

- Với một lớp ẩn, mạng có thể tính toán xấp xỉ một hàm liên tục bất kỳ đối với các biến tương ứng là các tín hiệu đầu vào.
- Với hai lớp ẩn, mạng có thể tính toán xấp xỉ một hàm bất kỳ. Tuy vậy, số neuron trong các lớp ẩn có thể tăng theo hàm mũ đối với số đầu vào và cho đến nay vẫn chưa có những cơ sở lý luận đầy đủ để khảo sát họ các hàm có thể xấp xỉ nhờ các mạng nhiều lớp.

### D. NGHIÊN CỨU SỰ HỘI TỤ VÀ ĐỘ PHỨC TẠP CỦA QUÁ TRÌNH HUẤN LUYỆN MẠNG

Phương pháp hiệu chỉnh trọng số liên kết neuron (15)(18) dựa trên nguyên tắc lan truyền ngược sai số có thể lý giải dựa trên nguyên lý tìm kiếm gradient trong không gian các tham số W sao cho cực tiểu hàm sai số tổng cộng:

$$E(w) = \frac{1}{2} \sum (Y_i - \text{out}_i)^2$$

Ở đây,  $Y_i$  là giá trị thực nghiệm quan sát được tại neuron i ở lớp ra,  $\text{out}_i$  là giá trị tính toán của mạng tại neuron thứ i ở lớp ra đối với mẫu  $X_i$ .

Khai triển E theo các trọng số thành phần, ta có:

$$E(w) = \frac{1}{2} \sum_i \left( Y_i - g\left(\sum_j w_j a_j\right) \right)^2 = \frac{1}{2} \sum_i Y_i - g\left(\sum_j w_j g\left(\sum_k w_k a_k\right)\right)$$

Lấy đạo hàm riêng của E theo các  $w_j$ :

$$\frac{\partial E}{\partial w_j} = -a_j(Y_i - \text{out}_i)g'(\text{Net}_i) = -a_j \delta_i$$

$$\frac{\partial E}{\partial w_j} = -a_i \delta_j$$

Việc hiệu chỉnh vector trọng số  $W = (w_{ij})$  sao cho  $E(W) \rightarrow \min$  dẫn tới việc xác định vector giá số  $\Delta W = (\Delta w_{ij})$  ngược hướng với vector gradient ( $\partial E / \partial w_{ij}$ ). Nói cách khác,

$$\Delta w_{ij} = -\alpha(-\delta_j; a_j) = \delta_j a_j$$

$$\Delta w_k = -\alpha(-\delta_j; a_k) = \delta_j a_k$$

Công thức này phù hợp với các công thức (15) (18) tương ứng.

Độ phức tạp thời gian của mạng nhiều lớp chủ yếu phụ thuộc vào thời gian huấn luyện mạng với một tập mẫu nào đó. Giả sử có m mẫu vào và  $|W|$  trọng số. Mỗi lần đưa tất cả các mẫu đi qua mạng (gọi là một vòng lặp (epoch)) phải tốn  $O(m|W|)$  thao tác neuron. Trong trường hợp xấu nhất, số vòng lặp sẽ phụ thuộc hàm mũ vào số đầu vào n. Do vậy, chi phí thời gian sẽ là  $O(k^m|W|)$ .

Hơn nữa quá trình học không phải lúc nào cũng hội tụ và có thể dẫn tới các cực tiểu địa phương của hàm E. Khi dùng mạng neuron nhiều lớp để biểu diễn tất cả các hàm logic có n đầu vào, ta phải dùng cỡ  $2^n/n$  nút ẩn, mạng này có khoảng  $O(2^n)$  trọng số, do vậy phải tiêu tốn  $O(2^n)$  bit để biểu diễn các hàm logic.

## E. MỘT SỐ VẤN ĐỀ VỀ MẠNG NORON NHIỀU LỚP.

- Mạng neuron nhiều lớp truyền thẳng là cách biểu diễn các đối tượng dựa trên các giá trị của các thuộc tính của chúng tương đối hiệu quả, tuy rằng chúng chưa vết cạn hết mọi khía cạnh khác nhau về đối tượng đó. Cách tiếp cận mạng loại này tỏ ra khả hiệu quả khi các quan sát (tín hiệu vào) có miền giá trị liên tục. Do vậy, có thể xem là tốt hơn so với những cách tiếp cận truyền thống dựa trên logic mệnh đề và cây quyết định.

- Khả năng tổng quát hóa: mạng loại này có thể đưa ra những kết quả mang tính tổng quát hóa, tuy rằng kiểu phụ thuộc giữa đầu ra và đầu vào không quá rầm.

- Khả năng dung thứ lỗi: mạng được luyện mẫu theo nguyên tắc hồi qui tuyển tính nên có thể chấp nhận sai số trong tập dữ liệu vào. Tuy vậy, mạng không thể đưa ra được những kết quả tính toán không chắc chắn, không chính xác kiểu như mạng Bayes.

- Mạng được sử dụng như một hộp đen, biểu thị quan hệ nào đó giữa tín hiệu ra và tín hiệu vào, mà không cần chỉ rõ dạng giải tích tường minh của mỗi quan hệ đó. Tuy vậy, điểm bất lợi của cách tiếp cận mạng chính là ở chỗ không thể lý giải các kết quả ra một cách rõ ràng như đối với suy diễn logic hay cây quyết định.

#### 7.4.4.2. Mạng nơron nhiều lớp ngược hướng (Counter-propagation Neural Network)

Trong cách tiếp cận lan truyền ngược hướng, các tín hiệu mẫu ra (chứ không phải là sái số) được lan truyền ngược trên mạng nhằm hiệu chỉnh các trọng số. Mỗi khi có các tín hiệu vào, đầu ra của mạng ngược hướng được xác định dựa trên trọng số liên kết giữa nơron trung tâm trong lớp ẩn Kohonen và các nơron ra.

### A. KIẾN TRÚC MẠNG

Mạng ngược hướng có 2 lớp: Lớp Kohonen và lớp ra. Các nơron của lớp vào được nối đầy đủ với các nơron ở lớp Kohonen. Tại mỗi bước, chỉ có một số nơron lâng giêng với nơron trung tâm được chọn được nối với các nơron ra và chỉ các trọng số liên kết tương ứng được hiệu chỉnh.

Điểm khác biệt ở đây là kết quả đầu ra không được lưu trong mạng dưới dạng tập tất cả các trọng số tại mỗi nơron ra, mà chỉ là một phần trong số đó. Từ đó có thể thấy, số nơron ở lớp Kohonen bằng số kết quả đầu ra cần lưu trữ và số nơron ra bằng số biến thành phần trong mỗi kết quả ra. Chẳng hạn, cần có 1000 kết quả ra khác nhau, mỗi kết quả được biểu diễn bởi 4 thành phần, ta phải xây dựng mạng Kohonen có 1000 nơron và một lớp ra có 4 nơron tương ứng với 4 thành phần. Số nơron của lớp vào bằng số biến vào. Thông thường, các tín hiệu  $X = (x_1, \dots, x_m)$  phải được chuẩn hóa sao cho  $\|x\|=1$ . Để làm như vậy, ta đưa thêm thành phần mới  $x_{m+1}$  sao cho:

$$x_{m+1} = \sqrt{1 - \|x\|^2}$$

### B. HUẤN LUYỆN MẠNG

Giả sử có tập mẫu  $\{(X_i, Y_i)\}$ .

Quá trình học được tiến hành như sau:

- Với mỗi vectơ mẫu  $X_i$ , chọn nơron trung tâm c theo nguyên tắc:

hoặc có tín hiệu ra lớn nhất  $out_c = \max(out_j) = \max(\sum w_{ij} x_{i,j})$

hoặc có vectơ trọng số gần với tín hiệu vào nhất

$$err_c \leftarrow \min(err_j) = \min(\sum_i (x_{i,j} - w_{ij})^2)$$

- Hiệu chỉnh các trọng số liên kết dẫn tới lớp Kohonen theo một trong 2 công thức:

$$w_{ji} = w_{ji} + \eta(t) a(d_{ij})(1 - x_{i,j} w_{ji})$$

hoặc  $w_j = w_j + \eta(t) u(d_{ij})(x_i - w_j)$  (xem thêm phần 7.4.3.3)

Sau đó các trọng số ( $w_j$ ) được chuẩn hóa như sau:

$$w_j = \frac{w_j}{\|w_j\|} \quad \text{với} \quad \|w_j\| = \sqrt{\sum_i w_j^2}$$

- Hiệu chỉnh các trọng số liên kết giữa lớp ra và lớp Kohonen

$$w_j = w_j + \eta(t) u(d_{kj})(y_k - w_j)$$

(nơron j thuộc lớp Kohonen và nơron k thuộc lớp ra)

Các trọng số  $w_j$  không cần phải chuẩn hóa.

## C. SỬ DỤNG MẠNG.

Mạng có thể nhận các giá trị tín hiệu vào, ra là số thực. Ngoài ra, mạng làm việc giống như bảng tra. Người ta còn dùng mạng ngược hướng để biểu thị các quan hệ phụ thuộc giữa các biến dựa trên các trọng số liên kết ở lớp ra.

### 7.4.5. Ứng dụng mạng nơron lan truyền ngược hướng cho nhận dạng ký tự

#### 7.4.5.1 Mô tả

Một trong những thủ tục quan trọng và tiêu tốn nhiều thời gian của các phương pháp truyền thống là làm mảnh ký tự. Thủ tục này làm giảm độ rộng các nét của ký tự xuống còn một điểm nhằm phát hiện bộ khung xương của ký tự. Thủ tục này tạo điều kiện cho việc tìm kiếm các dấu hiệu đặc trưng của ký tự, song nó cũng làm cho một số ký tự trở nên giống nhau hơn. Do đó gây khó khăn cho các thủ tục ra quyết định sau này. Thủ tục này sẽ không làm việc tốt khi chất lượng quét của scanner tồi hoặc bản thân ký tự được lấy từ văn bản mờ hoặc có nét đứt. Do đó nó cần được các thủ tục tiền xử lý như làm trơn đường biên, khử nhiễu trợ giúp.

Các phương pháp ra quyết định trong phương pháp nhận dạng truyền thống được cài đặt tĩnh trong chương trình. Các phương pháp này dựa vào việc nghiên cứu các đặc trưng của các ký tự như số giao điểm hoặc cấu trúc các ký tự như số điểm kết thúc, số chu trình, số thành phần liên thông, v.v. Nhìn chung các phương pháp truyền thống đều gặp khó khăn khi chất lượng ảnh sau khi số hoá tương đối kém.

#### 7.4.5.2 Nhận dạng bằng mạng nơron lan truyền ngược hướng

Mạng nơron nói chung và mạng lan truyền ngược hướng nói riêng là sự mô phỏng sinh

học bằng máy tính bộ não người. Nó có khả năng học từ kinh nghiệm hay từ một tập mẫu. Quá trình học của mạng lan truyền ngược hướng là quá trình học có giám sát với một mẫu  $(X_s, Y_s)$  cho trước, ở đây  $X_s$  là vector vào (ma trận điểm ảnh của một ký tự) và  $Y_s$  là giá trị ASCII của ký tự đó. Thực chất việc học của mạng là biến đổi và ánh xạ topô các ký tự xuống một mặt phẳng hai chiều tương ứng với các neuron. Sau khi học xong, mạng lan truyền ngược hướng hoạt động như một bảng tra với đầu vào là các vector điểm ảnh của các ký tự. Quá trình học của mạng khá dài song quá trình nhận dạng rất nhanh. Một trong những ưu điểm chính của của mạng là không đòi hỏi các quá trình tiền xử lý như làm mờ, làm tròn đường biên hay khử nhiễu. Ưu điểm này có được nhờ khả năng xử lý các vector vào có một phần bị hỏng (Partly corrupted). Ngoài ra mạng có thể có khả năng nhận dạng cả các font chữ khác nhau. Khi đó chỉ cần cung cấp một tập mẫu của font chữ đó cho quá trình học.

Quá trình học của mạng lan truyền ngược hướng là quá trình học có giám sát. Do đó nó cần có một tập mẫu chuẩn  $(X_s, Y_s)$ . Trong quá trình học vector vào  $X_s$  đi vào mạng Kohonen, ở đây diễn ra quá trình học cạnh tranh. Vector lời giải  $Y_s$  đi vào lớp ra theo hướng ngược lại làm thay đổi giá trị các trọng số của các neuron trên lớp ra. Giá thiết chúng ta có mạng lan truyền ngược hướng gồm  $N$  neuron trên lớp Kohonen và  $M$  neuron trên lớp ra.  $W_{ij}$  là trọng số thứ  $i$  của neuron thứ  $j$  trên lớp Kohonen.  $C_j$  là trọng số của neuron thứ  $i$  trên lớp ra nối với neuron thứ  $j$  trên lớp Kohonen. Quá trình học của mạng lan truyền ngược hướng bao gồm các bước sau đây:

- Một đối tượng gồm cặp vector  $(X_s, Y_s)$  được lấy ra từ tập mẫu.
- Vector  $X_s$  đi vào lớp Kohonen.
- Neuron trung tâm được chọn theo phương trình.
- Tất cả các trọng số của neuron trên lớp Kohonen được điều chỉnh theo phương trình .
- Các trọng số của neuron trên lớp ra được điều chỉnh theo phương trình:

$$C_j^{(new)} = C_j^{(old)} + \eta(t).a(d_s - d_j).(y_s - C_j^{(old)})$$

- Quá trình lặp lại đối với đối tượng tiếp theo.

Mỗi lần tất cả các đối tượng mẫu đã đi qua mạng được gọi là một lượt. Thông thường cần phải thực hiện từ vài trăm đến hàng nghìn lượt để mạng ổn định. Khi chọn được các hàng số đặc trưng của quá trình học  $a_{max}, a_{min}$  thích hợp, quá trình học của mạng luôn hội tụ.

#### 7.4.5.3 Cài đặt mạng lan truyền ngược hướng cho nhận dạng ký tự

Một mạng tổng quát cho việc nhận dạng ký tự được cài đặt trên ngôn ngữ C như một lớp (Class) có tên gọi là Netcount. Các tham số của mạng là các biến thành viên còn các chức năng của mạng được thiết kế cho các hàm thành viên. Mạng chỉ có một neuron trên lớp ra và có kiểu là ký tự.

```
Class Netcount
{protected:
int dai, rong, N;
float amax, amin, *W[1600];
char C[1600];
public:
Netcount(int, int);
Void hoc(char*, long T);
Char doan (char*);}
```

Các trọng số  $W_{ij}$  được cấp phát động cho bằng các con trỏ W. Khoảng cách giữa neuron có toạ độ  $k_j, l_j$  với neuron trung tâm  $k_i, l_i$  được tính theo công thức:

$$D = \max(\min(|k_j - k_i|, |k_j - k_i + dai|), \min(|l_j - l_i|, |l_j - l_i + rong|))$$

Hàm phụ thuộc topo  $a(d_{ij} - d)$  được dùng trong chương trình là hàm tam giác:

$$a(d_{ij}) = \begin{cases} 0 & \text{nếu } D_{ij} = D_{\max} \\ \frac{D_{\max} - D}{D_{\max}} & \text{nếu } D_{ij} < D_{\max} \end{cases}$$

Trong đó  $D_{\max}$  là khoảng cách từ lân cận xa nhất có thể có của mạng:

$$D_{\max} = \max(dai/2, rong/2) + 1;$$

Nhìn chung để cài đặt mạng neuron cho nhận dạng ký tự cần:

- **Tổ chức số liệu**

Tập mẫu được tổ chức trong một tệp số liệu. Các cặp  $(X_i, Y_i)$  được viết lần lượt theo từng dòng. Một điều đặt ra là phải số thực hóa các vectơ vào khoảng  $[0, 1]$  vì các trọng số của mạng là các số thực. Các nghiên cứu cho thấy việc số thực hóa làm cho mạng có khả năng đoán nhận các ký tự từ các ảnh số sai lệch lớn hơn. Hơn nữa, với việc tổ chức số thực hóa, chúng ta có thể làm giảm kích thước của vectơ vào và có khả năng làm việc đối với các ký tự có kích thước ảnh khác nhau. Thực tế chỉ ra các phương pháp số thực hóa khác nhau sẽ ảnh hưởng đến khả năng cực đại mà mạng có thể đoán nhận từ các ảnh sai lệch.

- **Cấu trúc và các tham số học**

Mục đích của việc xây dựng mạng là xác định số lượng neuron trên lớp Kohonen.

Với số lượng neuron trên lớp Kohonen càng lớn, khả năng đoán nhận các ký tự từ các ảnh có tỷ lệ sai lớn hơn. Tuy nhiên, khi tăng số lượng các neuron, khả năng nhận biết sẽ tiến sát tới khả năng cực đại mà mạng có thể đoán nhận với các ảnh sai (phụ thuộc vào phương pháp số thực hoà). Chúng ta cũng dễ nhận thấy thời gian học và thời gian đoán nhận, cũng như bộ nhớ của máy tính tăng tỷ lệ, có thể hàm mũ với số lượng neuron trên lớp Kohonen. Thực tế, việc xây dựng mạng là công việc thử nghiệm, dần dần tăng kích thước mạng cho đến khi đạt được các chỉ tiêu mong muốn.

Các giá trị trọng số ban đầu thực sự không quan trọng với quá trình học nhưng chúng phải được gán bằng các số ngẫu nhiên từ 0 đến 1.

Các tham số học  $a_{max}$ ,  $a_{min}$  ảnh hưởng không nhiều đến quá trình học nếu chúng thỏa mãn các điều kiện sau:  $a_{max} \in [0.3, 1]$ ;  $a_{min} \in [0, 0.1]$ .

Với giá trị  $a_{max} = 0.5$  và  $a_{min} = 0.01$  có thể là giá trị tốt cho quá trình học.

#### 7.4.5.4. Nhận dạng 37 ký tự sử dụng mạng lan truyền ngược hướng

Một tập mẫu 37 ký tự từ A → Z, 0 → 9 và ký tự '=' được tách ra từ tập ảnh quét bởi scanner có kích thước 32 x 32 điểm ảnh.

Bài thử nghiệm được tiến hành là:

- Không số thực hoà.
- Lọc các điểm ảnh bằng mặt nạ  $3 \times 3$ .

Phân mảnh ảnh thành 64 mảnh. Mỗi vùng có giá trị thực bằng tổng điểm số điểm ảnh đen (giá trị 1) chia cho 16

Bảng 1 thống kê khả năng nhận đúng ký tự từ các ảnh có tỷ lệ sai cực đại của mạng 20 x 20 neuron sau 3000 lượt học.

Bảng 2 thống kê sự phụ thuộc của khả năng nhận dạng các ảnh sai vào kích thước với việc số thực hoà là phân 64 mảnh.

Bảng 1

Không số thực hoà	Mặt nạ $3 \times 3$	Phân 64 mảnh
3%	15%	19%

Bảng 2

10 x 10	20 x 20	30 x 30	40 x 40
3%	19%	24%	25%

Hình 7.24 cho thấy topo của các ký tự của mạng 30 x 30 với phương pháp phân mảnh sau 3000 lần học.

Với việc phân bố của các ký hiệu ở hình bên ta dễ nhận thấy mạng đã phát hiện một cách khác quan các đặc trưng topo của các ký tự thường được dùng trong các phương pháp

nhận dạng cấu trúc truyền thống. Các ký tự có cấu trúc topo tương đối giống nhau được sắp xếp đặt gần nhau, như các ký tự có điểm kết thúc như nhau ('Z', '2'), ('S', '8'); các ký tự có một chu trình ('O', 'U', 'Q', 'R', 'Y', 'D'); Các ký tự có hai chu trình ('B', '8'). Một đặc điểm rất quan trọng là mạng đã phát hiện ra các ký tự có "tiềm năng" giống nhau như các ký tự ('H', 'E', 'W') rất dễ trở thành có hai chu trình khi ảnh bị sai lớn. Ký tự 'A' khi bị mất gốc cuối bên trái có thể trở thành số '4'; Ký tự 'U' rất dễ trở thành có chu trình. Ngoài ra mạng đã phát hiện các ký tự có một hay nhiều phần giống nhau khó có khả năng mô tả trong các chương trình nhận dạng truyền thống như mật độ các điểm đen như ('M', 'X', 'A'), hay nét cong của đường biên ký tự 'G' và 'O'.

### KẾT LUẬN

Từ ví dụ nhận dạng 37 ký tự cho thấy việc nhận dạng ký tự bằng mạng lan truyền ngược hướng có hiệu quả, đơn giản và nhanh hơn các phương pháp truyền thống. Nó có khả năng nhận dạng được các ký tự từ các ảnh có chất lượng tối với số điểm ảnh sai 25%. Lợi thế chính của mạng loại này xuất phát từ khả năng học các đặc trưng topo của các mẫu. Tuy nhiên với một tập mẫu khá lớn, việc sử dụng tài nguyên của máy tính sẽ rất lớn.



Hình 7.24.

### 7.5 CÁC HỆ NHẬN DẠNG CHỦ

Phần trên đã trình bày tổng quan về nhận dạng: các mô hình, phương pháp, v.v. Dưới đây, chúng ta sẽ đề cập tới một số bài toán nhận dạng tiêu biểu mà ứng dụng của nó khá

phổ biến: các hệ nhận dạng chữ OCR (Optical Character Recognizer).

Bài toán nhận dạng chữ là một bài toán lớn và được quan tâm từ lâu. Bài toán này được phân thành 2 nhánh lớn:

- Nhận dạng chữ in để phục vụ cho công tác đọc tự động văn bản, đẩy nhanh việc nhập thông tin vào máy.

- Nhận dạng chữ viết tay với các font chữ khác nhau, phục vụ cho các ứng dụng đọc và xử lý hoá đơn, văn bản, v.v.

Việc nhận dạng chữ tổng quát bao gồm chữ in, viết tay thường và hoa là một bài toán lớn liên quan đến nhiều vấn đề khác nhau trong hệ thống nhận dạng. Thuật toán cho nhận dạng loại này sẽ được đề cập trong phần cuối chương.

Trên thế giới hiện nay có nhiều chương trình nhận dạng chữ viết (chữ in và viết tay) bằng các thứ tiếng Anh, Nga, v.v. như các hệ OMNIPAGE, READ-WRITE, WORD-SCAN,... . Ở Việt Nam cũng có một số hệ như WORC của công ty 3C, VIET-IN của công ty SEATIC, VNDOCR của Viện Công nghệ thông tin, Image Scan của Trung Tâm Tự động hoá thiết kế. Dù là thiết kế để nhận dạng chữ nào, các hệ thống nhận dạng cũng hoạt động dựa trên một cơ chế tương tự như trình bày dưới đây.

### 7.5.1. Sơ đồ tổng quát của một hệ nhận dạng chữ

Về cơ chế, một hệ thống nhận dạng chữ thường gồm các khối chính, phù hợp với các giai đoạn xử lý sau:

- Khối xử lý sơ bộ;
- Khối tách chữ;
- Khối nhận dạng chữ;
- Khối phục hồi chữ (hoàn thiện về nội dung và hình thức, chừa lõi, v.v.)

Hình 7.25 cho ta cái nhìn tổng quát về hệ thống nhận dạng chữ viết.

### 7.5.2. Giai đoạn xử lý sơ bộ

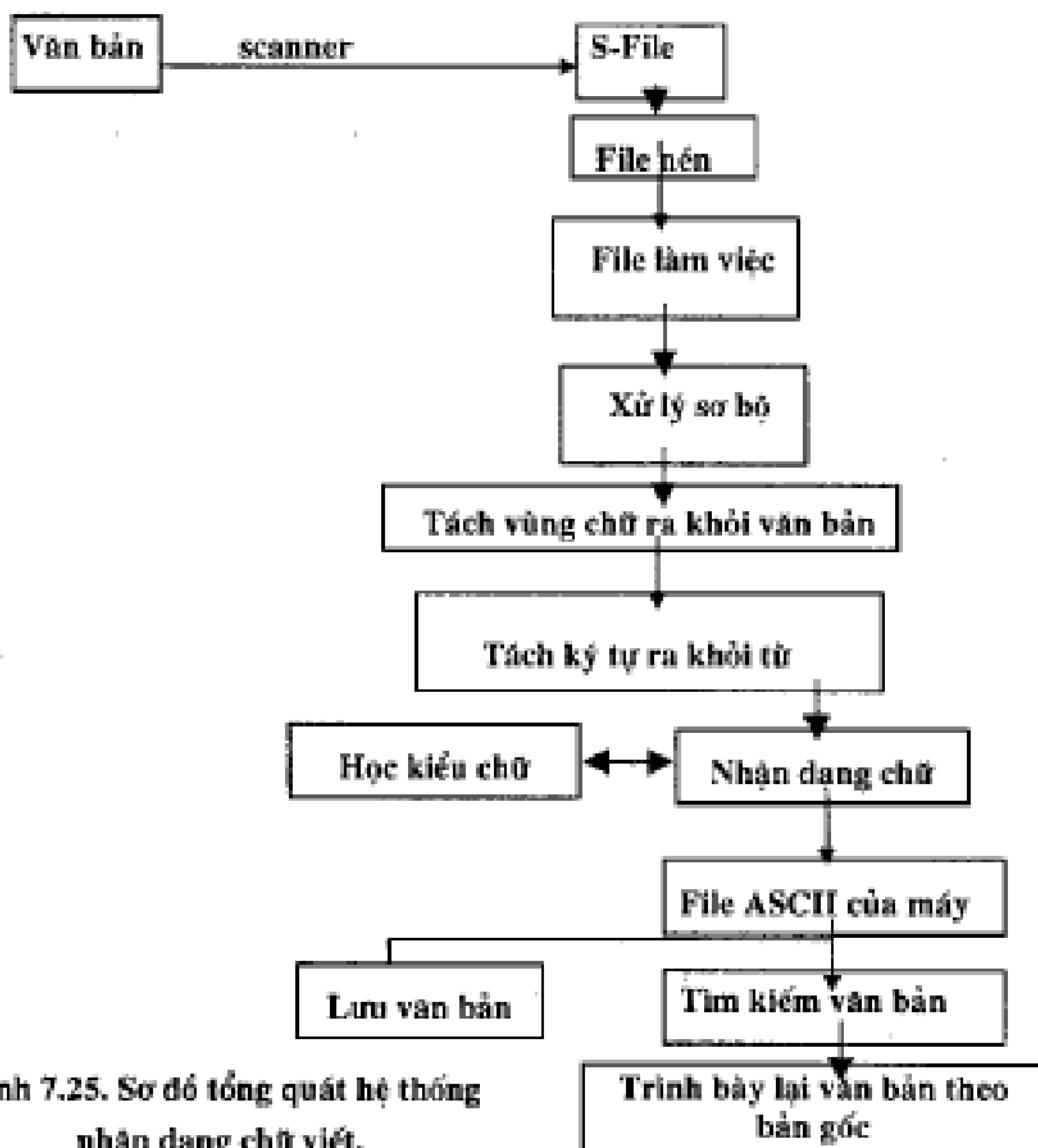
Đây là giai đoạn quan trọng ảnh hưởng đến kết quả nhận dạng. Tuỳ thuộc vào chất lượng ảnh được quét mà ta tiến hành các thủ tục xử lý khác nhau. Vì quá trình xử lý sơ bộ có thể làm chậm tốc độ xử lý của hệ thống nên nếu ảnh quét vào là tốt, ta có thể bỏ qua bước này. Xử lý sơ bộ thường gồm các bước sau:

- Khử nhiễu,

- Làm tròn biên chữ,
- Làm dày chữ,
- Làm mảnh chữ,
- Xoay văn bản đi một góc.

### *Khuỷu nhiễu*

Nhiều là điều không thể tránh khỏi trong các hệ thống xử lý tín hiệu. Như đã nêu trong chương 4 (4.1.2.3) có 2 loại nhiễu: nhiễu hệ thống và nhiễu ngẫu nhiên. Dù loại nhiễu nào cũng phải loại bỏ hoặc làm giảm tối đa ảnh hưởng của nó. Tuỳ theo từng loại nhiễu mà áp dụng các kỹ thuật lọc (xem mục 4.2.2 và 4.2.3 chương 4).



Hình 7.25. Sơ đồ tổng quát hệ thống nhận dạng chữ viết.

### Làm tròn biến chữ

Đôi khi chất lượng quét quá tồi, các đường biến chữ không còn dáng vẻ tròn tru như ban đầu mà hình thành các đường răng cưa. Trong trường hợp này phải áp dụng một số kỹ thuật để làm tròn biến chữ, lấp đầy các chỗ trống, xoá đi các điểm giả tạo trên biến. Hai kỹ thuật hay được sử dụng là kỹ thuật Unger và kỹ thuật Dineen.

**Kỹ thuật Dineen** dùng một mảng  $n \times n$  di chuyển trên tất cả các vị trí của ảnh mẫu. Một mẫu mới được tạo ra, trong đó mỗi phần tử tại tâm của nó sẽ được tính lại theo các phần tử lân cận. Nếu tổng số các phần tử trong cửa sổ lớn hơn ngưỡng 0 nào đó thì trong mẫu mới, vị trí tương ứng sẽ đen; ngược lại là trắng. Kích thước cửa sổ thường chọn là  $3 \times 3$  hay  $4 \times 4$ . Thực tế kỹ thuật Dineen là dùng trung bình trọng số.

**Kỹ thuật Unger** sử dụng một tập luật để lấp các chỗ trống trên ảnh:

Một điểm trên mẫu mới là đen nếu và chỉ nếu thoả 1 trong 2 điều kiện sau:

- P là điểm đen.
- Có ít nhất 3 trong 4 láng giềng:  $P_1, P_2, P_4, P_8$  là đen (hình 6.9.a chương 6).

Để loại bỏ các điểm sáng có lặp trên biến sau khi đã lấp đầy chỗ trống, Unger lại dùng một bộ luật áp dụng cho các phần tử trong phạm vi cửa sổ  $n \times n$ . Tập luật này được mô tả như sau: một điểm trên mẫu mới là đen nếu và chỉ nếu giá trị của nó bằng 1 và thoả một trong hai điều kiện:

- Có ít nhất 1 trong 3 phần tử:  $P_1, P_2, P_4$  là đen.
- Có ít nhất 1 trong 3 phần tử:  $P_6, P_7, P_8$  là đen.

hay:

- Có ít nhất 1 trong 3 phần tử:  $P_1, P_3, P_4$  là đen.
- Có ít nhất 1 trong 3 phần tử:  $P_2, P_6, P_8$  là đen.

Thực tế chúng ta rằng hai kỹ thuật trên đây áp dụng khá tốt cho văn bản tiếng Anh. Tuy nhiên với tiếng Việt do kích thước nhỏ nên cần phải áp dụng một số cải tiến khác.

### Lấp đầy chữ

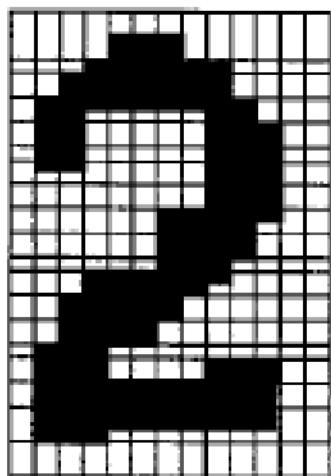
Thủ tục lấp đầy chữ áp dụng cho các ký tự bị đứt nét một cách ngẫu nhiên. Thiếu ví dụ như chữ "m" dễ bị coi thành 2 chữ "r" và "n".

Gọi  $P(i,j)$  là điểm tại toạ độ  $(i,j)$  trong ảnh.  $P(i,j)$  coi là biến nếu thoả 2 điều kiện:

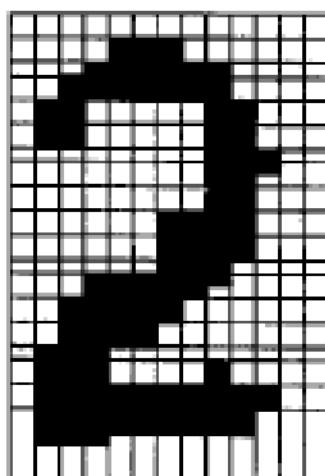
- $P(i,j) = 1 \& (\exists l \in V_i, j) \& (\exists 0 \in V_j)$

với  $V_{ij} = L_{ij}/P_{ij}$  mà  $L_{ij} = \{P_{u,v} : |u-i| < \epsilon, |v-j| < \epsilon\}$

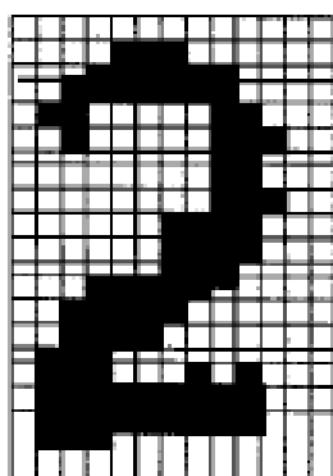
Có nghĩa là  $P(i,j)$  là biên nếu nó là điểm sáng và tồn tại ít nhất 2 trong 8 láng giềng có mức sáng khác nhau.



a) Ảnh gốc



b) Kỹ thuật Dineen  
(với  $n=3$ , ngưỡng=4)



c) Kỹ thuật Unger

Hình 7.26. Làm tròn biên chữ.

#### *Làm mảnh chữ*

Thực chất là kỹ thuật tạo khung ảnh (skeleton) đã nói trong mục 6.4.1.

#### *Xoay văn bản đi một góc:*

Do văn bản lúc đưa vào máy có thể lệch đi một góc  $\alpha$  nào đó. Trường hợp này cần tính lại tọa độ mới theo:  $X' = X \cos \alpha - Y \sin \alpha$  và  $Y' = X \sin \alpha + Y \cos \alpha$ , với  $\alpha$  là góc lệch so với lề.

#### 7.5.3. Giai đoạn tách chữ

Sau giai đoạn xử lý sơ bộ, văn bản (ảnh) đã được tăng cường, ta chuyển sang giai đoạn tách chữ. Chỉ có thể nhận dạng đúng nếu chữ đã được tách khỏi văn bản. Có nhiều thuật toán tách chữ từ đơn giản đến phức tạp áp dụng cho các font chữ khác nhau: tách chữ theo chiều ngang - đứng và tách chữ theo theo lược đồ xám.

##### *Tách chữ theo chiều ngang và đứng*

Với chữ in thường và in hoa, do quy cách ấn loát luôn nằm trong một ô nào đó.

Như vậy, quá trình tách chữ đồng nhất với việc tìm ra khuôn chữ tại vị trí của nó trong văn bản. Quá trình này gọi là tách chữ theo hình chữ nhật (ngang và đứng) bao quanh ký tự. Thao tác này đơn giản và nhanh. Tuy nhiên không thể áp dụng với mọi font chữ.

#### *Tách chữ theo lược đồ xám*

Khi máy quét tốt và đối với một số font, các dòng trong văn bản được phân cách khá tốt, việc tìm ra đường phân ranh giữa 2 dòng là khá dễ. Song thực tế luôn không phải là dễ nhất là với chữ Việt có dấu, các dòng có thể dính hay chữ bị nhòe.

Trong trường hợp này, đường phân ranh được hiểu là đường có ít điểm cắt nhất. Như vậy cần xây dựng lược đồ xám cho các dòng chữ và đường nằm ngang tại đây của chúng lồng lược đồ cần tìm. Kỹ thuật này có thể áp dụng cho nhận dạng chữ hoa [10].

#### 7.5.4. Một số thuật toán nhận dạng chữ

Nhận dạng chữ sau khi đã tách khỏi từ là giai đoạn quan trọng nhất và cũng là mục đích của các hệ nhận dạng chữ viết. Mỗi loại chữ có những đặc điểm riêng nên các kỹ thuật áp dụng cũng khác nhau.

##### 7.5.4.1. Kỹ thuật nhận dạng chữ in

Chữ in thường rõ nét và chất lượng khá tốt sau khi quét. Trong nhận dạng chữ in, người ta thường dùng một số kỹ thuật:

- Kỹ thuật đổi sánh từng điểm - xuất phát từ tâm,
- Kỹ thuật đổi sánh các dây cắt dọc và cắt ngang,
- Kỹ thuật nhận dạng theo hình chiếu,...

##### *Kỹ thuật đổi sánh từng điểm xuất phát từ tâm*

Chữ sau khi được tách khỏi từ, tâm nó được tính toán và toạ độ được xác định. Chữ được đổi sánh với chữ chuẩn (nhận dạng chữ viết là bài toán học có mẫu) từng điểm một, từ tâm ra biên. Các hình vành khăn lồng nhau có trọng tâm tạo thành lớp các điểm ảnh có cùng trọng số. Khoảng cách giữa 2 điểm ảnh  $x$  và  $x'$  được tính:

- $DIST(x, x') = \begin{cases} 0 & \text{nếu } x = x' \\ w_x \neq w_{x'} \text{ với } w_x \text{ là trọng số của lớp } x \end{cases}$
- Khoảng cách giữa 2 ký tự X và X' :  $DIST(X, X') = \sum_{x \in X, x' \in X'} DIST(x, x')$

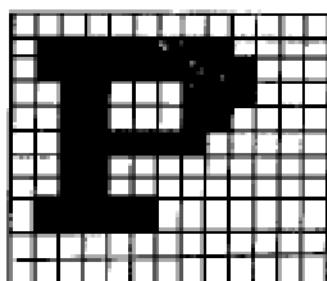
Nếu  $\text{DIST}(X, X') < \epsilon$  thì ký tự  $X$  được coi là  $X'$ ,  $\epsilon$  là giá trị ngưỡng định trước.

Thuật toán trên thực hiện khá nhanh, song nếu chất lượng quét tồi, các điểm chữ bị mất nhiều, trọng tâm bị lệch do đó sẽ ảnh hưởng đến chất lượng nhận dạng.

#### *Kỹ thuật nhận dạng dựa vào đổi sánh các điểm cắt dọc và cắt ngang*

Kỹ thuật này nhằm khắc phục một số nhược điểm của kỹ thuật trên. Giả sử chữ có lập có kích thước WChar và HChar. Chúng ta duyệt theo chiều ngang để tìm số điểm cắt theo chiều ngang.

Gọi  $H_i$  là số điểm cắt ngang của dòng thứ  $i$ . Như vậy  $H_1, H_2, \dots, H_{WChar}$  sẽ là dãy các điểm cắt ngang. Tương tự như vậy, gọi  $V_j$  là số điểm cắt dọc tại cột thứ  $j$ . Như vậy  $V_1, V_2, \dots, V_{HChar}$  sẽ là dãy các điểm cắt dọc. Bỏ qua các điểm 0 ở đầu và cuối 2 dãy, ta có 2 dãy con  $H = H_1, H_2, \dots, H_n$  và  $V_1, V_2, \dots, V_n$  sẽ là dãy các điểm cắt dọc, như chỉ ra trong hình 7.27 dưới đây:



Hình 7.27. Chữ P.

Khi đó quy tắc nhận dạng  $X'$  được xem là  $X$  nếu:  $H'_i \subseteq H_i$  hoặc  $H_i \subseteq H'_i$  và  $V'_j \subseteq V_j$ , hoặc  $V_j \subseteq V'_j$ . Nhìn chung, kỹ thuật này tương đối đơn giản, tốc độ cao và kết quả nhận dạng không phụ thuộc vào việc mất các điểm ở biên chữ. Tuy nhiên, kỹ thuật này đòi hỏi font phải chuẩn.

#### *Kỹ thuật nhận dạng dựa vào hình chiếu*

Kỹ thuật này là cải tiến của kỹ thuật trên, nhằm áp dụng cho nhiều kiểu font. Giả sử mẫu nhận dạng có kích thước  $n \times n$ . Gọi  $\xi_i$  là véctơ bậc  $n$  gồm các phần tử 0 và 1 tương ứng với hàng  $i$  (hay cột  $i$ ). Gọi  $\chi(\xi_i)$  là tổng số các phần tử 1 trong véctơ  $\xi_i$  và  $\beta(\xi_i)$  là số giao điểm của  $\xi_i$  với ảnh mẫu. Khi đó một hàng hay một cột được gọi là dài nếu:

$$\beta(\xi_i) = 1$$

$\chi(\xi_i) \geq \mu - \tau$ , với  $\mu$  là độ rộng của ký tự và  $\tau$  là ngưỡng cho trước (\*).

Ý nghĩa của hàng hay cột dài là chứng thể hiện chiều ngang hay chiều cao của ký tự. Đặt  $\xi_i^* = \xi_i \cup \xi_{i+1}$ . Nếu thoả mãn các điều kiện (\*), tức là:

$$\beta(\xi_i^*) = 1$$

$$\gamma(\xi_i^*) \geq \mu - \epsilon$$

khi đó ta có thể viết  $\beta(\xi_i^*) = 1$ . Để trích ra các đặc trưng của mẫu, ảnh được duyệt theo chiều ngang hay đứng như phương pháp trên. Tuy nhiên, ở đây ta có:

$$H_i = \beta(\xi_i^*) \quad \text{và} \quad V_i = \beta(\xi_i^*).$$

Tiếp đó, nếu trong các chuỗi H và V nếu  $H_i = H_i + 1$  hoặc  $V_i = V_i + 1$  thì phần tử  $H_i + 1$  hoặc  $V_i + 1$  bị xóa khỏi chuỗi. Cuối cùng ta thu được các chuỗi  $H'$  và  $V'$  đặc trưng cho ký tự.   
Thí dụ:

nếu  $H = 000111222221111111110$  thì  $H' = 012110$   
và  $V = 01112111133322222111111000$  thì  $V' = 0123210$

Quá trình nhận dạng trở thành so sánh các cặp  $H'$  và  $V'$ . Kỹ thuật này có ưu điểm là có thể áp dụng cho nhiều font. Song nếu chất lượng quyết tối, ảnh có nhiều răng cưa già thì chuỗi đặc trưng sẽ lệch nhiều so với chuỗi chuẩn. Ngoài các kỹ thuật kể trên còn có một số kỹ thuật khác như thống kê giao điểm, đồ thị đối sánh, v.v.

#### 7.5.4.2 Kỹ thuật nhận dạng chữ viết tay có ràng buộc

##### Một số ràng buộc

Vì chữ viết tay khá đa dạng, có thể gây nét hay có thể là các tiếng nước ngoài như tiếng Lào, Trung quốc, v.v. Nếu xét như vậy thì quá đa dạng không thể xét trong một thuật toán. Ở đây ta chỉ giới hạn các chữ hé La tinh:

- Các chữ đầu vào là thuộc hệ La tinh, có thể gồm các chữ số từ 0 đến 9 và viết trong một số kiểu font hạn chế (không quá bay bướm, không quá nghiêng).
- Cần có sự khác nhau giữa số "0" và chữ "o" vì nếu viết thường thì rất giống nhau. Nên ta giả định chúng được viết như kiểu viết của máy tính.
- Vì văn bản có thể có các đường kẻ, để nhận dạng được, ta giả định là chúng cách nhau một khoảng chấp nhận được.

### Kỹ thuật nhận dạng

Kỹ thuật nhận dạng ở đây dựa vào lý thuyết ra quyết định. Người ta xác định các đặc trưng của cấu trúc chữ như: số nhắt cắt ngang, các nét cong hay thẳng, mờ hay đóng, v.v. Cách sử dụng các dấu hiệu cũng khác nhau. Theo các tác giả [2], chữ được chia thành 2 nhóm lớn:

- Nhóm thứ nhất là nhóm gồm các chữ có ít nhất là một nhắt cắt một. Nhóm này gồm các chữ như: C E F G I J L P S T Y Z, các số từ 1 đến 7 và số 9

- Nhóm thứ hai gồm các chữ còn lại và 2 số 0 và số 8.

Sử dụng thêm tính chất đóng mở, ta lại chia nhóm hai thành 4 nhóm nhỏ:

- Đóng trên và đóng dưới: B D O Q 0 8

- Đóng trên mở dưới: A M R

- Mở trên đóng dưới: U V

- Mở trên, mở dưới: H K N X

Trên cơ sở đó, thêm các tính chất như tính cong hay thẳng bên phải, bên trái ta sẽ phân biệt được các chữ trong các nhóm nhỏ.

Đối với nhóm 1 do đặc tính của nó nên phải dùng phương pháp của số di động để xem xét. Dựa vào lát cắt, người ta chia chữ làm 6 phần và biểu diễn bởi một vectơ

$V: \{v1, v2, v3, v4, v5, v6\}$ :

$$V_i = \begin{cases} 1 & \text{nếu có một điểm đen trên phần } i \\ 0 & \text{nếu không} \end{cases}$$

I	II
III	IV
V	VI

Ví dụ như  $V(C) = (0, 1, 1, 0, 0, 1)$ ,  $V(J) = (1, 1, 0, 0, 1, 0)$

Kỹ thuật trên đã được cài đặt trong hệ nhận dạng VIETIN của công ty SEATIC.

#### 7.5.4.3. Thuật toán nhận dạng chữ tổng quát

Khác với kỹ thuật trên dựa vào lý thuyết ra quyết định trên cơ sở không dấu hiệu, kỹ thuật này dựa vào cấu trúc chữ. Theo kỹ thuật này, mỗi ký tự nhận dạng được biểu diễn bởi một xâu hay tổng quát hơn bởi một đồ thị của các dạng nguyên thuỷ và mối quan hệ giữa chúng. Như đã nêu trong phần nhận dạng cấu trúc, quá trình nhận dạng là quá trình phân tích cú pháp hay đối sánh đồ thị. Một văn bản coi như một dạng phức tạp cấu thành từ các dạng trung gian. Các dạng trung gian lại có thể coi được cấu tạo từ các dạng con (là kỹ

tự). Cuối cùng, mỗi ký tự được cấu thành từ các dạng nguyên thuỷ. Quá trình nhận dạng có thể biểu diễn theo sơ đồ sau:

Dạng nguyên thuỷ → Dạng con → Dạng trung gian → Dạng phức tạp

Kỹ thuật nhận dạng này bao gồm 3 công đoạn:

- Phân hoạch ký tự- biểu diễn dạng: Phân hoạch tập nhận dạng thành N tập đơn theo như lý thuyết phân hoạch không gian.
- Trích chọn các dạng nguyên thuỷ: Văn bản sau khi được xử lý sẽ bộ sưu qua phần trích chọn các đặc trưng mà ở đây là các điểm kết thúc, chạc ba.
- Nhận dạng dấu: Nhận dạng dấu là công đoạn quan trọng, nhất là trong nhận dạng chữ Việt. Dòng dấu thường nhỏ hơn và khó nhận dạng hơn.

#### *Phân hoạch ký tự-biểu diễn dạng*

Gọi  $\mu$  là tập các đối tượng nhận dạng:

$$\mu = \{A, B, C, \dots, Z\}$$

Mục tiêu của phương pháp là phân hoạch tập  $\mu$  thành N tập đơn nhất  $\xi_i$  sao cho:

$$\xi_i \cap \xi_j = \emptyset \text{ với } i \neq j, i = 1, 2, \dots, N \text{ (là số ký tự nhận dạng)}$$

$$\cup \xi_i = \mu$$

Bằng cách sử dụng một loạt các quy tắc, các dấu hiệu, thí dụ như :

- l: là số chu trình (loop),
- j: số điểm nối (junctions point: ngã 3, ngã tư),
- e: số điểm ngoặt (turning point),
- f: số điểm kết thúc (end point),
- t: hướng (trên, dưới, phải trái).

Ta phân hoạch tập đối tượng đã cho thành các tập con. Áp dụng tiếp các quy tắc, dấu hiệu này, ta lại 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:

$\mu_1 = \{A, D, O, P, Q, R\}, \mu_2 = \{B\}, \mu_3 = \{C, E, F, \dots\}$  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  $\mu_{11}$  ta thu được 5 tập nhỏ:

$$\mu_{11} = \{A, R\}, \mu_{12} = \{D\}, \mu_{13} = \{O\}, \mu_{14} = \{Q\}, \mu_{15} = \{P\}.$$

Nếu các tập thu được chưa phải là đơn nhất, ta áp dụng thêm các quy tắc khác như j để làm mịn nó. Với tập  $\mu_{11}$ , áp dụng quy tắc j ta chia nó thành 2 tập {A} và {R} vì chữ A có 2 điểm nối (chữ 3) mà chữ R chỉ có một. Cuối cùng ta có một phân hoạch không gian theo yêu cầu và chuyển sang bước trích chọn đặc trưng.

#### *Trích chọn các dạng nguyên thủy*

Các dạng nguyên thủy cần xác định ở đây là các điểm chạc ba và các điểm kết thúc.

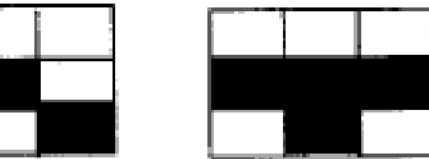
Các điểm này được định nghĩa như hình dưới đây. Một cách tổng quát như đã nói trong phần trên, điểm kết thúc là điểm có duy nhất một láng giềng đen; còn điểm chạc ba là điểm có NZP=3.



a) các điểm kết thúc



b) các điểm chạc ba



Hình 7.28. Một số dạng chạc ba và điểm kết thúc.

Các ký tự nhận dạng được làm mảnh sẽ được duyệt theo dòng để tìm kiếm các cột đen trên ảnh. Sau đó, quá trình duyệt tiến hành từ điểm xuất phát và theo thuật toán lân cận. đương nhiên thuật toán này có sử dụng một stack để lưu trữ vì cần xét hết 8 láng giềng để phát hiện các kiểu dấu hiệu. Việc xây dựng thuật toán này coi như bài tập tự giải.

#### *Nhận dạng dấu*

Tùy theo các ngôn ngữ ta có các tập dấu khác nhau. Với tiếng Việt, tập dấu gồm: các ký hiệu : [^ ~ . ? ^ v ^ ].

Nói chung, kích thước dòng dấu là quá nhỏ để có thể nhận dạng theo cấu trúc. Phương pháp áp dụng ở đây dựa vào thống kê giao điểm. Dấu “.” dễ dàng tìm được nhờ vào vị trí của nó trong font. Các dấu khác sẽ được nhận dạng dựa vào các đặc trưng.

#### *Thí dụ:*

- Dấu ^: có số điểm cắt ngang là H={1,2} và tồn tại 2 điểm y<sub>1</sub>, y<sub>2</sub> nhỏ hơn 2 giao của

dấu và một lát cắt qua tâm dấu.

- Dấu v: có số điểm cắt ngang là  $H = \{2,1\}$  và tồn tại 2 điểm  $y_1, y_2$  lớn hơn 2 giao của dấu và một lát cắt qua tâm dấu.

Bằng cách tương tự như vậy ta nhận dạng ra các dấu còn lại.

#### 7.5.5. Phục hồi văn bản

Thường trong các hệ nhận dạng tốt, tỉ lệ nhận dạng có thể đạt khá cao. Tuy nhiên còn lại một số vấn đề phải giải quyết như nhận dạng sai hay các trạng trí khác mà quá trình nhận dạng không thực hiện được. Đây cũng là công đoạn cuối cùng mà các hệ nhận dạng phải tiến hành. Thường là dùng văn bản gốc, phối hợp với từ điển hay một số phương pháp suy diễn khác.

### 7.6. KẾT LUẬN

Có rất nhiều vấn đề về nhận dạng khác mà chúng ta chưa đề cập đến như nhận dạng tín hiệu, nhận dạng tiếng nói, v.v. Các vấn đề này nằm trong lý thuyết nhận dạng.

Mục đích của chương chỉ nhằm cung cấp một số khái niệm về nhận dạng có liên quan đến xử lý ảnh và xem xét một vấn đề được quan tâm nhiều là nhận dạng chữ viết. Kỹ thuật mới về nhận dạng dựa trên mạng nơron đang được đánh giá rất cao và các kết quả thử nghiệm cho kết quả khá tốt, mà ở đây chúng tôi mới đề cập tới một cách rất sơ lược. Bạn đọc quan tâm xin tìm đọc thêm các tài liệu về nhận dạng.

# 8

## NÉN DỮ LIỆU ẢNH IMAGE COMPRESSION

### 8.1. TỔNG QUAN VỀ NÉN DỮ LIỆU ẢNH

Chương này nhằm cung cấp một số khái niệm (thuật ngữ) như: nén, tỉ lệ nén, các ý tưởng dẫn tới các phương pháp nén khác nhau và cách phân loại, đánh giá các phương pháp nén.

#### 8.1.1. Một số khái niệm

##### *Nén dữ liệu (Data Compression)*

Nén dữ liệu là quá trình làm giảm lượng thông tin "dư thừa" trong dữ liệu gốc và do vậy, lượng thông tin thu được sau nén thường nhỏ hơn dữ liệu gốc rất nhiều. Với dữ liệu ảnh, kết quả thường là 10 : 1. Một số phương pháp còn cho kết quả cao hơn. Theo kết quả nghiên cứu được công bố gần đây tại Viện kỹ thuật Georgie, kỹ thuật nén fractal cho tỉ số nén là 30 trên 1[6].

Ngoài thuật ngữ "nén dữ liệu", do bản chất của kỹ thuật này nó còn có một số tên gọi khác như: giảm độ dư thừa, mã hoá ảnh gốc.

Từ hơn hai thập kỷ nay, có rất nhiều kỹ thuật nén đã được công bố trên các tài liệu về nén và các phần mềm nén dữ liệu đã xuất hiện ngày càng nhiều trên thương trường. Tuy nhiên, chưa có phương pháp nén nào được coi là phương pháp vạn năng (Universal) vì nó phụ thuộc vào nhiều yếu tố và bản chất của dữ liệu gốc. Trong chương này, chúng ta không thể hy vọng xem xét tất cả các phương pháp nén. Hơn nữa, các kỹ thuật nén dữ liệu chung đã được trình bày trong nhiều tài liệu chuyên ngành. Ở đây, chúng ta chỉ đề cập các phương pháp nén có đặc thù riêng cho dữ liệu ảnh.

##### *Tỉ lệ nén (Compression rate)*

Tỉ lệ nén là một trong các đặc trưng quan trọng nhất của mọi phương pháp nén. Tuy nhiên, về cách đánh giá và các kết quả công bố trong các tài liệu cũng cần được quan tâm xem xét. Nhìn chung, người ta định nghĩa tỉ lệ nén như sau:

$$\text{Tỷ lệ nén} = \frac{1}{r} \times \%$$

với  $r$  là tỷ số nén được định nghĩa;  $r = \text{kích thước dữ liệu gốc}/\text{kích thước dữ liệu thu được sau nén}$ . Như vậy hiệu suất của nén là:  $(1 - \text{tỷ lệ nén}) \times \%$ .

Trong các trình bày sau, khi nói đến kết quả nén, chúng ta dùng tỷ số nén, thí dụ như 10 trên 1 có nghĩa là dữ liệu gốc là 10 phần sau khi nén chỉ có 1 phần.

Tuy nhiên, cũng phải thấy rằng những số đo của một phương pháp nén chỉ có giá trị với chính sự nén đó, vì rằng hiệu quả của nén còn phụ thuộc vào kiểu dữ liệu định nén. Tỷ lệ nén cũng chỉ là một trong các đặc trưng cơ bản của phương pháp nén. Nhiều khi tỷ lệ nén cao cũng chưa thể nói rằng phương pháp đó là hiệu quả hơn các phương pháp khác, vì còn các chi phí khác như thời gian, không gian và thậm chí cả độ phức tạp tính toán nữa. Thí dụ như nén phục vụ trong truyền dữ liệu; vấn đề đặt ra là hiệu quả nén có tương hợp với đường truyền không.

Cũng cần phân biệt nén dữ liệu với nén bằng truyền. Mục đích chính của nén là giảm lượng thông tin dư thừa và dần tối giảm kích thước dữ liệu. Tuy vậy, đôi khi quá trình nén cũng làm giảm băng truyền tín hiệu số hóa thấp hơn so với truyền tín hiệu tương tự.

### 8.1.2. Các loại dư thừa dữ liệu

Như trên đã nói, nén nhằm mục đích giảm kích thước dữ liệu bằng cách loại bỏ dư thừa dữ liệu. Việc xác định bản chất các kiểu dư thừa dữ liệu rất có ích cho việc xây dựng các phương pháp nén dữ liệu khác nhau. Nói một cách khác, các phương pháp nén dữ liệu khác nhau là do sử dụng các kiểu dư thừa dữ liệu khác nhau. Người ta coi có 4 kiểu dư thừa chính:

- \* **Sự phân bố ký tự**

Trong một dãy ký tự, có một số ký tự có tần suất xuất hiện nhiều hơn một số dãy khác. Do vậy, ta có thể mã hoá dữ liệu một cách có độ đọng hơn. Các dãy ký tự có tần suất cao được thay bởi một từ mã nhị phân với số bit nhỏ; ngược lại các dãy có tần suất thấp sẽ được mã hoá bởi từ mã có nhiều bit hơn. Đây chính là bản chất của phương pháp mã hoá Huffman (xem mục 8.2.2).

- \* **Sự lặp lại của các ký tự**

Trong một số tình huống như trong ảnh, 1 ký hiệu (bit "0" hay bit "1") được lặp đi

lặp lại một số lần. Kỹ thuật nén dùng trong trường hợp này là thay dây lặp đó bởi dây mới gồm 2 thành phần: số lần lặp và kí hiệu dùng để mã. Phương pháp mã hóa kiểu này có tên là mã hóa loạt dài RLC (Run Length Coding). Phương pháp mã hóa RLC sẽ được trình bày trong mục 8.2.1.

- **Những mẫu sử dụng tần suất**

Có thể có dây ký hiệu nào đó xuất hiện với tần suất tương đối cao. Do vậy, có thể mã hóa bớt ít bớt hơn. Đây là cơ sở của phương pháp mã hóa kiểu từ điển do Lempel-Ziv đưa ra và có cải tiến vào năm 1977, 1978 và do đó có tên gọi là phương pháp nén LZ77, LZ78. Năm 1984, Terry Welch đã cải tiến hiệu quả hơn và đặt tên là LZW (Lempel-Ziv-Welch). Phương pháp nén này sẽ được trình bày trong 8.2.3.

- **Dộ dư thừa vị trí**

Do sự phụ thuộc lẫn nhau của dữ liệu, đôi khi biết được ký hiệu (giá trị) xuất hiện tại một vị trí, đồng thời có thể đoán trước sự xuất hiện của các giá trị ở các vị trí khác nhau một cách phù hợp. Chẳng hạn, ảnh biểu diễn trong một lưới hai chiều, một số điểm ở hàng dọc trong một khối dữ liệu lại xuất hiện trong cùng vị trí ở các hàng khác nhau. Do vậy, thay vì lưu trữ dữ liệu, ta chỉ cần lưu trữ vị trí hàng và cột. Phương pháp nén dựa trên sự dư thừa này gọi là phương pháp mã hóa dự đoán.

Cách đánh giá độ dư thừa như trên hoàn toàn mang tính trực quan nhằm biểu thị một cái gì đó xuất hiện nhiều lần. Đối với dữ liệu ảnh, ngoài đặc thù chung đó, nó còn có những đặc thù riêng. Thí dụ như có ứng dụng không cần toàn bộ dữ liệu thô của ảnh mà chỉ cần các thông tin đặc trưng biểu diễn ảnh như biến ảnh hay vùng đồng nhất. Do vậy, có những phương pháp nén riêng cho ảnh dựa vào biến đổi ảnh hay dựa vào biểu diễn ảnh. Các phương pháp này sẽ lần lượt trình bày trong mục 8.3 và 8.4.

### 8.1.3. Phân loại các phương pháp nén

Có nhiều cách phân loại các phương pháp nén khác nhau. Cách thứ nhất dựa vào nguyên lý nén. Cách này phân các phương pháp nén thành 2 họ lớn:

- **Nén chính xác hay nén không mất thông tin:** họ này bao gồm các phương pháp nén mà sau khi giải nén ta thu được chính xác dữ liệu gốc.
- **Nén có mất mát thông tin:** họ này bao gồm các phương pháp mà sau khi giải nén

tù không thu được dữ liệu như bản gốc. Trong nén ảnh, người ta gọi là các phương pháp “*tóm lý thị giác*”. Các phương pháp này lợi dụng tính chất của mắt người, chấp nhận một số vẩn xoắn trong ảnh khi khôi phục lại. Tuy nhiên, các phương pháp này chỉ có hiệu quả khi mà độ vẩn xoắn là chấp nhận được bằng mắt thường hay với dung sai nào đó.

Cách phân loại thứ hai dựa vào cách thức thực hiện nén. Theo cách này, người ta cũng phân thành hai họ:

- **Phương pháp không gian (Spatial Data Compression):** các phương pháp thuộc họ này thực hiện nén bằng cách tác động trực tiếp lên việc lấy mẫu của ảnh trong miền không gian.

- **Phương pháp sử dụng biến đổi (Transform Coding):** gồm các phương pháp tác động lên sự biến đổi của ảnh gốc mà không tác động trực tiếp như họ trên [6].

Có một cách phân loại khác nữa, cách phân loại thứ ba, dựa vào triết lý của sự mã hóa. Cách này cũng phân các phương pháp nén thành 2 họ:

- **Các phương pháp nén thế hệ thứ nhất:** gồm các phương pháp mà mức độ tính toán là đơn giản, thí dụ như việc lấy mẫu, gán tử mã, v.v.

**Các phương pháp nén thế hệ thứ hai:** dựa vào mức độ bao hoà của tỷ lệ nén.

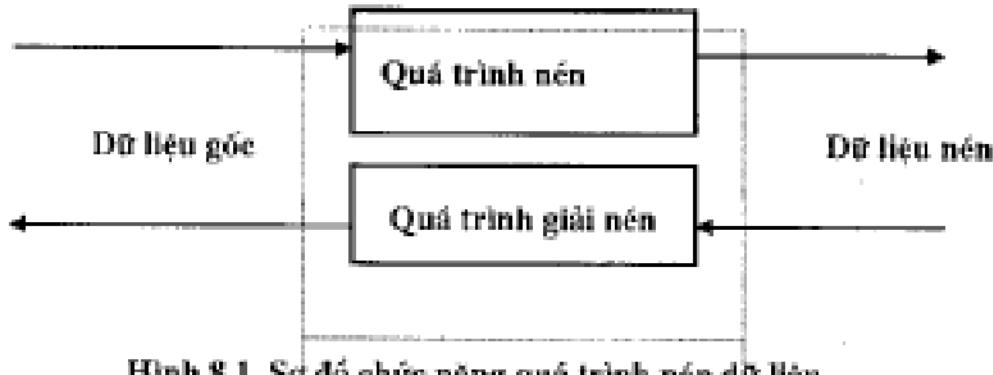
Trong các phần trình bày dưới đây, ta sẽ theo cách phân loại này.

Cũng còn phải kể thêm một cách phân loại thứ tư do Anil.K.Jain nêu ra. Theo cách của Jain, các phương pháp nén gồm 4 họ chính:

- Phương pháp điểm.
- Phương pháp dự đoán.
- Phương pháp dựa vào biến đổi.
- Các phương pháp tổ hợp (Hybrid).

Thực ra cách phân loại này là chia nhỏ của cách phân loại thứ ba và dựa vào cơ chế thực hiện nén. Xét một cách kỹ lưỡng nó cũng tương đương cách phân loại thứ ba.

Nhìn chung, quá trình nén và giải nén dữ liệu có thể mô tả một cách tóm tắt theo sơ đồ hình 8.1 dưới đây.



Hình 8.1. Sơ đồ chức năng quá trình nén dữ liệu.

## 8.2. CÁC PHƯƠNG PHÁP NÉN THẾ HỆ THỨ NHẤT

Trong lớp các phương pháp này, ta lần lượt xem xét các phương pháp:

- Mã hoá loạt dài RLC (Run Length Coding)
- Mã hoá Huffman
- Mã hoá LZW(Lempel Ziv-Wenck)
- Mã hoá khối (Block Coding).

### 8.2.1. Phương pháp mã hoá loạt dài

Phương pháp mã hoá loạt dài lúc đầu được phát triển dành cho ảnh số 2 mức: mức đen (1) và mức trắng (0) như các văn bản trên nền trắng, trang in, các bản vẽ kỹ thuật.

Nguyên tắc của phương pháp là phát hiện một loạt các bit lặp lại, thí dụ như một loạt các bit 0 nằm giữa hai bit 1, hay ngược lại, một loạt bit 1 nằm giữa hai bit 0. Phương pháp này chỉ có hiệu quả khi chiều dài dãy lặp lớn hơn một ngưỡng nào đó. Dãy các bit lặp gọi là loạt hay mạch (run). Tiếp theo, thay thế chuỗi đó bởi một chuỗi mới gồm 2 thông tin: chiều dài chuỗi và bit lặp (ký tự lặp). Như vậy, chuỗi thay thế sẽ có chiều dài ngắn hơn chuỗi cần thay.

Cần lưu ý rằng, đối với ảnh, chiều dài của chuỗi lặp có thể lớn hơn 255. Nếu ta dùng 1 byte để mã hoá thì sẽ không đủ. Giải pháp được dùng là tách chuỗi đó thành 2 chuỗi: một chuỗi có chiều dài 255, chuỗi kia là số bit còn lại.

Phương pháp RLC được sử dụng trong việc mã hoá lưu trữ các ảnh Bitmap theo dạng PCX, BMP (đã nêu trong chương 2).

Phương pháp RLC có thể chia thành 2 phương pháp nhỏ: phương pháp dùng chiều dài từ mã cố định và phương pháp thích nghi như kiểu mã Huffman. Giả sử các mạch gồm M bits. Để tiện trình bày, đặt  $M = 2^n - 1$ . Như vậy mạch cũ được thay bởi mạch mới gồm m bits.

Với cách thức này, mọi mạch đều được mã hoá bởi từ mã có cùng độ dài. Người ta cũng tính được, với  $M = 15$ ,  $p = 0,9$ , ta sẽ có  $m = 4$  và tỷ số nén là 1,95.

Với chiều dài cố định, việc cài đặt thuật toán là đơn giản. Tuy nhiên, tỷ lệ nén sẽ không tốt bằng dùng chiều dài biến đổi hay gọi là mã RLC thích nghi.

### 8.2.2. Phương pháp mã hoá Huffman

#### *Nguyên tắc*

Phương pháp mã hoá Huffman là phương pháp dựa vào mô hình thống kê. Dựa vào dữ liệu gốc, người ta tính tần suất xuất hiện của các ký tự. Việc tính tần suất được thực hiện bằng cách duyệt tuần tự tập gốc từ đầu đến cuối. Việc xử lý ở đây tính theo bit. Trong phương pháp này, người ta gán cho các ký tự có tần suất cao một từ mã ngắn, các ký tự có tần suất thấp từ mã dài. Nói một cách khác, các ký tự có tần suất càng cao được gán mã càng ngắn và ngược lại. Rõ ràng với cách thức này, ta đã làm giảm chiều dài trung bình của từ mã hoá bằng cách dùng chiều dài biến đổi. Tuy nhiên, trong một số tình huống khi tần suất là rất thấp, ta có thể không được lợi một chút nào, thậm chí còn bị thiệt một bit.

#### *Thuật toán*

Thuật toán bao gồm 2 bước chính:

- Giai đoạn tính tần suất của các ký tự trong dữ liệu gốc: Duyệt tập gốc một cách tuần tự từ đầu đến cuối để xây dựng bảng mã. Tiếp sau đó là sắp xếp lại bảng mã theo thứ tự tần suất giảm dần.
- Giai đoạn thứ hai: mã hoá. Duyệt bảng tần suất từ cuối lên đầu để thực hiện ghép 2 phần tử có tần suất thấp nhất thành một phần tử duy nhất. Phần tử này có tần suất bằng tổng 2 tần suất thành phần. Tiến hành cập nhật lại bảng và đương nhiên loại bỏ 2 phần tử đã xét. Quá trình được lặp lại cho đến khi bảng chỉ có một phần tử. Quá trình này gọi là quá trình tạo cây mã Huffman vì việc tập hợp được tiến hành nhờ một cây nhị phân với 2 nhánh. Phần tử có tần suất thấp ở bên phải, phần tử kia ở bên trái. Với cách tạo cây này, tất cả các bit dữ liệu/ký tự là nút lá; các nút trong là các nút tổng hợp. Sau khi cây đã tạo xong, người ta tiến hành gán mã cho các nút lá. Việc mã hoá rất đơn giản: mỗi lần xuống bên phải ta thêm 1 bit "1" vào từ mã; mỗi lần xuống bên trái ta thêm 1 bit "0". Tất nhiên có thể làm ngược lại, chỉ có giá trị mã thay đổi còn tổng chiều dài là không đổi. Cũng chính do lý do này mà cây có tên gọi là cây mã Huffman như trên đã gọi.

Quá trình giải nén tiến hành theo chiều ngược lại khá đơn giản. Người ta cũng phải dựa vào bảng mã tạo ra trong giai đoạn nén (bảng này được giữ lại trong cấu trúc đầu của tệp nén cùng với dữ liệu nén). Thị dụ, với một tệp dữ liệu mà tần suất các ký tự cho bởi:

Ký tự	Tần suất	Ký tự	Tần suất	Xác suất
"1"	152	"0"	1532	0,2770
"2"	323	"6"	602	0,1088
"3"	412	"."	536	0,0969
"4"	226	" "	535	0,0967
"5"	385	"3"	112	0,0746
"6"	602	"5"	385	0,0696
"7"	92	"2"	323	0,0585
"8"	112	"_"	315	0,0569
"9"	87	"4"	226	0,0409
"0"	1532	"+"	220	0,0396
"."	536	"1"	152	0,0275
"+"	220	"8"	112	0,0203
"_"	315	"7"	92	0,0167
" "	535	"9"	87	0,0158

Bảng tần suất

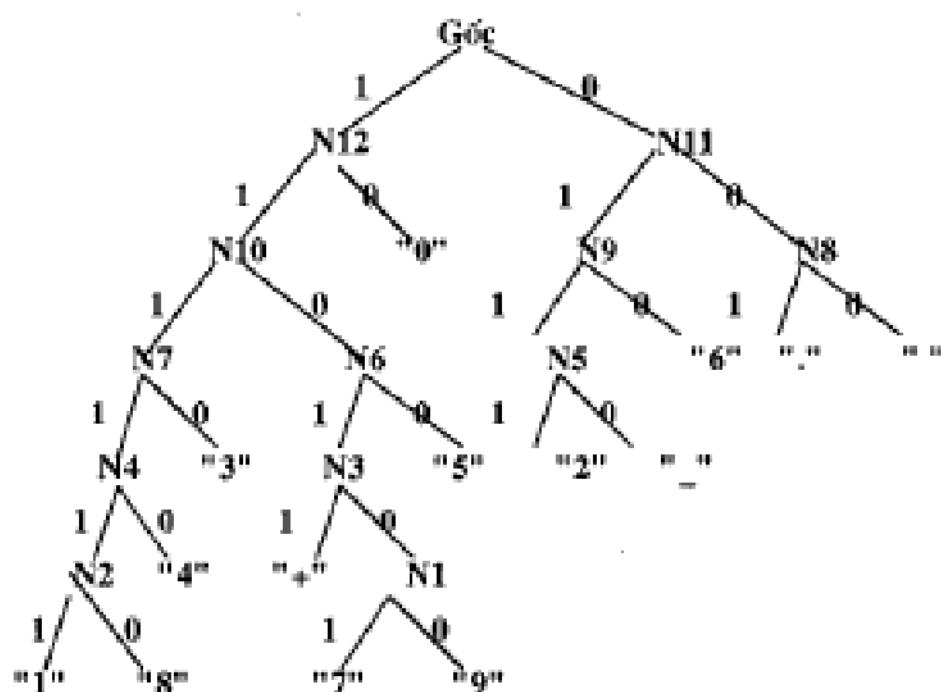
Bảng tần suất sắp theo thứ tự giảm dần

Lưu ý rằng, trong phương pháp Huffman, mã của ký tự là duy nhất và không mã nào là phần bắt đầu của mã khác. Vì vậy, khi đọc tệp nén từng bit từ đầu đến cuối ta có thể duyệt cây mã cho đến một lá, tức là ký tự đã được giải nén.

Bảng từ mã gán cho các ký tự bởi mã hóa Huffman

"0"	10	"_"	0110
"6"	010	"4"	11110
"."	001	"+"	11011
" "	000	"1"	11111
"3"	1110	"8"	111110
"5"	1100	"7"	110101
"2"	0111	"9"	110100

Cây mã Huffman tương ứng



Hình 8.2. Cây mã Huffman

### 8.2.3. Phương pháp LZW

#### *Mở đầu*

Khái niệm nén từ điển được Jacob Lempel và Abraham Ziv đưa ra lần đầu tiên vào năm 1977, sau đó phát triển thành một họ giải thuật nén từ điển LZ. Năm 1984, Terry Welch đã cải tiến giải thuật LZ thành một giải thuật mới hiệu quả hơn và đặt tên là LZW. Phương pháp nén từ điển dựa trên việc xây dựng từ điển lưu các chuỗi ký tự có tần suất lặp lại cao và thay thế bằng từ mã tương ứng mỗi khi gặp lại chúng. Giải thuật LZW hay hơn các giải thuật trước nó ở kĩ thuật tổ chức từ điển cho phép nâng cao tần suất nén.

Giải thuật nén LZW được sử dụng cho tất cả các loại file nhị phân. Nó thường được dùng để nén các loại văn bản, ảnh đen trắng, ảnh màu, ảnh đa mức xám... và là chuẩn nén cho các dạng ảnh GIF và TIFF. Mức độ hiệu quả của LZW không phụ thuộc vào số bit màu của ảnh.

#### *Phương pháp*

- Giải thuật nén LZW xây dựng một từ điển lưu các mẫu có tần suất xuất hiện cao

trong ảnh. Từ điển là tập hợp những cặp từ vựng và nghĩa của nó. Trong đó, từ vựng sẽ là các từ mã được sắp xếp theo thứ tự nhất định. Nghĩa là một chuỗi con trong dữ liệu ảnh. Từ điển được xây dựng đồng thời với quá trình đọc dữ liệu. Sự có mặt của một chuỗi con trong từ điển khẳng định rằng chuỗi đó đã từng xuất hiện trong phần dữ liệu đã đọc. Thuật toán liên tục "tra cứu" và cập nhật từ điển sau mỗi lần đọc một kí tự ở dữ liệu đầu vào.

Do kích thước bộ nhớ không phải vô hạn và để đảm bảo tốc độ tìm kiếm, từ điển chỉ giới hạn 4096 ở phần tử dùng để lưu lớn nhất là 4096 giá trị của các từ mã. Như vậy độ dài lớn nhất của từ mã là 12 bits ( $4096 = 2^{12}$ ). Cấu trúc từ điển như sau:

0	0
1	1
...	...
...	...
255	255
256	<i>(Clear Code)</i>
257	<i>(End Of Information)</i>
258	Chuỗi
259	Chuỗi
...	...
...	...
4095	Chuỗi

+ 256 từ mã đầu tiên theo thứ tự từ 0..255 chứa các số nguyên từ 0..255. Đây là mã của 256 kí tự cơ bản trong bảng mã ASCII.

+ Từ mã thứ 256 chứa một mã đặc biệt là "mã xoá" (CC - *Clear Code*). Mục đích việc dùng mã xoá nhằm khắc phục tình trạng số mẫu lặp trong ảnh lớn hơn 4096. Khi đó một ảnh được quan niệm là nhiều mảnh ảnh, và từ điển là một bộ từ điển gồm nhiều từ điển con. Cứ hết một mảnh ảnh người ta lại gửi một mã xoá để báo hiệu kết thúc mảnh ảnh cũ, bắt đầu mảnh ảnh mới đồng thời khởi tạo lại từ điển cho mảnh ảnh mới. Mã xoá có giá trị là 256.

+ Từ mã thứ 257 chứa mã kết thúc thông tin (EOI - *End Of Information*). Mã này có giá trị là 257. Như chúng ta đã biết, một file ảnh GIF có thể chứa nhiều ảnh. Mỗi một ảnh sẽ được mã hoá riêng. Chương trình giải mã sẽ lặp đi lặp lại thao tác giải mã từng ảnh cho đến khi gặp mã kết thúc thông tin thì dừng lại.

+ Các từ mã còn lại (từ 258 đến 4095) chứa các mẫu thường lặp lại trong ảnh. 512 phần tử đầu tiên của từ điển biểu diễn bằng 9 bit. Các từ mã từ 512 đến 1023 biểu diễn bởi 10 bit, từ 1024 đến 2047 biểu diễn bởi 11 bit và từ 2048 đến 4095 biểu diễn bởi 12 bit.

#### Ví dụ minh họa cơ chế nén của LZW

Cho chuỗi đầu vào là "ABCBCABCABCD" (Mã ASCII của A là 65, B là 66, C là 67).

Từ điển ban đầu đã gồm 256 ký tự cơ bản.

Dầu vào	Dầu Ra	Thực hiện
A (65)		A đã có trong từ điển $\Rightarrow$ Đọc tiếp
B (66)	65	Thêm vào từ điển mã 258 đại diện cho chuỗi AB
C (67)	66	Thêm vào từ điển mã 259 đại diện cho chuỗi BC
B	67	Thêm vào từ điển mã 260 đại diện cho chuỗi CB
C		BC đã có trong từ điển $\Rightarrow$ Đọc tiếp
A	259	Thêm vào từ điển mã 261 đại diện cho chuỗi BCA
B		AB đã có trong từ điển $\Rightarrow$ Đọc tiếp
C	258	Thêm vào từ điển mã 262 đại diện cho chuỗi ABC
A	67	Thêm vào từ điển mã 263 đại diện cho chuỗi CA
B		AB đã có trong từ điển $\Rightarrow$ Đọc tiếp
C		ABC đã có trong từ điển $\Rightarrow$ Đọc tiếp
D	262	Thêm vào từ điển mã 263 đại diện cho chuỗi ABCD

Chuỗi đầu ra sẽ là:

65 - 66 - 67 - 259 - 258 - 67 - 262

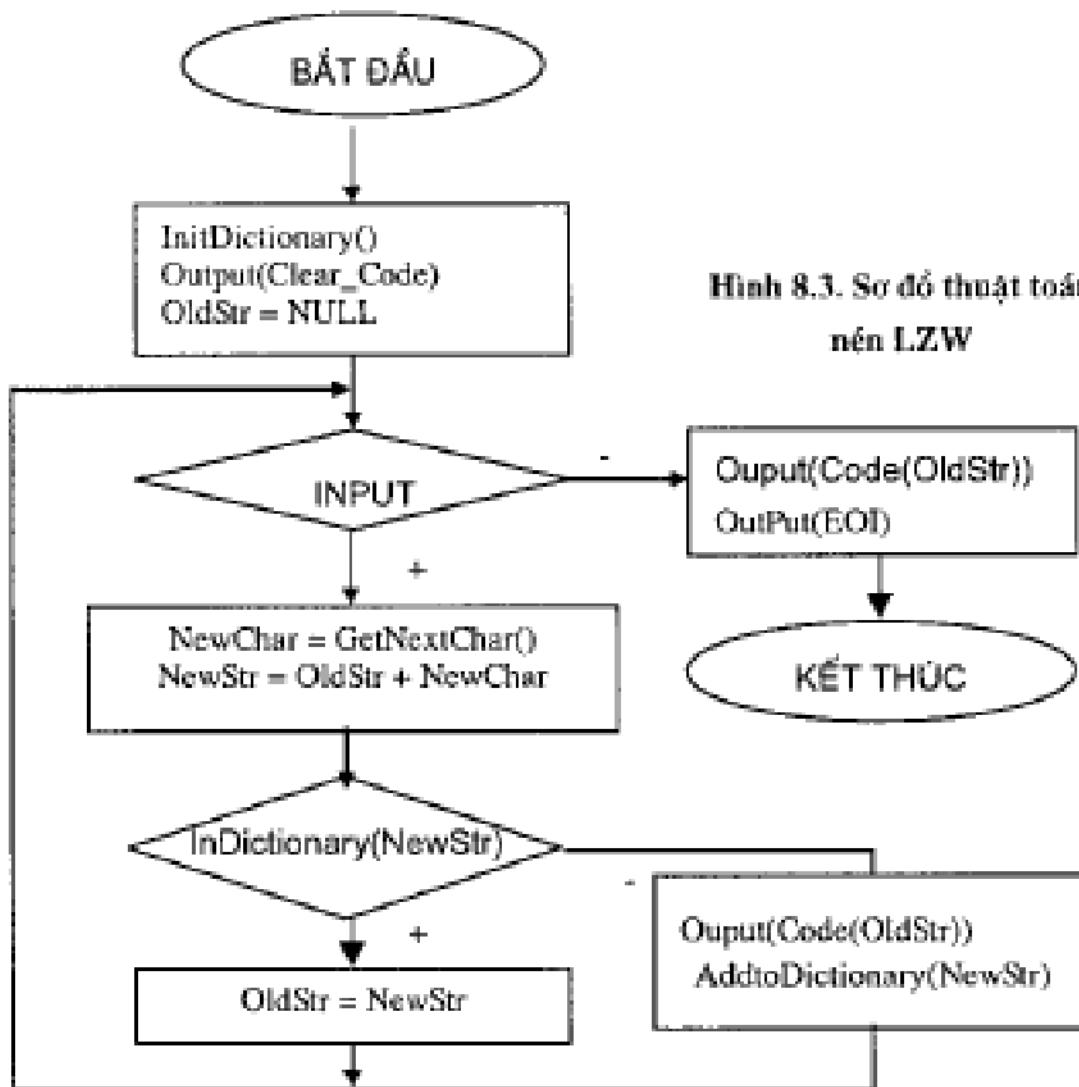
Dầu vào có kích thước:  $12 \times 8 = 96$  bits. Dầu ra có kích thước là:  $4 \times 8 + 3 \times 9 = 59$  bits

Tỉ lệ nén là:  $96:59 \approx 1,63$ .

#### Thuật toán

- Giá trị cờ INPUT = TRUE khi vẫn còn dữ liệu đầu vào và ngược lại.
- Các chức năng của các hàm:

- *InitDictionary()*: Hàm này có chức năng khởi tạo từ điển. Đặt giá trị cho 256 phần tử đầu tiên. Gán mã xoá (*Clear Code*) cho phần tử thứ 256 và mã kết thúc thông tin (*End Of Information*) cho phần tử thứ 257. Xoá giá trị tất cả các phần tử còn lại.
- *Hàm Output()*: gửi chuỗi bit ra file. Chuỗi bit này có độ dài là 9,10,11 hoặc 12 tùy thuộc vào vị trí trong từ điển của từ mà gửi ra. Các chuỗi bit này được nối tiếp vào với nhau.
- *Hàm GetNextChar()*: Trả về một ký tự từ chuỗi ký tự đầu vào. Hàm này cập nhật giá trị của cờ INPUT xác định xem còn dữ liệu đầu vào nữa hay không.
- *Hàm AddtoDictionary()* sẽ được gọi khi có một mẫu mới xuất hiện. Hàm này sẽ cập nhật mẫu này vào phần tử tiếp theo trong từ điển. Nếu từ điển đã đầy nó sẽ gửi ra mã xoá(*Clear Code*) và gọi đến hàm *InitDictionary()* để khởi tạo lại từ điển.
- *Hàm Code()*: Trả về từ mã ứng với một chuỗi.



Hình 8.3. Sơ đồ thuật toán  
nén LZW

Tư tưởng của đoạn mã trên có thể hiểu như sau: Nếu còn dữ liệu đầu vào thì tiếp tục đọc. Một chuỗi mới sẽ được tạo ra từ chuỗi cũ(*chuỗi này ban đầu trống, chuỗi này phải là chuỗi đã tồn tại trong từ điển*) và kí tự vừa đọc vào. Sau đó kiểm tra xem chuỗi mới đã có trong từ điển hay chưa. Mục đích của công việc này là hi vọng tìm được chuỗi có số kí tự lớn nhất đã tồn tại trong từ điển. Nếu tồn tại ta lại tiếp tục đọc một kí tự tiếp theo và lặp lại công việc. Nếu chưa có trong từ điển, thì gởi chuỗi cũ ra ngoài và thêm chuỗi mới vào từ điển. Có thể xem lại phần ví dụ để hiểu rõ hơn.

#### *Gửi nén dữ liệu nén bằng LZW*

Giải thuật giải nén gần như ngược với giải thuật nén . Với giải thuật nén, một từ mã ứng với một chuỗi sẽ được ghi ra tệp khi chuỗi ghép bởi chuỗi trên với kí tự vừa đọc chưa có mặt trong từ điển. Người ta cũng cập nhật ngay vào từ điển từ mã ứng với chuỗi tạo bởi chuỗi cũ với kí tự vừa đọc. Kí tự này đồng thời là kí tự đầu tiên trong chuỗi ứng với từ mã sẽ được ghi ra tiếp theo. Đây là điểm mấu chốt cho phép xây dựng thuật toán giải nén.

Thuật toán được mô tả như sau:

```

while(GetNextCode() != EOI) do
    Begin
        if FIRST_CODE /* Mã đầu tiên của mỗi mảnh ảnh*/
            Then Begin
                OutBuff(code);
                OldStr := code;
            End;
        If code = CC/* Mã xoá*/
            Then Begin
                InitDictionary();
                FIRST_CODE = TRUE;
            End;
        NewStr := DeCode(code);
        OutBuff(NewStr);
    End;

```

```

OldString = OldStr + FirstChar(NewStr);
AddtoDictionary(OldStr);
OldString := NewStr;
End;

```

+ Giá trị cờ FIRST\_CODE = TRUE chỉ mã vừa đọc là mã đầu tiên của mỗi mảnh ảnh. Mã đầu tiên có cách xử lý khác so với các mã tiếp theo.

+ Mã CC báo hiệu hết một mảnh ảnh. Mã EOI báo hiệu hết toàn bộ thông tin ảnh.

+ Chức năng của các hàm:

- *GetNextCode()* : Hàm này đọc thông tin đầu vào(*dữ liệu nén*) trả về mã tương ứng. Chúng ta nhớ lại rằng, dữ liệu nén gồm chuỗi các từ mã nối tiếp nhau. Ban đầu là 9 bit, sau đó tăng lên 10 bit rồi 11, 12 bit. Nhiệm vụ của hàm này không phải đơn giản. Để biết được tại thời điểm hiện thời, từ mã dài bao nhiêu bit ta phải luôn theo dõi từ điển và cập nhật độ dài từ mã tại các phần tử thứ 512, 1024, 2048.

- *OutBuff()* Hàm này gửi chuỗi giá trị đã giải mã ra vùng nhớ đệm.

- *DeCode()* Hàm này tra cứu từ điển và trả về chuỗi kí tự tương ứng với từ mã.

- *FirstChar()* Lấy kí tự đầu tiên của một chuỗi. Kí tự vừa xác định nối tiếp vào chuỗi kí tự cũ (*đã giải mã ở bước trước*) ta được chuỗi kí tự có mặt trong từ điển khi nén. Chuỗi này sẽ được thêm vào từ điển giải nén.

- *Hàm Output()* : gửi chuỗi bit ra file. Chuỗi bit này có độ dài là 9,10,11 hoặc 12 tuỳ thuộc vào vị trí trong từ điển của từ mã gửi ra.Các chuỗi bit này được nối tiếp vào với nhau.

#### *Trường hợp ngoại lệ và cách xử lý*

Đối với giải thuật LZW tồn tại một trường hợp được sinh ra nhưng chương trình giải nén có thể không giải mã được. Giả sử c là một kí tự, S là một chuỗi có độ dài lớn hơn 0. Nếu mã k' của từ điển chứa giá trị là cS. Chuỗi đầu vào là cScS. Khi đọc đến cSc chương trình sẽ tạo mã k' chứa cSc. Ngay sau đó k' được dùng thay thế cho cSc. Trong chương trình giải nén, k' sẽ xuất hiện trước khi nó được định nghĩa. Rất may là từ mã vừa đọc trong trường hợp này bao giờ cũng có nội dung trùng với tổ hợp của từ mã cũ với kí tự đầu tiên của nó. Điều này giúp cho quá trình cài đặt chương trình khắc phục được trường hợp ngoại lệ một cách dễ dàng.

### 8.2.4. Phương pháp mã hóa khối (Block Coding)

#### *Nguyên tắc*

Phương pháp này lúc đầu được phát triển cho ảnh số 2 mức xám, sau đó hoàn thiện thêm bởi các phương pháp thích nghi và mở rộng cho ảnh số đa cấp xám.

Cho một ảnh số  $I(x,y)$  kích thước  $M \times N$ . Người ta chia nhỏ ảnh số thành các khối hình chữ nhật kích thước  $k \times l$ ,  $(k,l)$  là rất nhỏ so với  $M, N$ . Như vậy ảnh gốc coi như gồm các khối con xếp cạnh nhau và có  $N \times M / (k \times l)$  khối con.

Ta có thể dùng phương pháp mã hóa Huffman cho từng khối của ảnh gốc, nghĩa là gán cho mỗi từ khối một từ mã nhị phân như ở phần trên. Một khuyết điểm là khi dùng mã hóa tối ưu Huffman đó là số lượng khối quá lớn. Giải pháp ở đây là dùng mã hóa gần tối ưu, đơn giản hơn để thực hiện mã hóa.

Ta giả thiết các khối là độc lập với nhau và số cấu hình là  $2^k$ . Gọi  $p(i,k,l)$  là xác suất xuất hiện cấu hình  $i$ , entropy tương ứng là:

$$H(k,l) = - \sum_{i=1}^{2^k} p(i,k,l) \log_2 P(i,k,l)$$

Giá trị  $H(k,l)$  có thể diễn giải là số bit/khối.

Các từ mã gán cho các khối  $k \times l$  được tạo bởi các điểm trắng (cấu hình trống) là "0". Các từ mã gán cho các khối  $k \times l$  khác gồm k+l bit màu ("1" cho đen, "0" cho trắng) đi tiếp sau 1 bit tiền tố "1".

Việc mã hóa theo khối cũng được sử dụng nhiều trong các phương pháp khác như phương pháp dùng biến đổi sẽ trình bày trong mục 8.3 để giảm bớt không gian lưu trữ.

#### *Thuật toán*

Giả sử  $p(0,k,l)$  xác suất của khối chỉ tạo bởi các điểm trắng là đã biết, t ý số nén có thể tính được dễ dàng. Xác suất này có thể được thiết lập bởi mô hình lý thuyết cho một kiểu khối đặc biệt. Do vậy, ta chia khối làm 2 loại: khối 1 chiều và khối 2 chiều.

- *Khối 1 chiều*

Xác suất  $p(0,k,l)$  tính được nhờ vào mô hình của quá trình Markov bậc một. Quá trình này được biểu diễn nhờ ma trận dịch chuyển trạng thái  $\Pi$ :

$$\Pi = \begin{pmatrix} p(t/t) & p(d/t) \\ p(t/d) & p(d/d) \end{pmatrix} \quad (8.1)$$

với : -  $p(t/t)$  là xác suất có điều kiện trắng sang trắng.

-  $p(d/d)$  là xác suất có điều kiện đen sang đen. Các xác suất khác có ý nghĩa tương tự.

Như vậy:  $p(0,k,l) = p(t)p(t)^{k-1}$ . (8.2)

Điều này có thể giải thích như sau: xác suất xuất hiện một khói k x 1 chỉ gồm các điểm trắng bằng xác suất xuất hiện 1 điểm trắng tiếp theo k-1 dịch chuyển trắng sang trắng. Dựa vào các quan hệ trên, ta tính được tỷ số nền  $C_r$ .

$$C_r = \frac{1}{k[1-p(t)]p(t)^{k-1}+1} \quad (8.3)$$

#### • Khối 2 chiều

Xác suất  $p(0,k,l)$  của các khói toàn trắng cũng tính được một cách tương tự như trên:

$$p(0,k,l) = p(t)p(t/t)^{k-1} [p(t/t)p(t/X=t, Y=t)^{l-1}]^{k-1} \quad (8.4)$$

Mỗi quan hệ này tương đương:

$$p(0,k,l) = p(t)p(t/t)^{k+l-2}p(t/X=t, Y=t)^{l-1-k+l-1} \quad (8.5)$$

và tỷ số nền sẽ cho bởi công thức:

$$C_r = \frac{1}{[1-p(t)]p(t/t)^{k+l-2}+1/k/l} \quad (8.6)$$

Thực tế, khi cài đặt người ta hay chọn khối vuông và giá trị thích hợp của k từ 4 đến 5.

#### 8.2.5. Phương pháp thích nghi

Thuật ngữ "thích nghi" thường dùng để chỉ sự thích hợp của các từ mã theo một nghĩa nào đấy. Như trong phương pháp RLC ở trên, thay vì dùng chiều dài từ mã cố định m bit, người ta dùng chiều dài biến đổi và trên cơ sở đó có phương pháp RLC thích nghi.

Trong phương pháp mã hoá khối, người ta dùng chiều dài khối cố định gồm k x l điểm ảnh. Tuy nhiên, với ảnh không thuần nhất, phương pháp mã hoá này bộc lộ nhiều nhược điểm. Vì rằng, với ảnh không thuần nhất, chính sự không thuần nhất của ảnh quyết định sự thích nghi với điều kiện cục bộ.

Một cải tiến cho vấn đề này là cố định một kích thước của khối, còn kích thước kia coi như là hàm của một tác động trung bình theo hàng (với  $l=1$ ) hay theo một nhóm hàng ( $l > 1$ ). Tác động được quan tâm cũng giống như các phương pháp trên là sự dịch chuyển các điểm trắng sang đen trên hàng. Một cách lý thuyết, người ta cũng tính được giá trị tối ưu của k ( $k_{opt}$ ):

$$k_{opt} = \begin{cases} \sqrt{\frac{N}{T}} & l=1 \text{ và } N \text{ là số điểm ảnh trên hàng} \\ \sqrt{l} & l > 1 \end{cases} \quad (8.7)$$

Trên cơ sở này, người ta áp dụng mã hóa khối tự động thích nghi cho một số ứng dụng [6]:

- Mã đoạn hay khối k x l tự động thích nghi với tác động cục bộ.
- Mã đoạn hay khối k x l tự động thích nghi 1 chiều.
- Mã khối k x l tự động thích nghi 2 chiều.

### 8.3. PHƯƠNG PHÁP MÃ HOÁ DỰA VÀO BIẾN ĐỔI THẾ HỆ THỨ NHẤT

Tuy rằng bản chất của các phương pháp nén dựa vào biến đổi rất khác với các phương pháp đã trình bày ở trên, song theo định nghĩa phân loại nén, nó vẫn được xếp vào họ thứ nhất. Vì có các đặc thù riêng nên chúng ta xếp riêng trong phần này.

#### 8.3.1. Nguyên tắc chung

Như trong 8.1.3, các phương pháp mã hóa dựa vào biến đổi làm giảm lượng thông tin dư thừa không tác động lên miền không gian của ảnh số mà tác động lên miền biến đổi. Các biến đổi được dùng ở đây là các biến đổi tuyến tính như: biến đổi KL, biến đổi Fourier, biến đổi Hadamard, Sin, Cosin, v,...,v.

Vì ảnh số thường có kích thước rất lớn, nên trong cài đặt người ta thường chia ảnh thành các khối chữ nhật nhỏ như đã nói trong 8.2.3. Thực tế, người ta dùng khối vuông kích thước cỡ 16 x 16. Sau đó tiến hành biến đổi từng khối một cách độc lập.

Chúng ta đã biết (xem chương Ba), dạng chung của biến đổi tuyến tính 2 chiều là:

$$X(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a(m,n,k,l)x(k,l) \quad (8.8)$$

với:

- $x(k,l)$  là tín hiệu vào
- $a(m,n,k,l)$  là các hệ số của biến đổi - là phần tử của ma trận biến đổi A.

Ma trận này gọi là *nhân* của biến đổi. Cách xác định các hệ số này là phụ thuộc vào từng loại biến đổi sử dụng. Đối với phần lớn các biến đổi 2 chiều, nhân có tính đối xứng và tách được:

$$A[m,n,k,l] = A'[m,k] A''[n,l]$$

Như đã chỉ ra trong 3.2.3, nếu biến đổi là KL thì các hệ số đó chính là các phần tử của véc tơ riêng.

### 8.3.2. Thuật toán mã hóa dùng biến đổi 2 chiều

Các phương pháp mã hóa dùng biến đổi 2 chiều thường gồm 4 bước sau:

#### b1) Chia ảnh thành khối

Ảnh được chia thành các khối nhỏ kích thước k x l và biến đổi các khối đó một cách độc lập để thu được các khối  $V_i$ ,  $i=0,1,\dots,B$  với  $B = N \times M / (k \times l)$ .

#### b2) Xác định phân phối bit cho từng khối

Thông thường các hệ số hiệp biến của các biến đổi là khác nhau. Mỗi hệ số yêu cầu lượng hoá với một số lượng bit khác nhau.

#### b3) Thiết kế bộ lượng hoá

Với phần lớn các biến đổi, các hệ số  $v(0,0)$  là không âm. Các hệ số còn lại có trung bình 0. Để tính các hệ số, ta có thể dùng phân bố Gauss hay Laplace. Các hệ số được mã hoá bởi số bit khác nhau, thường từ 1 đến 8 bit. Do vậy cần thiết kế 8 bộ lượng hoá. Để dễ cài đặt, tín hiệu vào  $v_i(k,l)$  được chuẩn hoá để có dạng:

$$v_i(k,l) = v_i(k,l)/\sigma_{v_i} \quad (k,l) \neq (0,0) \quad (8.9)$$

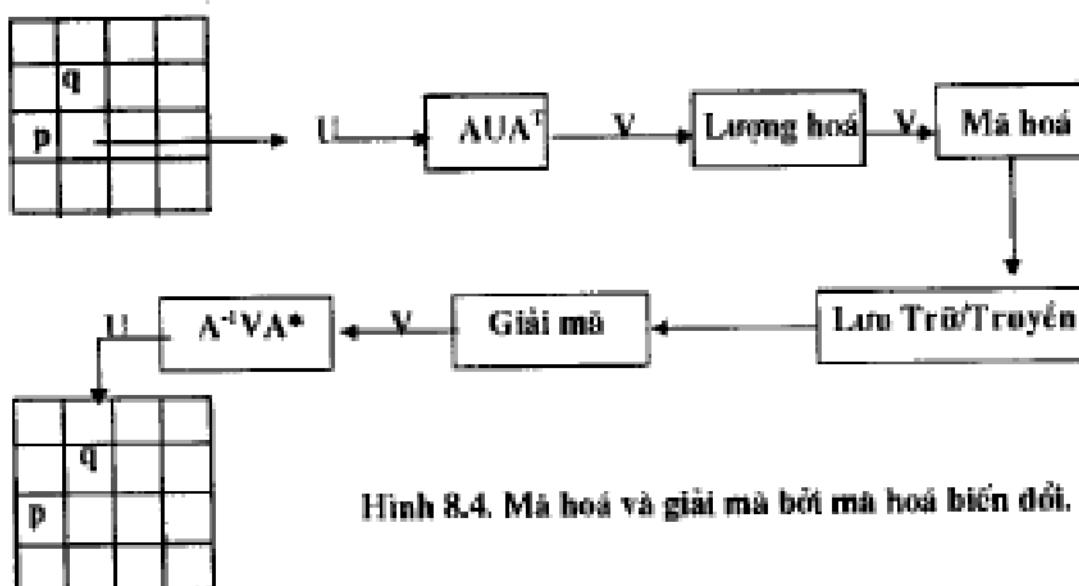
Trước khi thiết kế bộ lượng hoá, người ta tìm cách loại bỏ một số hệ số không cần thiết.

#### b4) Mã hoá

Tín hiệu đầu ra của bộ lượng hoá sẽ được mã hoá trên các từ bit để truyền đi hay lưu trữ lại. Quá trình mã hoá dựa vào biến đổi có thể được tóm tắt trên hình 8.4 dưới đây.

Nếu ta chọn phép biến đổi KL cho phương pháp sẽ có một số nhược điểm; khối lượng tính toán sẽ rất lớn vì phải tính ma trận hiệp biến, tiếp sau là phải giải phương trình tìm trị riêng và véc tơ riêng để xác định các hệ số. Vì lý do này, trên thực tế người ta thích

dùng các biến đổi khác như Hadamard, Haar, Sin và Cosin. Trong số biến đổi này, biến đổi Cosin thường hay được dùng hơn.



Hình 8.4. Mã hoá và giải mã bởi mã hoá biến đổi.

### 8.3.3. Mã hoá dùng biến đổi Cosin và chuẩn JPEG

#### 8.3.3.1. Phép biến đổi Cosine một chiều

Phép biến đổi Cosin rời rạc (DCT) được Ahmed đưa ra vào năm 1974. Kể từ đó đến nay nó được ứng dụng rất rộng rãi trong nhiều phương thức mã hoá ảnh khác nhau nhờ hiệu suất gần như tối ưu của nó đối với các ảnh có độ tương quan cao giữa các điểm ảnh lân cận. Biến đổi Cosin rời rạc được sử dụng trong chuẩn ảnh nén JPEG và định dạng phim MPEG.

##### Phép biến đổi Cosine một chiều

Phép biến đổi Cosin rời rạc một chiều được định nghĩa bởi:

$$X(k) = \frac{2\varepsilon_1}{N} \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad (8.10)$$

Trong đó:  $\varepsilon_1 = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k = 0 \\ 0 & \text{khi } k = [1, N-1] \end{cases}$

Khi dãy đầu vào  $x(n)$  là thực thì dãy các hệ số  $X(k)$  cũng là số thực. Tính toán trên trường số thực giảm đi một nửa thời gian so với biến đổi Fourier. Để đạt được tốc độ biến đổi thoả mãn yêu cầu của các ứng dụng thực tế, người ta đã cải tiến kỹ thuật tính toán và đưa ra nhiều thuật toán biến đổi nhanh Cosine. Một trong những thuật toán đó được giới thiệu dưới đây.

### Phép biến đổi Cosin nhanh

Phép biến đổi Cosin nhanh viết tắt là FCT (Fast Cosine Transform), dựa vào ý tưởng đưa bài toán ban đầu về tổ hợp của các bài toán biến đổi FFT trên các dãy con. Việc tiến hành biến đổi trên các dãy con sẽ đơn giản hơn rất nhiều so với dãy gốc. Vì thế, người ta tiếp tục phân nhỏ dãy tín hiệu đến khi chỉ còn một phần tử.

Giải thuật biến đổi Cosin nhanh không thực hiện trực tiếp trên dãy tín hiệu đầu vào  $x(n)$  mà thực hiện trên dãy  $x'(n)$  là một hoán vị của  $x(n)$ . Giải thiết số điểm cần tính FCT là luỹ thừa của 2:  $N = 2^M$ .

Dữ liệu vào  $x(n)$  sẽ được sắp xếp lại như sau:

$$x'(i) = x(2i) \text{ với } i = 0, 1, \dots, \frac{N}{2} - 1$$

$$x'(N-i-1) = x(2i+1) \text{ với } i = 0, 1, \dots, \frac{N}{2} - 1$$

Như vậy nửa đầu dãy  $x'(n)$  là các phần tử chỉ số chẵn của  $x(n)$  xếp theo chiều tăng dần của chỉ số. Nửa sau của  $x'(n)$  là các phần tử chỉ số lẻ của  $x(n)$  xếp theo chiều giảm dần của chỉ số.

Thay vào công thức (8.10) ta được:

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)\cos\frac{\pi(4n+1)k}{2N} + \sum_{n=0}^{N/2-1} x(2n+1)\cos\frac{\pi(4n+3)k}{2N}$$

Rút gọn biểu thức trên được:

$$X(k) = \sum_{n=0}^{N/2-1} x'(n)\cos\frac{\pi(4n+1)k}{2N}$$

Chia  $X(k)$  ra làm hai hai dãy, một dãy bao gồm các chỉ số chẵn, còn dãy kia bao gồm các chỉ số lẻ.

#### Phần chỉ số chẵn

$$X(2k) = \sum_{n=0}^{N/2-1} \left[ x'(n) + x'\left(n + \frac{N}{2}\right) \right] \cos\frac{\pi(4n+1)2k}{2(\frac{N}{2})}$$

Có thể chuyển về dạng:

$$X(2k) = \sum_{n=0}^{N/2-1} x'(n)\cos\frac{\pi(4n+1)k}{2N} \quad (8.11)$$

### Phản chí số lẻ

$$X(2k+1) = \sum_{n=0}^{N-1} \left[ x'(n) + x'\left(n + \frac{N}{2}\right) \right] \cos \frac{\pi(4n+1)}{2N}(2k+1)$$

Có thể biểu diễn dưới dạng:

$$X(2k+1) = \sum_{n=0}^{N-1} \left\{ \left[ x'(n) - x'\left(n + \frac{N}{2}\right) \right] 2 \cos \frac{\pi(4n+1)}{2N} \right\} \cos \left[ \frac{\pi(4n+1)}{2N} k \right] (2k+1) \quad (8.12)$$

$$\text{Ta có: } \cos[(2k+1)\Phi] = 2\cos\Phi\cos(2k\Phi) - \cos[(2k-1)\Phi]$$

Do vậy (8.12) trở thành:

$$X(2k+1) = \sum_{n=0}^{N-1} \left\{ \left[ x'(n) - x'\left(n + \frac{N}{2}\right) \right] 2 \cos \frac{\pi(4n+1)}{2N} \right\} \cos \left[ \frac{\pi(4n+1)}{2N} k \right] - X(2k-1) \quad (8.13)$$

$$C_j^i = \cos \frac{\pi(4i+1)}{2j}$$

$$h(n) = \left[ x'(n) - x'\left(n + \frac{N}{2}\right) \right] 2 \cos \frac{\pi(4n+1)}{2N}$$

$$\text{Đặt: } G(k) = X(2k)$$

$$g(n) = x'(n) + x'\left(n + \frac{N}{2}\right)$$

$$H(k) = X(2k+1)$$

Ta thu được:

$$H(k) = \sum_{n=0}^{N-1} h(n) \cos \frac{\pi(4n+1)}{2N} \quad (8.14)$$

$$G(k) = \sum_{n=0}^{N-1} g(n) \cos \frac{\pi(4n+1)k}{2N} \quad (8.15)$$

Có thể nhận ra ngay các công thức (8.14) (8.15) là phép biến đổi Cosin N/2 điểm của  $g(n)$  và  $h(n)$ . Như vậy bài toán biến đổi Cosine của dãy  $x'(n)$  đã được đưa về hai bài toán biến đổi Cosine của hai dãy con là  $g(n)$  và  $h(n)$  có kích thước bằng một nửa  $x'(n)$ . Hai dãy  $g(n)$  và  $h(n)$  tính toán được một cách dễ dàng,  $g(n)$  là tổng của nửa đầu dãy  $x'(n)$  với nửa sau của nó,  $h(n)$  là hiệu của nửa đầu dãy  $x'(n)$  với nửa sau của nó sau đó đem nhân với  $2C_n^2$ . Ta lặp lại quá trình chia đôi đối với các dãy con, dãy con của dãy con và cứ tiếp tục chia như thế. Giống như biến đổi Fourier, mỗi bước lặp cũng được coi là một tầng phân chia. Với  $N = 2^M$  thì số tầng phân chia là  $M$ .

Để dễ hình dung, đầu ra của mỗi tầng được kí hiệu là  $X_m(n)$  với  $m$  là tầng hiện thời. Ta xem  $x'(n)$  là biến đổi Cosin 0 tầng của  $x'(n)$ :

$$X_0(n) = x'(n)$$

$X_M(n)$  là biến đổi Cosin tầng  $M$  của  $x(n)$ , nó không phải là  $X(k)$ . Bởi vì cứ sau mỗi tầng, không chỉ thứ tự các phần tử trong  $X(k)$  bị xáo trộn mà các  $X(2k+1)$  còn được cộng với  $X(2k-1)$ . Đầu ra của một tầng là đầu vào của tầng tiếp theo.

$$X_i(n) = g(n) \quad \text{với } n = 0, 1, \dots, \frac{N}{2} - 1$$

$$X_i(n + \frac{N}{2}) = h(n) \quad \text{với } n = 0, 1, \dots, \frac{N}{2} - 1$$

Từ công thức tính  $g(n)$  và  $h(n)$  ta có:

$$X_i(i) = X_0(i) + X_0(i + \frac{N}{2})$$

$$X_i(i + \frac{N}{2}) = \left[ X_0(i) - X_0(i + \frac{N}{2}) \right] 2C_n^i$$

với  $n = 0, 1, \dots, \frac{N}{2} - 1$

Cứ sau mỗi tầng, số dãy con lại được nhân đôi. Xét phép biến đổi tại tầng thứ  $m$ , chúng ta phải lặp lại công việc biến đổi cho  $2^{m-1}$  dãy con. Mỗi dãy con đóng vai trò như dãy  $x'(n)$  trong tầng thứ nhất. Số phần tử trong một dãy là:  $\frac{N}{2^{m-1}}$ . Công đoạn biến đổi trên một dãy con gọi là một khối biến đổi. Mỗi dãy con sẽ tiếp tục được phân làm hai dãy nhỏ hơn. Công thức tổng quát tại mỗi khối là:

$$X_n(i) = X_{m-1}(i) + X_{m-1}(i + \frac{N}{2^m}) \quad (8.16)$$

$$X_m(i + \frac{N}{2^n})[X_m - l(i) - X_m - l(i + \frac{N}{2^{n-1}})]2C'N/2^{n-1}$$

Với  $i = k\frac{N}{2^{n-1}}, \dots, k\frac{N}{2^{n-1}} + \frac{N}{2^n}$ , trong đó  $k = 0, 1, \dots, 2^n - 1$

Phản xạ dụng công thức tổng quát trong phép biến đổi nhanh Fourier được trình bày khá chi tiết ở trên, chúng ta có thể xem lại phản này để hiểu hơn về công thức tổng quát cho một khối biến đổi nhanh Cosin.

Thuật toán biến đổi nhanh Cosin có thể mô tả bằng các bước sau:

**Bước 1:** Tính dãy hệ số  $C_j^i$ .

Xác định số tầng  $M = \log_2 N$ .

Tăng hiện thời  $m=1$ .

**Bước 2:** Nếu  $m \leq M$  thực hiện bước 3. Nếu không kết thúc.

(*Chưa hết các tầng*)

**Bước 3:** Khởi hiện thời  $k = 0$ .

**Bước 4:** Nếu  $k < 2^{m-1}$  Thực hiện bước 5. Nếu không thực hiện bước 6.

(*Chưa hết các khối trong một tầng*)

**Bước 5:** Tính toán  $X_m(i)$  trong khối theo công thức tổng quát (8.16), (8.17).

Tăng  $k$  lên 1. Quay về bước 4.

(*Chuyển đến khối tiếp theo*)

**Bước 6:** Tăng  $m$  lên 1. Quay về bước 2

(*Chuyển đến tầng tiếp theo*)

*Một số vấn đề lưu ý khi cài đặt thuật toán biến đổi Cosin nhanh*

Khác với biến đổi Fourier nhanh, trong biến đổi Cosin,  $x(n)$  không phải đầu vào trực tiếp và  $X(k)$  không phải là đầu ra trực tiếp. Ở đầu vào,  $x'(n)$  chỉ là cách sắp xếp lại  $x(n)$ . Chúng ta biết rằng tại mỗi tầng, đối với mỗi khối:

$$X(2i+1) = X(2i+1) + X(2i-1)$$

Nên ở đầu ra, sau khi tính được  $X_m(n)$  chúng ta phải thực hiện việc trả truy hồi từ tầng  $M$  về tầng 1 sau đó hoàn vị lại theo thứ tự đảo bit mới thu được hệ số biến đổi  $X(k)$  cần tính.

Bài toán sắp xếp lại theo thứ tự đảo bit đã đề cập trong phần biến đổi Fourier. Bài toán trừ truy hồi cài đặt khá đơn giản.

Dãy hệ số  $C_j^i$  được tính trước một lần. Trong các ứng dụng mà số điểm tính FFT không đổi hoặc chỉ nhận một số giá trị cụ thể, người ta thường tính trước  $C_j^i$  và ghi ra file. Khi thực hiện biến đổi thì đọc từ file để lấy thông tin này. Trong ứng dụng của chúng ta, ta tính trước  $C_j^i$  và lưu vào một mảng. Phép biến đổi sẽ truy cập bảng này để lấy hệ số cần thiết.

#### *Phép biến đổi Cosin ngược*

Phép biến đổi Cosin ngược được định nghĩa bằng công thức:

$$x(n) = \sum_{k=0}^{N-1} X(k) c_k \cos \frac{\pi k(2n+1)}{2N} \quad (8.18)$$

Với  $c_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k = 0 \\ 0 & \text{khi } k \neq 0 \end{cases}$

Phép biến đổi Cosin ngược sẽ được thực hiện theo chiều ngược lại với quy trình đã tiến hành trong phép biến đổi nhanh. Tuy nhiên, công việc này không được thuận lợi như phép biến đổi FFT ngược. Từ  $X(k)$  chúng ta phải khôi phục lại  $X_m(n)$  bằng cách thực hiện các phép cộng truy hồi và phép hoán vị theo thứ tự đảo bit. Công thức tổng quát cho mỗi khôi biến đổi ngược được xây dựng dựa trên công thức tổng quát trong biến đổi xuôi:

Với  $i = k \frac{N}{2^{m-1}}, \dots, k \frac{N}{2^{m-1}} + \frac{N}{2^m}$ , trong đó  $k = 0, 1, \dots, 2^m - 1$

$$X_{m-1}(i) = \frac{1}{2} X_m(i) + X_m(i + \frac{N}{2^m}) \frac{1}{2C_{\frac{N}{2^{m-1}}}} \quad (8.19)$$

$$X_{m-1}(i + \frac{N}{2^m}) = \frac{1}{2} X_m(i) - X_m(i + \frac{N}{2^m}) \frac{1}{2C_{\frac{N}{2^{m-1}}}} \quad (8.20)$$

Phép biến đổi ngược phải cài đặt riêng. Tuy vậy, tư tưởng chính của hai bài toán xuôi và ngược về cơ bản giống nhau. Đầu ra của phép biến đổi ngược sẽ là  $x'(n)$ . Muốn thu được  $x(n)$  ta phải đảo lại vị trí.

### 8.3.3.2. Phép biến đổi Cosin rời rạc hai chiều

Phép biến đổi Cosin rời rạc hai chiều được định nghĩa bởi:

$$X(k_1, k_2) = \frac{4\epsilon_{k_1} \epsilon_{k_2}}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cos \frac{\pi(2n_1+1)k_1}{2N_1} \cos \frac{\pi(2n_2+1)k_2}{2N_2}, \quad (8.21)$$

Trong đó,  $\epsilon_{k_1} = 0$  khi  $k_1 = 0$  và  $\epsilon_{k_1} = \frac{1}{\sqrt{2}}$  khi  $k_1 = 1, 2, \dots, N_1-1$

$$\epsilon_{k_2} = 0 \text{ khi } k_2 = 0 \text{ và } \epsilon_{k_2} = \frac{1}{\sqrt{2}} \text{ khi } k_2 = 1, 2, \dots, N_2-1$$

Phép biến đổi ngược được định nghĩa bởi công thức:

$$x(n_1, n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) \epsilon_{k_1} \epsilon_{k_2} \cos \frac{\pi(2n_1+1)k_1}{2N_1} \cos \frac{\pi(2n_2+1)k_2}{2N_2}, \quad (8.22)$$

$\epsilon_{k_1}, \epsilon_{k_2}$  nhận các giá trị như trong công thức biến đổi xuôi.

Để nâng cao tốc độ biến đổi người ta đã phát triển các giải thuật biến đổi nhanh Cosin hai chiều. Cách làm phổ biến nhất là tận dụng phép biến đổi nhanh Cosin một chiều. Ta biến đổi công thức (8.21) về dạng:

$$X(k_1, k_2) = \frac{2\epsilon_{k_1}}{N_1} \sum_{n_1=0}^{N_1-1} \left[ \frac{2\epsilon_{k_2}}{N_2} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cos \frac{\pi(2n_2+1)k_2}{2N_2} \right] \cos \frac{\pi(2n_1+1)k_1}{2N_1}, \quad (8.23)$$

Đặt:

$$X'(n_1, k_2) = \frac{2\epsilon_{k_2}}{N_2} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \cos \frac{\pi(2n_2+1)k_2}{2N_2} \quad (8.24)$$

Công thức (8.23) trở thành:

$$X(k_1, k_2) = \frac{2\epsilon_{k_1}}{N_1} \sum_{n_1=0}^{N_1-1} [X'(n_1, k_2)] \cos \frac{\pi(2n_1+1)k_1}{2N_1} \quad (8.25)$$

Công thức (8.24) là phép biến đổi Cosin rời rạc một chiều của  $x(n_1, n_2)$ , trong đó  $n_2$  là biến số, còn  $n_1$  đóng vai trò là tham số thu được kết quả trung gian  $X'(n_1, k_2)$ . Công thức (8.25) là phép biến đổi Cosin rời rạc của  $X'(n_1, k_2)$  với  $n_1$  là biến số còn  $k_2$  là tham số. Đến đây tư tưởng của thuật toán đã rõ ràng. Khi biến đổi nhanh Cosin hai chiều của một ma trận

ảnh, ta sẽ tiến hành biến đổi nhanh một chiều trên các điểm ảnh theo hàng, sau đó đem biến đổi nhanh một chiều theo cột của kết quả vừa thu được.

Biến đổi nhanh Cosin ngược hai chiều cũng được xây dựng dựa trên kết quả phép biến đổi nhanh Cosin ngược một chiều. Từ công thức (8.22) ta biểu diễn lại như sau:

$$x(n_1, n_2) = \sum_{k_1=0}^{N_1-1} \left[ \sum_{k_2=0}^{n_2-1} X(k_1, k_2) e_{k_2} \cos \frac{\pi(2n_1+1)k_1}{2N_1} \right] e_{n_2} \cos \frac{\pi(2n_2+1)k_2}{2N_2} \quad (8.26)$$

Đặt:

$$x'(k_1, n_2) = \sum_{k_2=0}^{n_2-1} [x'(k_1, k_2)] e_{k_2} \cos \frac{\pi(2n_2+1)k_2}{2N_2} \quad (8.27)$$

Khi đó công thức (8.26) sẽ trở thành:

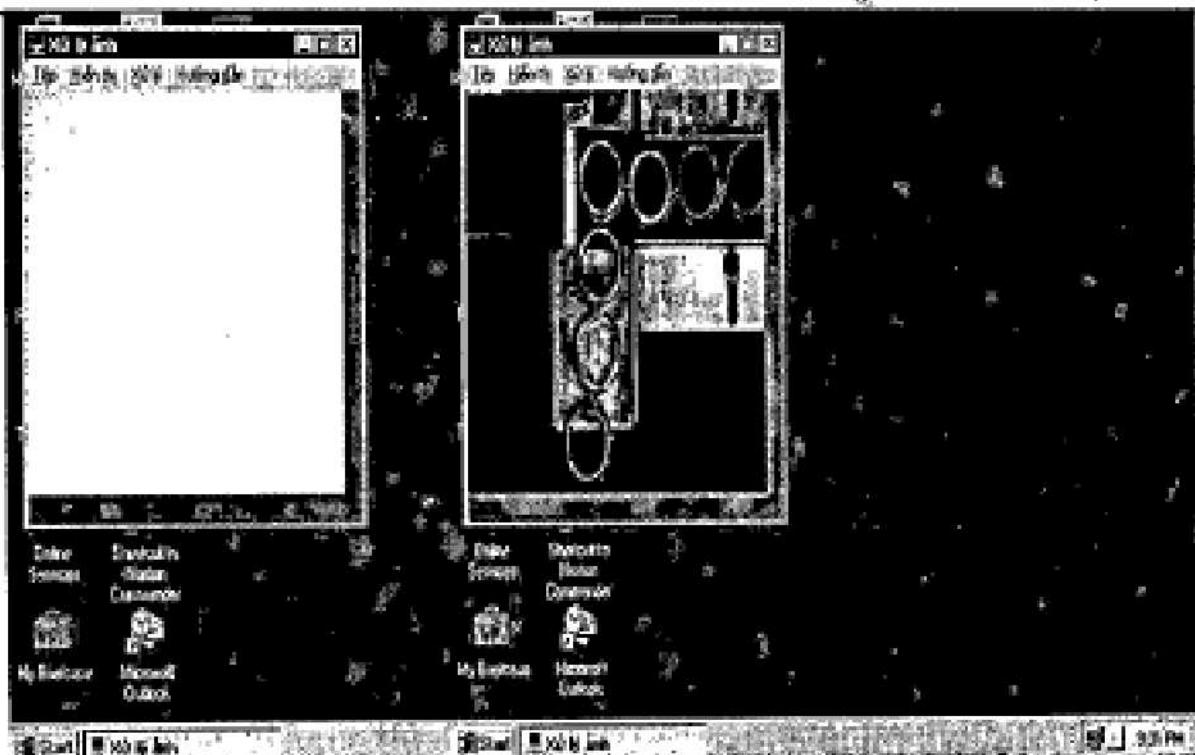
$$x'(k_1, n_2) = \sum_{k_2=0}^{n_2-1} X(k_1, k_2) e_{k_2} \cos \frac{\pi(2n_2+1)k_2}{2N_2} \quad (8.28)$$

Công thức (8.27) là phép biến đổi Cosin ngược rời rạc một chiều của  $X(k_1, k_2)$ , trong đó  $k_2$  là biến số, còn  $k_1$  đóng vai trò là tham số thu được kết quả trung gian  $x'(k_1, n_2)$ . Công thức (8.28) là phép biến đổi Cosin ngược rời rạc của  $x'(k_1, n_2)$  với  $k_1$  là biến số còn  $n_2$  là tham số. Như vậy, muốn khôi phục lại ảnh ban đầu từ ma trận hệ số biến đổi chúng ta sẽ biến đổi nhanh Cosin ngược rời rạc một chiều các hệ số theo hàng, sau đó đem biến đổi nhanh Cosin rời rạc một chiều theo cột các kết quả trung gian vừa tính được.

### 8.3.3.3. Biến đổi Cosin và chuẩn nén JPEG

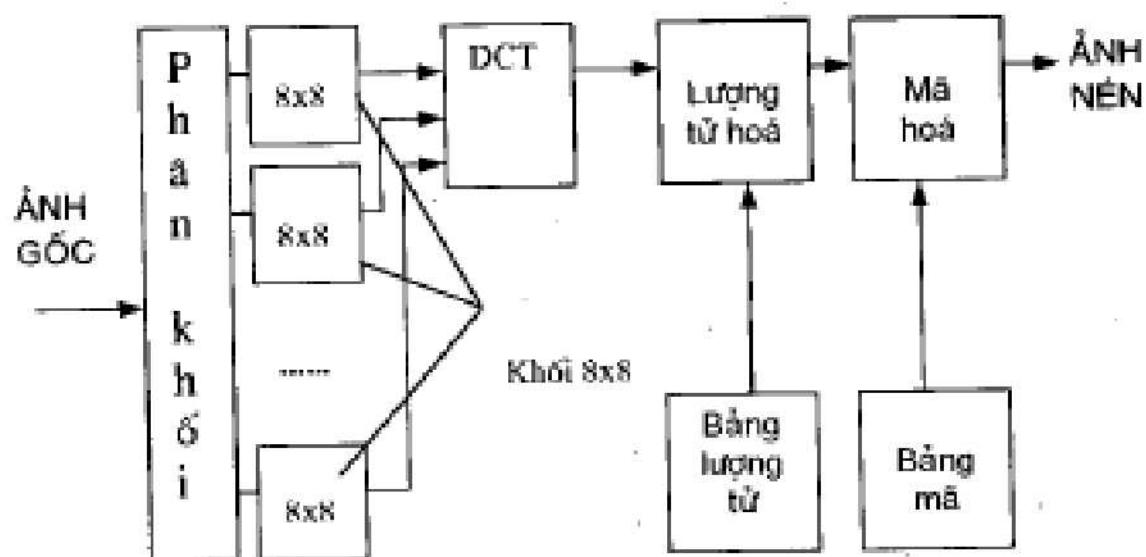
JPEG là viết tắt của *Joint Photographic Expert Group* (nhóm các chuyên gia phát triển chuẩn ảnh này). Chuẩn JPEG được công nhận là chuẩn ảnh quốc tế năm 1990 phục vụ các ứng dụng truyền ảnh cho các lĩnh vực như y học, khoa học-kỹ thuật, ảnh nghệ thuật...

Chuẩn JPEG được sử dụng để mã hoá ảnh đa mức xám, ảnh màu. Nó không cho trình bày chi tiết về một trong các dạng nén biến đổi chấp nhận mất mát thông tin dùng biến đổi Cosin của chuẩn JPEG: Biến đổi Cosin tuần tự (Sequential DCT-Based). Biến đổi Cosin tuần tự là kỹ thuật đơn giản nhất nhưng được dùng phổ biến nhất và nó đáp ứng được hầu hết các đặc tính cần thiết cho phần lớn các ứng dụng.

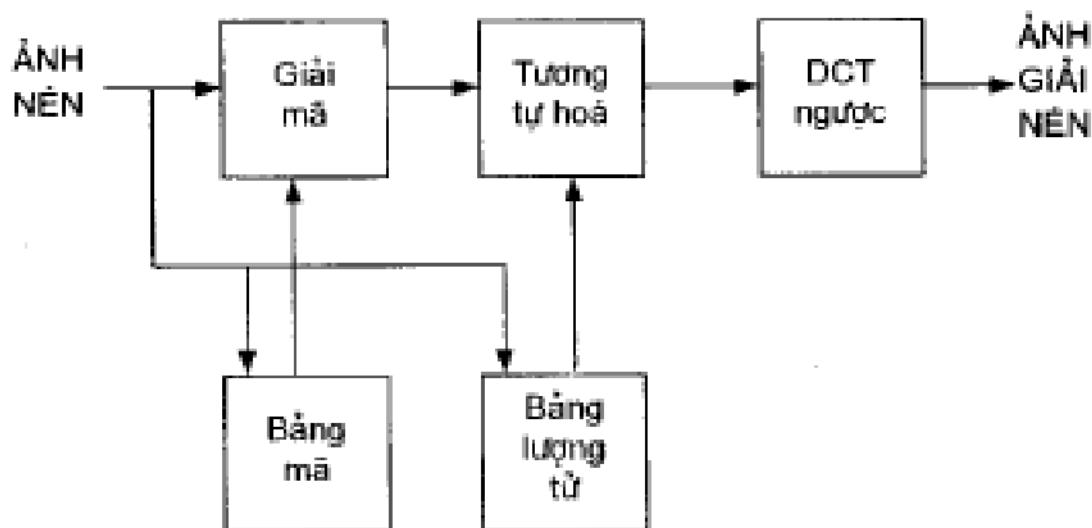


Hình 8.5. Biến đổi Cosin của ảnh (trái) và biến đổi ngược (ảnh gốc - phải)

Mã hóa JPEG bao gồm nhiều công đoạn như đã nêu trong 8.3.3.1. Sơ đồ thuật toán nén và giải nén được mô tả như dưới đây.



Quá trình giải nén sẽ được làm ngược lại, người ta giải mã từng phần ảnh nén tương ứng với phương pháp nén đã sử dụng trong phần nén nhờ các thông tin liên quan ghi trong phần header của file nén. Kết quả thu được là hệ số đú lượng tử. Các hệ số này được khôi phục về giá trị trước khi lượng tử hoá bằng bộ tương tự hoà. Tiếp đó đem biến đổi Cosin ngược ta được ảnh ban đầu với độ trung thực nhất định.



Bảng mã và bảng lượng tử trong sơ đồ giải nén được dựng lên nhờ những thông tin ghi trong phần cấu trúc đầu tệp (Header) của tệp ảnh nén. Quá trình nén chịu trách nhiệm tạo ra và ghi lại những thông tin này. Phần tiếp theo sẽ phân tích tác dụng của từng khối trong sơ đồ.

## A. PHẦN KHỐI

Chuẩn nén JPEG phân ảnh ra các khối 8x8. Công đoạn biến đổi nhanh Cosin hai chiều cho các khối 8x8 tỏ ra hiệu quả hơn. Biến đổi Cosin cho các khối có cùng kích cỡ có thể giảm được một phần các tính toán chung như việc tính hệ số  $C_j^i$ . Khi  $n=8$  chúng ta chỉ cần tính hệ số  $C_j^i$  cho 3 tầng ( $\theta = 2^3$ ), số các hệ số là:  $4 + 2 + 1 = 7$

Nếu với một ảnh 1024 x 1024, phép biến đổi nhanh Cosin một chiều theo hàng ngang hoặc hàng dọc ta phải qua 10 tầng ( $1024 = 2^{10}$ ). Số các hệ số  $C_j^i$  là:  $512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 1021$ . Thời gian tính các hệ số  $C_j^i$  với toàn bộ ảnh 1024x1024 lớn gấp 150 lần so với thời gian tính toán các hệ số này cho các khối.

Biến đổi Cosin đối với các khối có kích thước nhỏ sẽ làm tăng độ chính xác khi tính

tuần với số dấu phẩy tĩnh, giảm thiểu sai số do làm tròn sinh ra.

Do các điểm ảnh hàng xóm có độ tương quan cao hơn, do đó phép biến đổi Cosin cho từng khối nhỏ sẽ tập trung năng lượng hơn vào một số ít các hệ số biến đổi. Việc loại bỏ một số hệ số năng lượng thấp trong các khối chỉ tạo ra mất mát thông tin cục bộ giúp nâng cao chất lượng ảnh.

Ảnh sẽ được chia làm B khối với:

$$B = \left( \frac{M}{k} \right) \times \left( \frac{N}{l} \right) = M_B \times N_B$$

Các khối được xác định bởi bộ số  $(m,n)$  với  $m = [0..M_B - 1]$  và  $n = [0..N_B - 1]$ , ở đây m chỉ thứ tự của khối theo chiều rộng, n chỉ thứ tự của khối theo chiều dài. Phân khối thực chất là xác định tương quan giữa tọa độ riêng trong khối với tọa độ thực của điểm ảnh trong ảnh ban đầu. Nếu ảnh ban đầu ký hiệu  $\text{Image}[i,j]$  thì ma trận biểu diễn khối  $(m,n)$  là  $x[u,v]$  được tính:

$$x[u,v] = \text{Image}[mk + u, nl + v]$$

## B. BIẾN ĐỔI

Biến đổi là một công đoạn lớn trong các phương pháp nén sử dụng phép biến đổi. Nhiệm vụ của công đoạn biến đổi là tập trung năng lượng vào một số ít các hệ số biến đổi. Công thức biến đổi cho mỗi khối là:

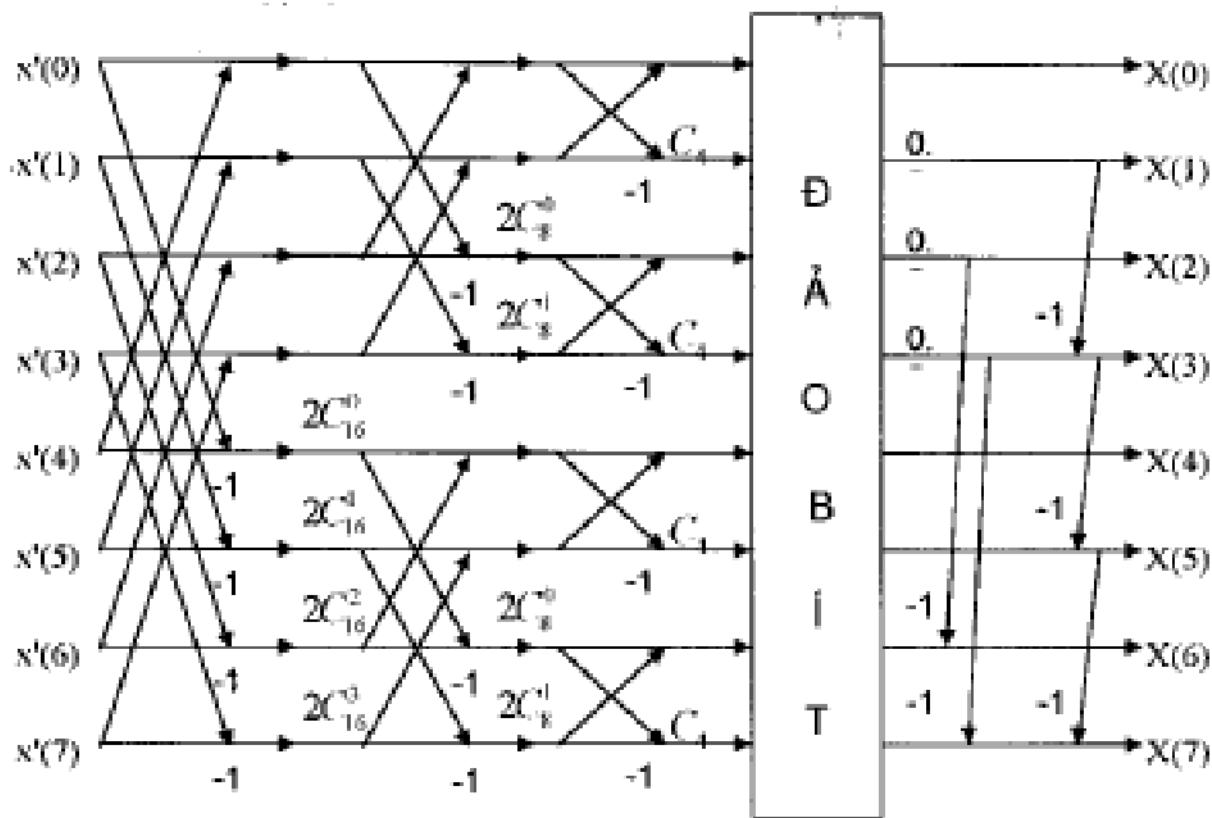
$$X(k_1, k_2) = \frac{\epsilon_{k_1} \epsilon_{k_2}}{4} \sum_{n_1=0}^7 \sum_{n_2=0}^7 x(n_1, n_2) \cos \frac{(2n_1+1)k_1 \pi}{16} \cos \frac{(2n_2+1)k_2 \pi}{16} \quad (8.29)$$

Trong đó  $\epsilon_{k_1} = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k_1 = 0 \\ 0 & \text{khi } (0 < k_1 < 8) \end{cases}$

$$\epsilon_{k_2} = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k_2 = 0 \\ 0 & \text{khi } (0 < k_2 < 8) \end{cases}$$

Thuật toán biến đổi nhanh Cosin hai chiều cho mỗi khối trong trường hợp này sẽ bao gồm 16 phép biến đổi nhanh Cosin một chiều. Đầu tiên, người ta biến đổi nhanh Cosin một chiều cho các dãy điểm ảnh trên mỗi hàng. Lần lượt thực hiện cho 8 hàng. Sau đó đem biến đổi nhanh Cosin một chiều theo từng cột của ma trận vừa thu được sau 8 phép biến đổi trên.

Cũng lần lượt thực hiện cho 8 cột. Ma trận cuối cùng sẽ là ma trận hệ số biến đổi của khối lượng ứng. Giải thuật biến đổi nhanh được mô tả như hình sau:



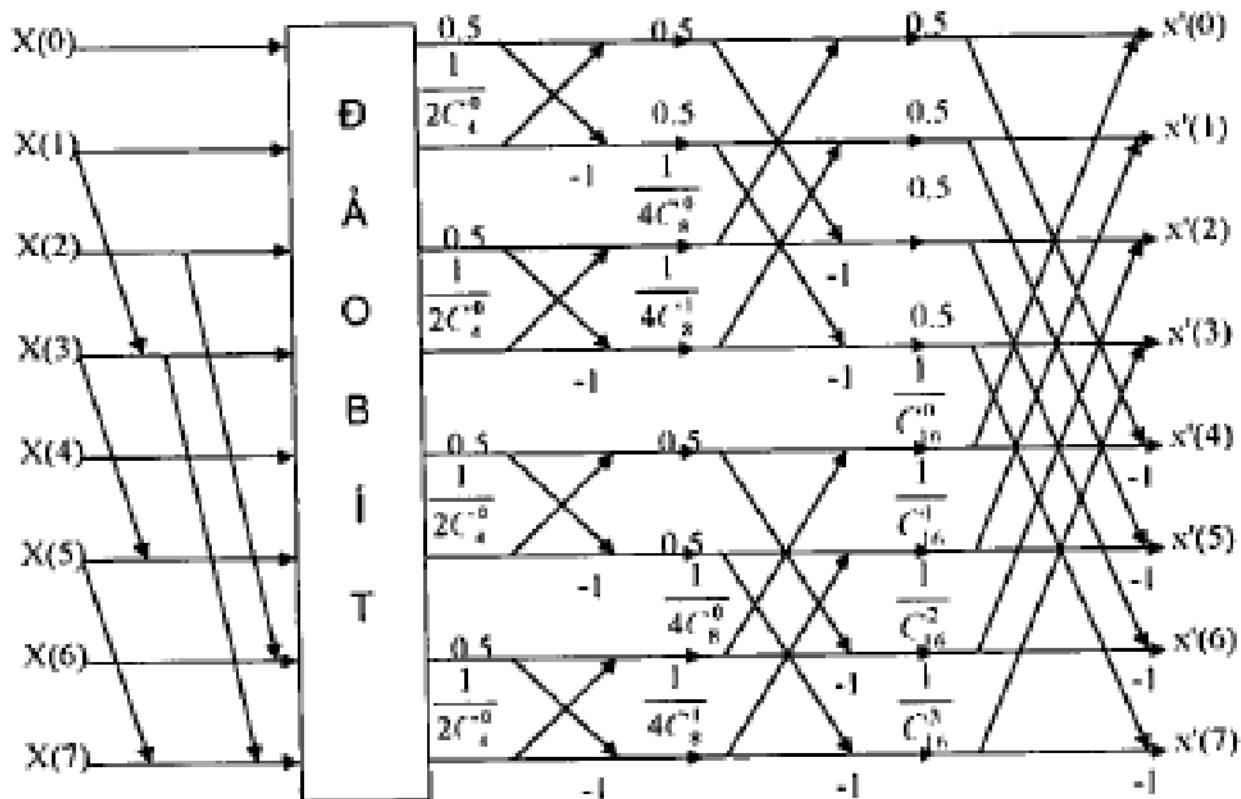
Trong sơ đồ giải nén ta phải dùng phép biến đổi Cosin ngược. Công thức biến đổi ngược cho khối 8x8:

$$x(n_1, n_2) = \frac{E_{k_1} E_{k_2}}{4} \sum_{l=0}^7 \sum_{k_1=0}^7 X(k_1, k_2) \cos \frac{(2n_1+1)k_1\pi}{16} \cos \frac{(2n_2+1)k_2\pi}{16} \quad (8.30)$$

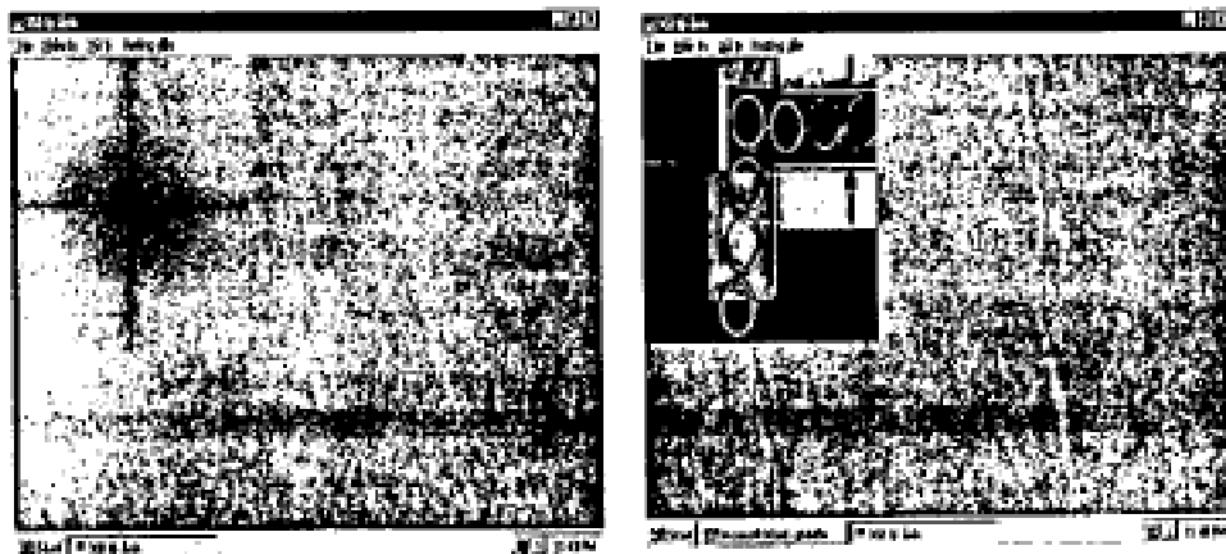
Trong đó  $E_{k_1} = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k_1 = 0 \\ 0 & \text{khi } (0 < k_1 < 8) \end{cases}$

$$E_{k_2} = \begin{cases} \frac{1}{\sqrt{2}} & \text{khi } k_2 = 0 \\ 0 & \text{khi } (0 < k_2 < 8) \end{cases}$$

Sơ đồ biến đổi ngược nhanh được biểu diễn như sau:



Có thể dễ dàng quan sát được độ nhạy rải nhanh của ảnh từ góc trên bên trái xuống góc dưới bên phải. Nặng lượng tập trung nhất vào điểm (0,0) của ma trận hệ số biến đổi.



Hình 8.6. Biến đổi FFT xugi(trái) và ngược (phải).

### C. LUONG TỬ HOÁ

Khối lượng tử hóa trong sơ đồ nén đóng vai trò quan trọng và quyết định tỉ lệ nén của chuẩn nén JPEG. Đầu vào của khối lượng tử hóa là các ma trận hệ số biến đổi Cosin của các khối điểm ảnh. Nguyên tắc lượng hóa đã trình bày trong mục 2.2.2.

Để giảm số bộ lượng tử, người ta tìm cách quy các hệ số ở các khối về cùng một khoảng phân bố. Chuẩn nén JPEG chỉ sử dụng một bộ lượng tử hóa. Giả sử rằng, các hệ số đều có hàm tính xác suất xuất hiện như nhau. Chúng ta sẽ căn chỉnh lại hệ số  $y_j$  bằng phép gán :

$$y_j = \frac{y_j - \mu_j}{\sigma_j}$$

Với  $\mu_j$  là trung bình cộng của hệ số thứ  $j$ .

$\sigma_j$  là độ lệch cơ bản của hệ số thứ  $j$ .

Như vậy chúng ta sẽ đồng nhất được mức quyết định và mức tạo lại cho tất cả các hệ số. Do đó, các hệ số sẽ được biểu diễn bằng cùng một số lượng bit. Có nhiều cách tiếp cận để tính được các mức quyết định và mức tạo lại. Lloyd — Max đưa ra giải thuật sau:

Bước 1: Chọn giá trị khởi tạo:

$$d_0 = y_L$$

$$d_N = y_H$$

$$r_0 = d_0$$

$N$  là số mức lượng tử

Bước 2: Cho  $i$  biến thiên từ 1 đến  $N-1$  thực hiện các công việc sau:

a. Tính  $d_i$  theo công thức:

$$d_{i-1} = \frac{\int_{d_{i-1}}^{d_i} y \cdot p(y) dy}{\int_{d_{i-1}}^{d_i} p(y) dy}$$

b. Tính  $r_i$  theo công thức:

$$r_i = 2d_i - r_{i-1}$$

$$\text{Bước 3: Tính } r' = \frac{\int_{d_{N-1}}^{d_N} y \cdot p(y) dy}{\int_{d_{N-1}}^{d_N} p(y) dy}$$

Bước 4: Nếu  $r_{N-1} \neq r'$  điều chỉnh lại  $r_0$  và lặp lại từ bước 2 đến bước 4.

Trong quá trình cài đặt thủ tục tạo ra bộ lượng tử hoá, Lloyd và Max đã có nhiều cải tiến để tính toán dễ dàng hơn. Xác định  $d_i$ , bằng công thức trong bước 2a được tiến hành theo phương pháp Newton-Raphson.

Sau đây là các bước mô tả toàn bộ công việc của khối lượng tử hoá tác động lên các hệ số biến đổi Cosin:

**Bước 1:** Tính trung bình cộng  $\mu$  và độ lệch cơ bản  $\sigma$  cho từng hệ số ở mỗi vị trí trong khối:

$$\mu_j = \frac{\sum y_j}{n}$$

$$\sigma_j = \frac{n \sum y_j^2 - (\sum y_j)^2}{n(n-1)}$$

Với  $y_j$  là hệ số thứ  $j, n$  là số khối.

**Bước 2:** Lựa chọn tỉ lệ số hệ số giữ lại trong một khối.

**Bước 3:** Giữ lại các hệ số có độ lệch cơ bản lớn hơn.

**Bước 4:** Lập ma trận  $T$  sao cho:  $T_{ij} = 1$  nếu hệ số  $(i,j)$  được giữ lại.

**Bước 5:** Cân chỉnh lại giá trị của các hệ số xoay chiều được giữ lại ở các khối:

$$C_{ij} = \frac{C_{ij} - \mu_{ij}}{\sigma_{ij}}$$

**Bước 6:** Tính phân bố của các giá trị xoay chiều đã cân chỉnh.

**Bước 7:** Tính độ lệch cơ bản  $\sigma_i$  của các phân bố vừa tính.

**Bước 8:** Lượng tử hoá các hệ số xoay chiều bằng cách sử dụng bộ lượng tử Lloyd-Max sau khi đã điều chỉnh mức quyết định và mức tạo lại của nó theo cách sau:

$$d_i \leftarrow d_i \times \sigma_i$$

$$r_i \leftarrow r_i \times \sigma_i$$

$$d_N = -d_0$$

Thành phần một chiều sẽ không lượng tử hoá. Đến đây, ta chuyển sang bước nén.

## D - NÉN

Đầu vào của khối nén gồm hai thành phần: thành phần các hệ số một chiều và thành phần các hệ số xoay chiều.

Thành phần các hệ số một chiều  $C_i(0,0)$  với  $i = 0, 1, \dots, 63$  chứa phần lớn năng lượng tín hiệu hình ảnh. Người ta không nên trực tiếp các giá trị  $C_i(0,0)$  mà xác định độ lệch của  $C_i(0,0)$ :

$$d_i = C_{i+1}(0,0) - C_i(0,0)$$

$d_i$  có giá trị nhỏ hơn nhiều so với  $C_i$  nên trong biểu diễn dấu phẩy động theo chuẩn IEEE754 thường chứa nhiều chuỗi bit 0 nên có thể cho hiệu suất nén cao hơn. Giá trị  $C_i(0,0)$  và các độ lệch  $d_i$  được ghi ra một tệp tạm. Tệp này được nén bằng phương pháp nén Huffman.

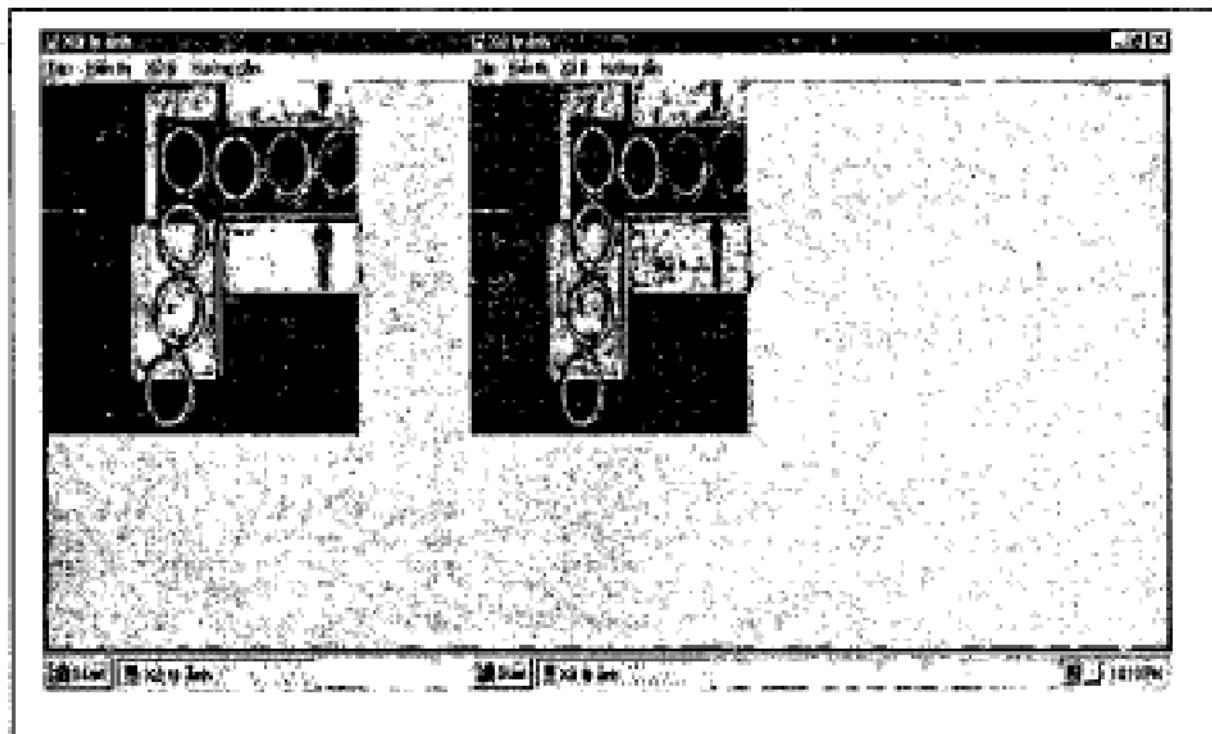
Thành phần các hệ số xoay chiều  $C_i(m,n)$  với  $1 \leq m \leq 7$ ,  $1 \leq n \leq 7$  chứa các thông tin chi tiết của ảnh. Để nâng cao hiệu quả nén cho mỗi bộ hệ số trong một khối người ta xếp lại chúng theo thứ tự ZigZag. Có thể hình dạng hình ZigZag như bảng trang dưới đây.

Tác dụng của sắp xếp lại theo thứ tự ZigZag là tạo ra nhiều loạt hệ số giống nhau. Chúng ta biết rằng năng lượng của khối bộ số giảm dần từ góc trên bên trái xuống góc dưới bên phải nên việc sắp xếp lại các hệ số theo thứ tự ZigZag sẽ tạo điều kiện cho các hệ số xấp xỉ nhau (cùng mức *lượng tử*) nằm trên một dòng.

0	2	3	9	10	20	21	35
1	4	8	11	19	22	34	36
5	7	12	18	23	33	37	48
6	13	17	24	32	38	47	49
14	16	25	31	39	46	50	57
15	26	30	40	45	51	56	58
27	29	41	44	52	55	59	62
28	42	43	53	54	60	61	63

Mỗi khối ZigZag này được mã hóa theo phương pháp RLE. Cuối mỗi khối đều ra của RLE, ta đặt dấu kết thúc khối EOB (*End Of Block*).

Sau đó, các khối được dồn lại và mã hóa một lần bằng phương pháp mã Huffman. Nhờ có dấu kết thúc khối nên có thể phân biệt được hai khối cạnh nhau khi giải mã Huffman. Hai bảng mã Huffman cho hai thành phần hệ số tất nhiên sẽ khác nhau.



a) Ảnh nén với độ mất mát  
thông tin 50% - mã 6 bit  
Tí lệ nén: 9,775

b) Ảnh nén với độ mất mát 50%  
thông tin 50% - mã 3 bit  
Tí lệ nén: 10,064

Hình 8.7 Một số kết quả nén với độ mất mát thông tin khác nhau (chuẩn JPEG)

Để có thể giải nén được, chúng ta phải ghi lại thông tin như: kích thước ảnh, kích thước khối, ma trận T, độ lệch tiêu chuẩn, các mức tạo lại, hai bảng mã Huffman, kích thước khối nén một chiều, kích thước khối nén xoay chiều... và ghi nối tiếp vào hai file nén của hai thành phần hệ số.

Cài đặt giải thuật cho nén JPEG thực sự phức tạp. Chúng ta phải nắm được các kiến thức về nén RLE, Huffman, biến đổi Cosin, xây dựng bộ lượng tử hóa Lloyd - Max... Nén và giải nén JPEG hơi chậm nhưng bù lại thời gian truyền trên mạng nhanh hơn do kích thước tệp nén nhỏ.

- Với những ưu điểm của mình, JPEG được ISO chấp nhận là chuẩn ảnh quốc tế và được biết đến dưới mã số ISO 10918-1 .

#### 8.4. PHƯƠNG PHÁP MÃ HOÁ THẾ HỆ THỨ HAI

Phương pháp mã hóa dựa vào biến đổi thế hệ thứ hai, như đã nói trong phần giới thiệu chương, có thể phân thành 2 lớp nhỏ:

Lớp phương pháp sử dụng các phép toán cục bộ để tổ hợp điều ra theo cách thức hợp lý và lớp phương pháp sử dụng biểu diễn ảnh. Dưới đây, trong lớp phương pháp thứ nhất chúng ta sẽ xem xét một phương pháp có tên gọi là "Kim tự tháp Laplace"; còn trong lớp phương pháp thứ hai sẽ đề cập 2 phương pháp là vùng gián tăng và phương pháp tách - hợp.

#### 8.4.1. Phương pháp Kim tự tháp Laplace (Pyramide Laplace)

Phương pháp này là tổ hợp của 2 phương pháp: mã hoá thích nghi và biến đổi. Tỷ số nén là khá cao, thường là 10 trên 1. Về nguyên tắc, phương pháp này dựa vào mô hình quan sát phân cấp của hệ thống quan sát của con người.

Bắt đầu từ ảnh gốc  $x(m,n)$  qua bộ lọc dài thấp ta thu được tín hiệu  $x_i(m,n)$ . Bộ lọc này được thiết kế để tính trung bình cục bộ dựa vào đáp ứng xung 2 chiều gần với đường cong Gauss. Bộ lọc này đóng vai trò "dự đoán" với sai số  $e_i(m,n)$  tính bởi:

$$e_i(m,n) = x(m,n) - x_i(m,n) \quad (8.31)$$

Như vậy là mã hoá của  $x_i(m,n)$  và  $e_i(m,n)$  là tương đương với mã hoá của  $x(m,n)$ . Với cách biến đổi như trên,  $e_i(m,n)$  thuộc loại dài cao. Vì mắt người ít cảm nhận được tín hiệu với tần số cao nên ta có thể dùng một lượng bit ít hơn để mã hoá cho nó. Mặt khác, tín hiệu  $x_i(m,n)$  thuộc loại dài thấp, nên theo lý thuyết lấy mẫu số mẫu sẽ ít hơn.

Quá trình này được lặp lại bằng cách dùng các bộ lọc thấp khác nhau và ta sẽ thu được các tín hiệu  $x_i(m,n)$ ,  $i = 1, 2, \dots$ . Với mỗi lần lặp, kích thước của ảnh sẽ giảm đi một lượng bằng  $f_i/f_{i+1}$ . Theo cách này, ta có một cấu trúc xếp chồng tựa như cấu trúc kim tự tháp mà kích thước giảm dần từ gốc đến đỉnh. Nhân chập Gauss được dùng ở đây có kích thước 5 x 5. Các tín hiệu ra sau đó được lượng hoá và mẫu hoá.

Theo kết quả đã công bố [6], với bộ lọc dài thấp một chiều tách được với các trọng số:  $g(0) = 0,7$ ,  $g(-1) = g(1) = 0,25$  và  $g(-2) = g(2) = 0,1$ . Tỷ số nén dao động từ 6 trên 1 đến 32 trên 1. Tuy nhiên, nếu tỷ số nén cao, ảnh kết quả sẽ có biến dạng.

#### 8.4.2 Phương pháp mã hoá dựa vào biểu diễn ảnh

Như đã biết, trong xử lý ảnh, tùy theo các ứng dụng mà ta cần toàn bộ ảnh hay chỉ những đặc tính quan trọng của ảnh. Các phương pháp phân vùng ảnh nếu trong chương sau như hợp vùng, tách, tách và hợp là rất hữu ích và có thể ứng dụng để nén ảnh. Có thể có nhiều phương pháp khác, song dưới đây chúng ta chỉ đề cập tới 2 phương pháp: vùng gián tăng và phương pháp tách-hợp.

### 8.4.2.1 Mã hóa dựa vào vùng giá trị

Kỹ thuật vùng giá trị thực chất là hợp các vùng có cùng một số tính chất nào đó. Kết quả của nó là một ảnh được phân đoạn giống như một ô trong trò xếp chữ (puzzle). Tuy nhiên, cần lưu ý rằng tất cả các đường bao thu được không tạo nên một ảnh giống ảnh gốc.

Như nêu trong chương 6 (mục 6.3), việc xác định tính chất miền đồng nhất xác định độ phức tạp của phương pháp. Để đơn giản, tiêu chuẩn chọn ở đây là *khoảng màu xám*. Như vậy, miền đồng nhất là tập hợp các điểm ảnh có mức xám thuộc khoảng đã chọn. Cũng cần lưu ý thêm rằng, ảnh gốc có thể gồm có đường bao và các kết cấu (texture). Trong miền texture, độ xám biến đổi rất chậm. Do vậy, nếu không chú ý sẽ chia ảnh thành quá nhiều miền và gây nên các bao già. Giải pháp để khắc phục hiện tượng này là dùng một bộ lọc thích hợp hay lọc trung vị.

Sau giai đoạn này, ta thu được ảnh phân đoạn với các đường biên kin, độ rộng 1 pixel. Để loại bỏ các đường bao già, ta có thể dùng phương pháp gradient (xem chương 5). Sau khi đã thu được các đường bao đúng, người ta tiến hành mã hóa (xắp xì) đường bao bởi các đường cong hình học, thí dụ bởi các đoạn thẳng hay đường cong. Nếu ảnh gốc có độ phân giải không thích hợp, người ta dùng khoảng 1.3 bit cho một điểm biên.

Phương pháp này thể hiện ưu điểm: đó là mô hình tham số. Các tham số ở đây là số vùng, độ chính xác mô tả. Tất nhiên tham số khoảng mốc xám là quan trọng nhất vì nó có ảnh hưởng đến tỷ số nén. Một tham số cũng không kém phần quan trọng là số điểm của các đường bao bị coi là già. Thường số điểm này không vượt quá 20 điểm.

### 8.4.2.2 Phương pháp tách - hợp

Cũng như đã chỉ ra trong chương 6, phương pháp tách-hợp khắc phục được một số nhược điểm của phương pháp phân vùng dựa vào tách vùng hay hợp vùng. Trong phương pháp mã hóa này, người ta thay tiêu chuẩn chọn vùng đơn giản ở trên bằng một tiêu chuẩn khác hiệu quả hơn.

Nguyên tắc chung của phương pháp là theo mô hình *biên - texture*. Nhìn chung đường biên dễ nhạy cảm với mắt người, còn texture thì ít nhạy cảm hơn. Người ta mong muốn rằng đường phân ranh giữa các vùng là đồng nhất với các đường bao. Lưu ý rằng, cần quyết định phân vùng một phần của ảnh sao cho nó không được cắt chéo đường bao. Đây là một tiêu chuẩn kiểm tra quan trọng. Các đường bao thường nhận được bởi các bộ lọc thông cao, đẳng hướng.

Để có thể quản lý các điểm thuộc một vùng một cách tốt hơn, tiêu chuẩn kiểm tra thứ hai cũng được xem xét đó là dấu: "*các điểm nằm về một phía của đường bao có cùng dấu*".

Nhìn chung, phương pháp gồm 2 giai đoạn. Giai đoạn đầu thực hiện việc tách vùng. Giai đoạn sau thực hiện hợp vùng.

Quá trình tách thực hiện trước. Người ta chia ảnh gốc thành các vùng nhỏ kích thước  $9 \times 9$ . Tiếp theo, tiến hành xếp xì các vùng ảnh đó bằng một đa thức có bậc nhô hơn 3. Sau quá trình tách, ta thu được trong một số vùng của ảnh, các hình vuông liên tiếp. Chúng sẽ tạo nên một miền gốc lớn và không nhất thiết vuông. Như vậy, trong trường hợp này phải xếp xì bằng rất nhiều các đa thức giống nhau. Rõ ràng là việc mã hóa riêng biệt các đa thức là điều kém hiệu quả và người ta ngũ đến hợp các vùng để giảm độ dư thừa này.

Quá trình hợp được tiến hành như sau: nếu 2 vùng có thể xếp xì bởi 2 đa thức tương tự, người ta hợp chúng làm một và chỉ dùng một đa thức xếp xì. Nếu mức độ thay đổi là thấp, ta sẽ có nhiều cặp vùng tương tự. Để có thể nhận được kết quả không phụ thuộc vào lần hợp đầu, người ta xây dựng đồ thị "Vùng kế cận". Các nút của đồ thị này là các vùng và các liên hệ biểu diễn mối không tương đồng. Sự liên hệ với mức không tương đồng thấp chỉ ra rằng 2 vùng cần hợp lại.

Sau bước hợp này, đồ thị được cập nhật lại và quá trình hợp được lặp lại cho đến khi tiêu chuẩn là thỏa. Quá trình hợp dừng có thể quyết định bởi chất lượng ảnh nén hay một tiêu chuẩn nào khác.

Ta có thể thấy rằng phương pháp này khá phức tạp song bù lại nó cho tỷ số nén khá cao: 60 trên 1 [6].

## 8.5. KẾT LUẬN

Mỗi phương pháp nén đều có những ưu điểm và nhược điểm. Tính hiệu quả của phương pháp không chỉ phụ thuộc vào tỷ số nén mà còn vào nhiều chỉ tiêu khác như: độ phức tạp tính toán, nhạy cảm với nhiễu, chất lượng, kiểu ảnh, v.v.

Nén là một vấn đề lớn được quan tâm nhiều và có liên quan đến nhiều lĩnh vực khác nhau. Chúng ta không hy vọng có thể trình bày tất cả trong một chương. Song dù sao, chương này cũng cung cấp một số khái niệm về các phương pháp khai dụng và một số phương pháp mới về nén dữ liệu nhất là nén ảnh. Bảng tổng kết dưới đây cung cấp cho chúng ta một cách nhìn toàn diện về các phương pháp nén.

Bảng so sánh kết quả một số phương pháp nén

Phương pháp	Tỷ số Nén	Độ phức tạp	Chất lượng	Nhạy cảm với nhiễu	Kiểu ảnh
RLC	10	đơn giản	rất tốt	lớn	nhiệt phán
Dự đoán	2-4	đơn giản	rất tốt	trung bình	mọi ảnh
Biến đổi	10-15	phức tạp	tốt	rất kém	đa cấp xám
Pyramide Laplace	5-10	trung bình	tốt	lớn	đa cấp xám
Vùng giá tăng	20-30	phức tạp	Trung bình	rất lớn	đa cấp xám
Tách và hợp	60-70	rất phức tạp	Trung bình	rất lớn	đa cấp xám

### Bài tập chương 8

- Viết một chương trình nén và giải nén theo phương pháp RLC (đơn giản, dọc, ngang hay kết hợp).
- Viết một chương trình nén và giải nén theo phương pháp Huffman.
- Viết một chương trình nén và giải nén theo phương pháp LZW.
- Viết thủ tục thực hiện biến đổi Cosin thuận.
- Viết thủ tục thực hiện biến đổi Cosin ngược.
- Viết thủ tục thực hiện lượng hóa theo thuật toán Lloyd-Max.

**PHỤ LỤC A****Định dạng ảnh (Image File Format)****1. Định dạng ảnh IMG**

Ảnh IMG là ảnh đen trắng. Phần đầu của ảnh IMG có 16 bytes chứa các thông tin cần thiết :

- + 6 bytes đầu: dùng để đánh dấu định dạng ảnh IMG. Giá trị của 6 bytes này viết dưới dạng Hexa:

0x0001 0x0008 0x0001.

- + 2 bytes tiếp theo: chứa độ dài mẫu tin. Đó là độ dài của dãy các bytes kế liền nhau mà dãy 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. Đó là cách lưu trữ nén đã được đề cập chi tiết trong chương 8.

- + 4 bytes tiếp: mô tả kích cỡ pixel

- + 2 bytes tiếp : số pixel trên một dòng ảnh

- + 2 bytes 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 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 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**

Tùy theo các bit cao của byte đầu được bật hay tắt. Nếu bit cao được bật (giá trị 1) thì đây là gói nén các bytes chỉ gồm bit 0, số các byte được nén được tính bởi 7 bit thấp còn lại. Nếu bit cao tắt (giá trị 0) thì đây là gói nén các byte gồm toàn bit 1. Số các byte được nén được tính bởi 7 bit thấp còn lại.

Các gói tin của file IMG phong phú như vậy là do ảnh IMG là ảnh đen trắng, do vậy chỉ cần 1 bit 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 giá trị 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 nhất. Nó sử dụng phương pháp mã hóa RLE (Run-LengthEncoded) để nén dữ liệu ảnh. Quá trình nén và giải nén được thực hiện trên từng dòng 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(xem hình 2.10).

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 kiểu PCX/PCC 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 có bảng màu.

+ 1 byte: chỉ ra phương pháp mã hoá. Nếu là 0 thì mã hoá theo phương pháp BYTE PACKED, nếu không là phương pháp RLE.

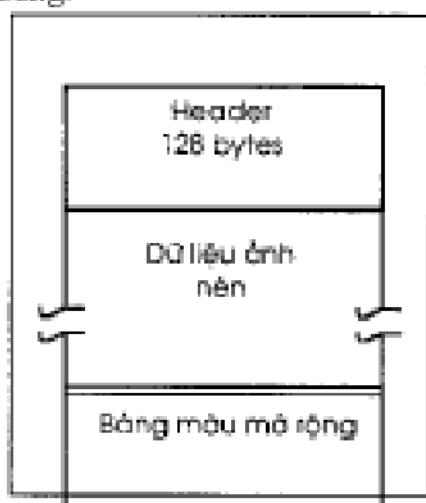
+ 1 byte: số bit cho một điểm ảnh plane.

+ 1 word: tọa độ góc trái trên của ảnh. Với kiểu PCX nó có giá trị là (0,0); còn PCC thì khác (0,0).

+ 1 word: tọa độ góc phải dưới.

+ 1 word: kích thước bê rộng và bê cao ảnh.

- + 1 word: số điểm ảnh.
- + 1 word: độ phân giải màn hình.
- + 1 word.
- + 48 byte: chia 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 bit plane mà ảnh sử dụng. Với ảnh 16 màu, giá trị này là 4, với ảnh 256 màu (1 pixel/8 bit) thì số bit plane lại là 1.
- + 1 byte: số bytes cho một dòng quét ảnh.
- + 1 word: kiểu bảng màu.
- + 58 byte: không dùng.



Cấu trúc tệp ảnh dạng PCX.

Tóm lại, định dạng ảnh PCX thường được dùng để lưu trữ ảnh vì 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 nó làm tăng kích thước lưu trữ. Và cũng vì nhược điểm này mà một số ứng dụng lại sử dụng một kiểu định dạng khác mềm dẻo hơn: định dạng TIFF (Tagged Image Efile 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): có trong tất cả các tệp TIFF và gồm 8 byte:

+ 1 word: chỉ ra kiểu tạo tệp trên máy tính PC hay 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. Có thể coi đó là đặc trưng của file TIFF vì nó không thay đổi.

+ 2 word: giá trị Offset theo byte tính từ đầu file tới cấu trúc IFD(Image File Directory) là cấu trúc thứ hai của file. Thứ tự các byte ở đây phụ thuộc vào dấu hiệu trưởng đầu tiên.

- Phần thứ 2 (IFD): nó không ở ngay sau cấu trúc IFH mà vị trí của nó được xác định bởi trường Offset trong đầu tệp. Có thể có một hay nhiều IFD cùng tồn tại trong file (nếu file có nhiều hơn 1 ảnh).

Một IFD 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 trả tới IFD tiếp theo. Nếu đây là IFD cuối cùng thì trường này có giá trị là 0.

- Cấu trúc phần dữ liệu 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 đã được xây dựng.

+ 2 byte: kiểu dữ liệu của tham số ảnh. Có 5 kiểu tham số cơ bản:

a) 1: BYTE (1 byte).

b) 2: ASCII (1 byte).

c) 3: SHORT (2 byte).

d) 4: LONG (4 byte).

e) 5: RATIONAL (8 byte).

+ 4 byte: trường độ dài (bộ đếm) chứa số lượng chỉ mục của kiểu dữ liệu đã 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 thực liên quan tới dấu hiệu, tức là dữ liệu 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 bốn kiểu khác nhau được lưu trữ trong byte dấu hiệu nén.

Như đã nói ở trên, file ảnh TIFF là dùng để giải quyết vấn đề khó mở rộng của file PCX. Tuy nhiên, với cùng một ảnh thì việc dùng file PCX chiếm ít không gian nhỏ hơn.

#### 4. Định dạng ảnh GIF(Graphics Interchanger Format)

Cách lưu trữ kiểu PCX có lợi về không gian lưu trữ với ảnh đen trắng kích thước tệp có thể nhỏ hơn bản gốc từ 5 đến 7 lần. Với ảnh 16 màu, kích thước ảnh nhỏ hơn ảnh gốc 2-3 lần, có trường hợp có thể gấp xấp xỉ ảnh gốc. Tuy nhiên, với ảnh 256 màu thì nó bộc lộ rõ khả năng nén rất kém. Điều này có thể lý giải như sau: khi số màu tăng lên, các lôại dài xuất hiện ít hơn và vì thế, lưu trữ theo kiểu PCX không còn lợi nữa. Hơn nữa, nếu ta muốn lưu trữ nhiều đối tượng trên một tệp ảnh như kiểu định dạng TIFF, đòi hỏi có một định dạng khác thích hợp.

Định dạng ảnh GIF do hãng CompuServer Incorporated (Mỹ) để xuất lần đầu tiên vào năm 1990. Với định dạng GIF, những vuông mắc mà các định dạng khác gặp phải khi số màu trong ảnh tăng lên không còn nữa. Khi số màu càng tăng thì ưu thế của định dạng GIF càng nổi trội. Những ưu thế này có được là do GIF tiếp cận các thuật toán nén LZW(Lempel-Ziv-Welch). Bản chất của kỹ thuật nén LZW là dựa vào sự lặp lại của một nhóm điểm chứ không phải loạt dài giống nhau. Do vậy, dữ liệu càng lớn thì sự lặp lại càng nhiều (xem chương 8). Định dạng ảnh GIF cho chất lượng cao, độ phân giải đồ họa cũng đạt cao, cho phép hiển thị trên hầu hết các phần cứng đồ họa.

Định dạng tổng quát của ảnh GIF như sau:

- Chữ ký của ảnh.
- Bộ mô tả hiển thị.
- Bản đồ màu tổng thể.

- Mô tả một đối tượng của ảnh

- Đầu phân cách
- Bộ mô tả ảnh
- Đầu phân cách
- Bản đồ màu cục bộ
- Dữ liệu ảnh

GIF note
GIF header
Global Palette
Header Image (10 byte)
Palette of Image 1(nếu có)
Data of Image 1
; ký tự liên kết
.....
; GIF terminator

Phần mô tả này lặp n lần nếu ảnh chứa n đối tượng.

- Phần đầu cuối ảnh GIF(terminator)

Chữ ký của ảnh GIF có giá trị là GIF87a. Nó gồm 6 ký tự, 3 ký tự đầu chỉ ra kiểu định dạng, 3 ký tự sau chỉ ra version của ảnh.

- Bộ hình hiển thị: chứa mô tả các thông số cho toàn bộ ảnh GIF:

- + Độ rộng hình raster theo pixel: 2 byte;
- + Độ cao hình raster theo pixel: 2 byte;
- + Các thông tin về bản đồ màu, hình hiển thị,...
- + Thông tin màu nền: 1 byte;
- + Phản ứng dừng: 1 byte.

- Bản đồ màu tổng thể: mô tả bộ màu tối ưu đòi hỏi khi bit M = 1. Khi bộ màu tổng thể được thể hiện, nó sẽ xác lập ngay bộ mô tả hình hiển thị. Số lượng thực thể bản đồ màu lấy theo bộ mô tả hình hiển thị ở trên và bằng  $2^m$ , với m là lượng bit trên một pixel khi mỗi thực thể chứa đựng 3 byte (biểu diễn cường độ màu của ba màu cơ bản Red-Green-Blue). Cấu trúc của khối này như sau:

Bit	Thứ tự byte	Mô tả
màu Red	1	giá trị màu đỏ theo index 0
màu Green	2	giá trị màu xanh lục theo index 0
màu Blue	3	giá trị màu xanh lá theo index 0

màu Red	4	giá trị màu đỏ theo index 1
màu Green	5	giá trị màu xanh lục theo index 1
màu Blue	6	giá trị màu xanh lơ theo index 0
.....	.....	.....

- Bộ mô tả ảnh: định nghĩa vị trí thực tế và phản mô rộng của ảnh trong phạm vi không gian ảnh đã có trong phần mô tả hình hiển thị. Nếu ảnh biểu diễn theo ảnh xạ bản đồ màu cục bộ thì cờ định nghĩa phải được thiết lập. Mỗi bộ mô tả ảnh được chỉ ra bởi ký tự kết nối ảnh. Ký tự này chỉ được dùng khi định dạng GIF có từ 2 ảnh trở lên. Ký tự này có giá trị 0x2c (ký tự dấu phẩy). Khi ký tự này được đọc qua, bộ mô tả ảnh sẽ được kích hoạt.

Bộ mô tả ảnh gồm 10 byte và có cấu trúc như sau:

Các bit	Thứ tự byte	Mô tả
00101100	1	Ký tự liên kết ảnh (‘,’)
Căn trái ảnh	2,3	Pixel bắt đầu ảnh tĩnh từ trái hình hiển thị
Căn định trên	4,5	Pixel cuối ảnh bắt đầu tĩnh từ định trên hình hiển thị
Độ rộng ảnh	6,7	Chiều rộng ảnh tĩnh theo pixel
Độ cao ảnh	8,9	Chiều cao ảnh tĩnh theo pixel
M1000pixel	10	Khi bit M = 0 : sử dụng bản đồ màu tổng thể M = 1 : sử dụng bản đồ màu cục bộ I = 0 : định dạng ảnh theo thứ tự liên tục I = 1 : định dạng ảnh theo thứ tự xen kẽ pixel +1 : số bit/pixel của ảnh này.

- Bản đồ màu cục bộ: bản đồ màu cục bộ chỉ được chọn khi bit M của byte thứ 10 là 1. Khi bản đồ màu được chọn, bản đồ màu sẽ chiều theo bộ mô tả ảnh mà lấy vào cho đúng. Tại phần cuối ảnh, bản đồ màu sẽ lấy lại phần xác lập sau bộ mô tả hình hiển thị.

Lưu ý là trường “pixel” của byte thứ 10 chỉ được dùng khi bản đồ màu được chỉ định. Các tham số này không những chỉ cho biết kích thước ảnh theo pixel mà còn chỉ ra số thực thể bản đồ màu của nó.

- Dữ liệu ảnh: chuỗi các giá trị có thứ tự của các pixel màu tạo nên ảnh. Các pixel được xếp liên tục trên một dòng ảnh, từ trái qua phải. Các dòng ảnh được viết từ trên xuống dưới.

- Phần kết thúc ảnh: cung cấp tính đồng bộ cho đầu cuối của ảnh GIF. Cuối của ảnh sẽ xác định bởi kí tự ";" (0x3b).

Định dạng GIF có rất nhiều ưu điểm và đã được công nhận là chuẩn để lưu trữ ảnh màu thực tế (chuẩn ISO 10918-1). Nó được mọi trình duyệt Web (Web Browser) hỗ trợ với nhiều ứng dụng hiện đại. Cùng với nó có chuẩn JPEG (Joint Photograph Expert Group). GIF dùng cho các ảnh đồ họa (Graphic), còn JPEG dùng cho ảnh chụp (Photographic).

## 5. Định dạng ảnh JPEG (Joint Photograph Expert Group)

### 5.1. Sơ lược lịch sử JPEG

JPEG (Joint Photographic Expert Group) là tên của một tổ chức nghiên cứu các chuẩn nén cho ảnh tone liên tục (trước đây là IOS) được thành lập vào năm 1982. Năm 1986, JPEG chính thức được thiết lập giữa nhóm IOS/IEC và ITU. Tiêu chuẩn này có thể được ứng dụng cho nhiều lĩnh vực: lưu trữ ảnh, Fax màu, truyền ảnh báo chí, ảnh cho y học, camera số.

- Tháng 2-1989: JPEG ấn bản 1 ra đời.
- Tháng 8-1990: JPEG ấn bản 8 ra đời (ấn bản cuối).
- Tháng 7-1994: JPEG được công nhận bởi 22 thành viên chính của IOS/IEC JTCI và trở thành 10918-1.
- Tháng 8-1994: Dự thảo công nhận cho phần JPEG mở rộng.
- Tháng 11-1994: CD cho ISO/IEC 10918-3 kết thúc.

JPEG được tổ chức thành các đoạn (segments) và dùng hệ thống đánh dấu (marker) để nhận dạng. Mỗi marker gồm 2 bytes và có ý nghĩa riêng: header, các bảng mã, điểm bắt đầu và điểm kết thúc của số liệu ảnh. Mỗi đoạn có chiều dài tối đa là 65535 và bắt đầu bởi marker nhận diện và kết thúc bằng marker.

### 5.2. Hệ thống vạch dấu và cách tổ chức thông tin trong tệp JPG

**Chú ý:** Định dạng tệp JPEG/JPIF sử dụng kiểu định dạng Motorola cho từ, không dùng kiểu Intel, có nghĩa là byte cao trước, byte thấp sau (ví dụ: từ FFA0 sẽ được ghi trong tệp JPEG theo thứ tự : FF tại phần thấp offset , A0 tại phần cao offset).

Một marker gồm 2 bytes, bắt đầu bằng 0xFF và kết thúc bằng byte nằm trong khoảng 0 và 0xFF. Ví dụ: 'FFDA', 'FFC4', 'FF00'. Byte thứ hai chỉ rõ ý nghĩa của vạch dấu.

Ví dụ:

- Vạch dấu cho việc bắt đầu quá trình giải mã được gọi là SOS (Start Of Scan) có giá trị là 'FFDA'.

- Vạch dấu khác gọi là DQT (Define Quantization Table) có giá trị là 0xFFDB chỉ ra rằng: trong tệp JPG , sau vạch dấu và sau 3 bytes sẽ là kế tiếp 64 bytes (hệ số bảng lượng tử).

Nếu trong quá trình đọc tệp JPG, bắt gặp giá trị 0xFF, sau đó lại bắt gặp a bytes khu 0 và byte này không có ý nghĩa của vạch dấu (không tìm ra vạch dấu tương ứng của byte đó), sau đó nếu gặp byte 0xFF thì phải nhảy qua và tiếp tục quá trình đọc tệp JPG (trong một số JPG, trình tự 0xFF liên tục có mục đích là để lấp đầy và nhất thiết phải bỏ qua). Do vậy, bất kỳ khi nào nếu bắt gặp 0xFF , thì phải kiểm tra ngay byte tiếp theo xem giá trị 0xFF đó là vạch dấu hay bỏ qua.

Điều gì sẽ xảy ra khi chúng ta thực sự cần mã hóa giá trị byte 0xFF trong tệp JPG như giá trị byte bình thường ( không phải là vạch dấu hoặc byte lấp đầy ) ? Giả sử chúng ta cần ghi mã Huffman bắt đầu với 11111111 (8 bits of 1) như là byte liên tục. Theo chuẩn thì chỉ cần đơn giản là tạo ra các số 0 kế tiếp, và ghi theo tuần tự 'FF00' vào tệp JPG, nhưng khi mã hóa JPG gấp 2 byte 'FF00' kế tiếp, nó sẽ coi đó là byte: 0xFF như một byte bình thường.

Mặt khác, điều gì sẽ xảy ra nếu trong khi đang mã hóa Huffman và đang chèn thêm bits vào các bytes của tệp ảnh JPG và việc chèn bit chưa kết thúc mà ta phải ghi một vạch dấu cần chỉnh ? Đối với byte cần chỉnh của vạch dấu, ta thiết lập giá trị 1 cho những bit còn lại cho tới bắt đầu của byte tiếp theo, sau đó ghi vạch dấu vào byte tiếp theo.

#### Một số vạch dấu quan trọng trong định dạng ảnh JPG

- SOI : Start Of Image ( Bắt đầu ảnh ) = 'FFD8'. Vạch dấu này bắt buộc trong bất kỳ tệp ảnh nào ở vị trí đầu tệp.

- EOI : End Of Image ( Kết thúc ảnh ) = FFD9'
- RST $i$  : Restart Markers ( Vạch dấu khởi đầu lại ) = FFD $i$  ( $i \in [0..7]$ , RST0 = FFD0, ..., RST7=FFD7). Vạch dấu khởi đầu lại xuất hiện trong dòng bytes ở những khoảng cách đều nhau, trong quá trình giải mã sau SOS. Chúng xuất hiện theo thứ tự: RST0 -- interval -- RST1 -- interval -- RST2 -- ... -- RST6 -- interval -- RST7 -- interval -- RST0 -- ... Một số JPGs không có vạch dấu khởi động lại. Nhớ rằng đối với những byte định vị của vạch dấu những bits còn lại đều được đặt bằng giá trị 1, như vậy khi giải mã phải bỏ qua những khoảng được điền đầy những bit không có nghĩa ( giá trị thiết lập bằng 1) và vạch dấu RST.
- SOF0 : Start Of Frame 0 ( Bắt đầu khung ảnh ) = FFC0
- SOS : Start Of Scan ( Bắt đầu kiểm tra quét ảnh) = FFDA
- APP0 : Là vạch dấu dùng để nhận biết tệp ảnh JPG khi sử dụng đặc tả JFIF = FFB0
- COM : Comment ( Chủ giải ) = FFFE
- DNL : Define Number of Lines (Xác định số dòng ) = FFDC
- DRI : Define Restart Interval (Định nghĩa khoảng cách bắt đầu lại ) = FFDD
- DQT : Define Quantization Table (Định nghĩa bảng lượng tử hóa ) = FFDB
- DHT : Define Huffman Table (Định nghĩa bảng Huffman ) = FFC4

Trong tệp ảnh JPG về cơ bản có 2 loại bảng Huffman : Một cho DC và một cho AC. Trên thực tế có 4 bảng Huffman: 2 cho DC, AC thể hiện độ chói và 2 cho DC, AC thể hiện thành phần màu. Chú ý rằng chiều dài từ mã Huffman giới hạn bởi 16 bits. Các bảng đó trong tệp ảnh JPG cũng định dạng với nội dung như sau:

- 1) 16 bytes : byte i chứa số mã Huffman với chiều dài i ( chiều dài theo bits) i nằm trong khoảng từ 1 tới 16.

2) Chiều dài bảng (theo bytes) =  $\sum_{i=1}^{16}$  chiều dài từ mã i, tại vị trí [k][j], với

(k từ 1..16, j từ 0, ..., chiều dài từ mã-1).

Từ bảng này ta có thể tìm được mã Huffman thực sự liên kết với một byte cụ thể. Ví dụ số mã đối với chiều dài cho trước cho ví dụ cụ thể này được tính

như sau (chúng có thể có các giá trị khác):

- Chiều dài mã là 1, ta có  $\text{nr\_codes}[1] = 0$ , và bỏ qua chiều dài mã này
- Chiều dài mã là 2 ta có 2 mã: 00, 01
- Chiều dài mã là 3 ta có 3 mã : 100, 101, 110
- Chiều dài mã là 4 ta có 1 mã : 1110
- Chiều dài mã là 5 ta có 1 mã : 11110
- Chiều dài mã là 6 ta có 1 mã : 111110
- Chiều dài mã là 7 ta có 0 mã : bỏ qua (nếu có 1 mã chiều dài mã 7, mã là 1111110)
- Chiều dài mã là 8 ta có 1 mã : 11111100 (mã vẫn tiếp tục dịch qua trái khi ta đã bỏ qua giá trị mã chiều dài mã 7).
- ....
- Chiều dài mã là 16, ... : tương tự .

### 5.3. Định dạng JPG/JFIF ( Jpeg Format Interchange File)

Chuẩn JPEG chỉ định một số vạch dấu dành riêng cho các ứng dụng. Các vạch dấu đó được gọi là APPn , n trong khoảng từ 0 tới 0xFF: APPn = FFE<sub>n</sub>. Đặc tả JFIF sử dụng vạch dấu APP0 (FFE0) để nhận dạng tệp JPG có sử dụng đặc tả này.

#### a) Cấu trúc tệp JPG

- Header (2 bytes): FFD8 - vạch dấu SOI.
- Một số "segments" ảnh (ý nghĩa của chúng sẽ được trình bày dưới đây)
- Kết thúc ảnh (2 bytes): FFD9 — vạch dấu EOI.

#### b) Mỗi segment gồm 2 phần:

- Header (4 bytes): FF, n, sh, sl  
FF: nhận dạng đoạn .  
n : type of segment ( kiểu đoạn 1 byte)  
sh, sl: kích thước đoạn (bao gồm cả hai bytes này trừ FF và n)
- Nội dung của đoạn : 65533 bytes tối đa.

#### *Chú ý*

- Có một số đoạn đặc biệt (biểu thị bằng dấu '\*' dưới đây). Nó không có đặc tả về kích thước và cũng không có nội dung
- Một số byte FF trong giữa đoạn cũng hợp lệ và sẽ bị bỏ qua cho đến hết.

c) Kiểu đoạn (đặc tả bởi byte thứ 2, byte đầu là FF)

\*TEM = \$01 thường là nguyên nhân lỗi giải mã, có thể lờ đi.

SOF0 = \$c0 Start Of Frame (JPEG tuân tu), chi tiết xem sau đây

SOF1 = \$c1 như trên

SOF2 = \$c2 không sử dụng

SOF3 = \$c3 không sử dụng

SOF5 = \$c5 không sử dụng

SOF6 = \$c6 không sử dụng

SOF7 = \$c7 không sử dụng

SOF9 = \$c9 mã hóa số học, thường không sử dụng

SOF10 = \$ca không sử dụng

SOF11 = \$cb không sử dụng

SOF13 = \$cd không sử dụng

SOF14 = \$ce không sử dụng

SOF15 = \$cf không sử dụng

DHT = \$c4 bảng mã Huffman

JPG = \$c8 không xác định / dành riêng (nguyên nhân lỗi giải mã)

DAC = \$cc bảng số học thường không sử dụng

\*RST0 = \$d0 RSTn sử dụng để đóng, có thể lờ đi

\*RST1 = \$d1 ...nt...

\*RST2 = \$d2 ...nt...

\*RST3 = \$d3 ...nt...

\*RST4 = \$d4 ...nt...

\*RST5 = \$d5 ...nt...

\*RST6 = \$d6 ...nt...

\*RST7 = \$d7 ...nt...

SOI = \$d8 bắt đầu ảnh

EOI = \$d9 kết thúc ảnh

SOS = \$da bắt đầu quê

DQT = \$db bảng lượng hoá

DNL = \$dc không sử dụng, lờ đi

DRI = \$dd khoảng bắt đầu lại

DHP = \$de bỏ qua

EXP = \$ef bỏ qua

APP0 = \$e0 vạch dấu cho ảnh JFIF

APP15 = \$ef bỏ qua

JPG0 = \$f0 bỏ qua

JPG13 = \$fd bỏ qua

COM = \$fe chủ giải

Tất cả các kiểu đoạn khác là để dành và có thể lờ đi hoặc bỏ qua.

#### *Kiểu đoạn SOFO: Start Of Frame 0:*

- \$ff, \$c0 (SOFO)
- chiều dài ( byte cao, byte thấp ), 8+components\*3
- độ chính xác dữ liệu (1 byte) thể hiện bits/mẫu, thường là 8 (12 và 16 không phù hợp với đa số phần mềm).
- độ cao của ảnh (2 bytes, Hi-Lo), > 0 nếu DNL không được hỗ trợ.
- độ rộng của ảnh (2 bytes, Hi-Lo), > 0 nếu DNL không được hỗ trợ.
- số thành phần(1 byte), 1: ảnh đa cấp xám; 3: ảnh màu YCbCr /YIQ, 4 : ảnh màu CMYK.
- mỗi thành phần: 3 bytes
  - + số hiệu thành phần id (1 = Y, 2 = Ch, 3 = Cr, 4 = I, 5 = Q).
  - + yếu tố lấy mẫu (bit 0-3 : chiều đứng , 4-7: chiều ngang ).
  - + số bảng lượng tử ).

Định dạng JFIF sử dụng hoặc 1 thành phần (Y: đa mức xám ) hoặc 3 thành phần (YCbCr, đối khi dùng YUV ).

#### *APP0: vạch dấu đoạn dùng đặc tả JFIF*

- \$ff, \$e0 (APP0)
- chiều dài ( byte cao, byte thấp ), phải  $\geq 16$  .
- JFIF#0 (\$4a, \$46, \$49, \$46, \$00), nhận dạng JFIF
- số tham chiếu chính, phải là 1 (nếu khác là lỗi).
- số tham chiếu phụ phải là 0..2 (nếu khác thì giải mã bằng cách khác).

- đơn vị mật độ của x/y
  - 0 : không mật độ, mật độ x/y được chỉ định bằng tỷ lệ thay thế
  - 1 : mật độ x/y là dots/inch
  - 2 : mật độ x/y là dots/cm
- mật độ x ( byte thấp, byte cao), # 0
- mật độ y ( byte thấp, byte cao), # 0
- chiều rộng viết ngắn gọn (thumbai width): 1 byte
- chiều cao viết ngắn gọn (thumbai height: 1 byte)
- n bytes để viết ngắn gọn (RGB 24 bit),  $n = \text{width} * \text{height} * 3$

Nếu đó không phải là JFIF#0, hoặc length is < 16, thì có thể đó không phải là đoạn JFIF và có thể bỏ qua.

- Nếu đơn vị=0, mật độ x =1, mật độ y =1, hiểu rằng tỷ lệ là 1:1 (tỷ lệ đều).
- Tệp JFIF với cách viết ngắn gọn thường rất hiếm, và thường được bỏ qua. Nếu không có cách viết ngắn gọn , thì width=0 và height=0.
- Nếu chiều dài không phù hợp với kích thước viết gọn, có thể đưa ra lời cảnh báo, sau đó tiếp tục giải mã.

#### *DRI: Xác định khoảng bắt đầu lại*

- \$ff, \$dd (DRI)
- chiều dài ( byte cao, byte thấp ), phải là = 4
- khoảng bắt đầu lại ( byte cao, byte thấp) trong đơn vị của khối MCUs, có nghĩa rằng mỗi n khối MCU có RSTn vạch dấu có thể tìm thấy.

Vạch dấu đầu tiên sẽ là RST0, tiếp đến RST1 v.v..., cuối cùng là RST7 rồi quay lại RST0.

#### *DQT: Xác định bảng lượng tử*

- \$ff, \$db (DQT)
- Chiều dài (byte thấp, byte cao)
- Thông tin bảng lượng tử 1 byte:
  - + bit 0..3: số bảng lượng tử QT (0..3, nếu khác thì lỗi)
  - + bit 4..7: độ chính xác của bảng lượng tử QT, 0 = 8 bit, nếu khác thì đó là 16 bit
- n bytes QT,  $n = 64 * (\text{độ chính xác} + 1)$

**Chú ý**

- Một đoạn đơn DQT có thể chứa bội số QTs, mà mỗi QT chứa byte thông tin của mình.
- Với độ chính xác = 1 (16 bit), theo thứ tự cao — thấp mỗi 64 từ.

**DHT: Define Huffman Table:**

- \$ff, \$e4 (DHT)
- Chiều dài (byte thấp, byte cao)
- Thông tin bảng HT 1 byte:
  - + bit 0..3: số bảng lượng tử QT (0..3, nếu khác là lỗi)
  - + bit 4 = 0 : bảng DC, 1: bảng AC
  - + 5..7: chưa dùng đến và có giá trị 0
- 16 bytes: Số ký hiệu dùng để mã với chiều dài từ 1..16. Tổng các bytes ấy chính là tổng số các mă, và nó phải <= 256.
- n bytes: bảng chứa ký hiệu theo thứ tự tăng chiều dài mă (n = tổng số mă )

**Chú ý** - Một đoạn đơn DHT có thể chứa nhiều HTs, mà mỗi HT có byte thông tin riêng.

**COM: Lời nhận xét/chú giải**

- \$ff, \$fe (COM)
- Chiều dài (byte thấp, byte cao) = L+2
- Nội dung: chuỗi byte chiều dài L

**SOS: Bắt đầu quét**

- \$ff, \$da (SOS)
- Chiều dài (byte thấp, byte cao) là  $6 + 2^k$  ( Số thành phần trong kiểm tra)
- Số thành phần trong kiểm tra (1 byte),  $\in [1, 4]$ , nếu khác là lỗi. Thường là 1 hoặc 3
- Mỗi thành phần: 2 bytes
- Thành phần (1 = Y, 2 = Cb, 3 = Cr, 4 = I, 5 = Q), xem SOFU
- Bảng Huffman sử dụng:
  - bit 0..3: AC table (0..3)
  - bit 4..7: DC table (0..3)
- 3 bytes tiếp: bỏ qua

**Chú ý** - Dữ liệu ảnh ( kiểm tra ) ở ngay tiếp sau đoạn SOS.

#### 5.4. Mẫu dữ liệu một tệp ảnh JPEG

```

ff d8 ff e0 00 10 4a 46 49 46 00 01 01 00 00 01 00 01 00 00 ff
fe 00 46 76 7a 83 84 85 86 b5 b6 b7 b8 b9 ba c2 03 00 04 ff db
00 43 00 08 06 06 07 07 07 09 09 08 0a 0c 14 0d 0c 0b 0b 0c 19
12 13 0e 14 1d 1a 1f 1e 1d 1a 1c 1c 20 24 2e 27 20 22 2e 23 1c
1c 28 37 29 2c 30 31 34 34 34 1f 27 39 3d 38 32 3c 2e 33 34 32
ff c0 00 0b 08 01 00 01 00 01 11 00 ff c4 00 1f 00 00 00 01 02 03 04 05 06 07
01 01 01 01 01 00 00 00 00 00 00 00 00 01 02 03 04 05 06 07
08 09 0a 0b ff c4 00 b5 10 00 02 01 03 03 02 04 03 05 06 07 08
00 00 01 7d 01 02 03 00 04 11 05 12 21 31 41 06 13 51 61 07 22
71 14 32 81 91 a1 08 23 42 b1 c1 15 52 d1 f0 24 33 62 72 82 09
0a 16 17 18 19 1a 25 26 27 28 29 2a 34 35 36 37 38 39 3a 43 44
45 46 47 48 49 4a 53 54 55 56 57 58 59 5a 63 64 65 66 67 68 69
6a 73 74 75 76 77 78 79 7a 83 84 85 86 87 88 89 8a 92 93 94 95
96 97 98 99 9a 92 a3 a4 a5 a6 a7 a8 a9 aa b2 b3 b4 b5 b6 b7 b8
b9 ba c2 c3 c4 c5 c6 c7 c8 c9 ca d2 d3 d4 d5 d6 d7 d8 d9 da e1
e2 e3 e4 e5 e6 e7 e8 e9 ea f1 f2 f3 f4 f5 f6 f7 f8 f9 fa ff da
00 08 01 01 00 00 3f 00 xx xx
xx xx xx xx xx xx ff d9

```

Nhìn vào bảng dữ liệu ảnh JPEG trên, ta thấy ngay một số vạch dấu đã giới thiệu ở trên (các vạch dấu được minh họa bởi dấu gạch dưới). Dựa vào nội dung của các vạch dấu đó ta sẽ biết được những việc cần phải làm để từ bảng dữ liệu trên ta có thể giải nén dữ liệu.

Vạch dấu	Giải thích ngắn gọn
FF D8	Start of Image (bắt đầu ảnh)
FF FE	Comments (lời bình có thể không có)
FF E0	Application data marker (vạch dấu dữ liệu ứng dụng)
FF DB	Define Quantization Table (định nghĩa bảng lượng tử )
FF C0	Start of Frame for Baseline DCT (bắt đầu khung của biến đổi cosin tuần tự )
FF C4	Define Huffman table (xác định bảng Huffman)
FF DA	Start of Data (bắt đầu dữ liệu)
XX XX	Encoded Data (dữ liệu mã hóa)
FF D9	End of Image (kết thúc ảnh)

Bất kể tệp ảnh JPEG nào cũng có thể kết xuất dưới dạng hệ số 16 và các vạch dấu kể trên cùng định dạng có thể theo dõi.

## PHỤ LỤC B

Phụ lục gồm listing của một số module chương trình chính, thực hiện các công đoạn của quá trình xử lý ảnh. Một số module phụ được lược bỏ cho tiện theo dõi.

Chương trình được viết trên C và Visual C và có thể coi như một công cụ minh họa các bước của xử lý ảnh: thể hiện cài đặt các kỹ thuật đã trình bày. Các ảnh minh họa trong cuốn sách này lấy từ kết quả thực hiện chương trình trên.

### 1. Nhóm chương trình nạp và lưu ảnh

Việc nạp ảnh từ tệp vào mảng số và lưu ảnh từ mảng lên tệp là cần thiết cho mọi chức năng xử lý. Chính vì thế, phần chương trình có tổ chức 2 modules riêng:

- BMP.H: thực hiện việc mở và đọc ảnh số từ "tệp \*.BMP" vào "mảng" 2 chiều.
- PCX.H: thực hiện việc mở và đọc ảnh số từ "tệp \*.PCX" vào "mảng" 2 chiều.
- BMP.H

// Chứa các khai báo về file ảnh BMP & các thủ tục mở file. //

```
typedef unsigned int WORD;
typedef unsigned long DWORD;
typedef unsigned char BYTE;
typedef struct tagBITMAPFILEHEADER
{
    WORD bfType;
    DWORD bfSize;
    WORD bfReserved1;
    WORD bfReserved2;
    DWORD bfoffBits;
} BITMAPFILEHEADER;
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize;
    long biWidth;
    long biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    long biXPelsPerMeter;
    long biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
```

```

} BITMAPINFOHEADER;
typedef struct tagRGBQUAD
{
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
BYTE huge *InImage;
unsigned long p,chieucao,chieurong;
int OpenBitmapFile(char *fname,int *fh,
                   BITMAPFILEHEADER *bmfh,
                   BITMAPINFOHEADER *bmihd);
int ReadBitmapPalette(int fh, int numcl, RGBQUAD *bmpals);
int GetScanLine(int fh, int BytePerLine, BITMAPINFOHEADER *bih, int x, int y);
void SetRGBPalette(int index,int red,int green,int blue);
void SetRGBPalettes(int index, int red, int green, int blue);
int GetBitmapToArray(char *fname);
extern void Error(unsigned OrderError);
***** //Mở file bitmap //
int OpenBitmapFile(char *fname,int *fh,BITMAPFILEHEADER
                   *bmfh, BITMAPINFOHEADER *bmihd)
{
    int tmp;
    *fh = _open(fname,O_RDONLY);
    if(*fh == -1) return -1; // không mở được file
    tmp = _read(*fh,bmfd,sizeof(BITMAPFILEHEADER));
    if (tmp == -1)
        | close(*fh); return -1;// lỗi đọc file
    |
    if(bmfd->bfType != 0x4D42)
    {
        |
        close(*fh);
        return -2;// không phải dạng BMP
    }
    tmp = _read(*fh,bmihd,sizeof(BITMAPINFOHEADER));
    if(tmp != sizeof(BITMAPINFOHEADER))
    {
        |
        close(*fh);
        return -2;//không phải dạng BMP
    }
    if (bmihd->biCompression == 0)
    {
        |
        bmihd->biSizeImage = (bmfd->bfSize-bmfd->bfOffBits);
    }
    return 0;
} //end of function
***** */

```

```

//Đọc bảng màu file bitmap //
int ReadBitmapPalette(int fh,int numel,RGBQUAD *bmpals)
{
    int numbyte,tmp;
    numbyte=numel*sizeof(RGBQUAD);
    lseek(fh,sizeof(BITMAPFILEHEADER)+sizeof(BITMAPINFOHEADER),
    SEEK_SET);
    tmp=_read(fh,(void*)bmpals,numbyte);
    if (tmp == -1)
        { close(fh);return -1; }
    if(tmp!=numbyte)
        { close(fh);return -2; }
    return 0;
} // end of function
//*****//
int GetScanLine(int fh,int BytePerLine,BITMAPINFOHEADER *bih)
//Đọc dòng ảnh vào mảng //
{
    BYTE *buff;
    int tmp,ppb,mask,i,j,cx,nb,cl;
    buff=(BYTE *)malloc(BytePerLine+1);
    if(buff==NULL)
        { return -3; //Không đủ bộ nhớ }
    tmp=_read(fh,buff,BytePerLine);
    if(tmp != BytePerLine)
        { return -1; }
    ppb=8/bih->biBitCount;
    mask=(1 << bih->biBitCount)-1;
    for(i=0,cx=0;i<BytePerLine;i++)
        { nb=buff[i];
            for(j=0;j<ppb;j++,cx++)
                { cl=nb>> (bih->biBitCount * (ppb-1-j));
                    cl=cl & mask;
                    InImage[p++]=cl;
                    if(cx == bih->biWidth) break;
                }
            if(cx >= bih->biWidth) break;
        }
    free(buff); return 0;
}
//*****//
void SetRGBPalette(int index,int red,int green,int blue)
{
    int r,g,b,cl;

```

```

red>>= 6; r=(red & 1) << 5; r+=(red & 2) << 1;
green >>= 6; g=(green & 1) << 4; g+=(green & 2);
blue >>= 6; b=(blue & 1) << 3; b+=(blue & 2) >> 1;
cl=r+g+b;
_AX=0x1000;
_BL=index;
_BH=cl;
    geninterrupt(0x10);
    |
//***** **** //
void SetRGBPalettes(RGBQUAD *pals,int numcl)
{
    int i;
    for(i=0;i<numcl;i++)
    {
        SetRGBPalette(i,(pals+i)->rgbRed,(pals+i)->rgbGreen,
                      (pals+i)->rgbBlue);
    }
}
//***** **** //
// Đọc File BMP vào mảng bộ nhớ //
int GetBitmapToArray(char *fname)
{
    int i,j,tmp,numcolor,BytePerLine,bmfilehand;
    BITMAPINFOHEADER bminfohdr;
    BITMAPFILEHEADER bmfilehdr;
    RGBQUAD *bmpalettes;
    tmp=OpenBitmapFile(fname,&bmfilehand,&bmfilehdr,
                       &bminfohdr);
    chieucao= bminfohdr.biHeight;
    chieurong= bminfohdr.biWidth;
    switch(tmp)
    {
        case -1: return(1); //break; //không mở được file
        case -2: return(2); //break; //tối file
    }
    numcolor=1 << (bminfohdr.biBitCount);
    BytePerLine=bminfohdr.biSizeImage/bminfohdr.biHeight;
    bmpalettes=(RGBQUAD far*) malloc(numcolor*sizeof(RGBQUAD));
    if(bmpalettes==NULL) return(3);
    tmp=ReadBitmapPalette(bmfilehand,numcolor,bmpalettes);
    if (tmp != 0) return(1);
    SetRGBPalettes(bmpalettes,numcolor);
    farfree(bmpalettes);
    lseek(bmfilehand,bmfilehdr.bfOffBits,SEEK_SET);
    if((InImage=(BYTE huge *) farcalloc(bminfohdr.biHeight*bminfohdr.biWidth,
                                         sizeof(BYTE)))==NULL)
        return(3);
    p=0;
}

```

```

for(i=0;i<bminfohdr.biHeight;i++)
{
    if (bminfohdr.biCompression==0)
        tmp=GetScanLine(bmfilehand,BytePerLine,&bminfohdr);
}
close(bmfilehand);
return(0);
}

```

• PCX.H

```

// Chứa các khai báo của File PCX
//
#define TRUE          1
#define FALSE         0
#define VIDEO         0x10
#define Enter          13
#define BackSpace      8
#define MAXSCREENWIDTH 640
#define MAXSCREENHEIGHT 480
#define BITSPERBYTE    8
#define MAXPLANES      4
#define MAXBYTESPERSCAN 640
#define MAXPALETTECOLORS 16
#define MAX256PALETTECOLORS 256
#ifndef _BYTE
#define _BYTE
typedef char BYTE;
#endif
#define NoError        0
#define EBadParms      -1
#define EFileNotFound   -2
#define EReadFileHdr    -3
#define ENotPCXFile    -4
#define ECorrupt        -5
#define EWrtFileHdr     -6
#define EWrtOutFile     -7
#define EWrtScanLine    -8
#define EPCCFile        -9
#define PCXHdrTag       0x0A
#define MaxRepCount     0x3F
#define PCX256ColorTag 0x0C
typedef struct
{
    BYTE Red;
    BYTE Green;
    BYTE Blue;
}ColorRegister;
struct PCXFileHeader
{

```

```
BYTE Header;
BYTE Version;
BYTE Encode;
BYTE BitPerPix;
unsigned X1;
unsigned Y1;
unsigned X2;
unsigned Y2;
unsigned Hres;
unsigned Vres;
};

struct PCXInfo
{
    BYTE Vmode;
    BYTE NumOfPlanes;
    unsigned BytesPerLine;
    BYTE unused[60];
};

struct ExtendedPalette
{
    BYTE ExtendedPalette;
    ColorRegister Palette[MAX256PALETTECOLORS];
};

struct PCX_File
{
    struct PCXFileHeader      PCXHeader;
    ColorRegister             Palette[MAXPALETTECOLORS];
    struct PCXInfo            Info;
} PCX_FileType;

int looping;
struct PCX_File PCXData;
unsigned ImageWidth,ImageHeight;
FILE *PCXFile;
BYTE ScanLine[MAXBYTESPERSCAN];
BYTE PixelColorNum[MAXSCREENWIDTH];
struct ExtendedPalette Color256Palette;
unsigned ImageWidth,ImageHeight,NumOfPlanes;
unsigned Is256ColorFile=FALSE;
struct PCX_File PCXData;
int ReadPCXFileHdr(char *filename,int Verbose);
static int ExpandScanLine(FILE *InFile);
unsigned InstallPCXFilePalette(void);
void DisplayPCXFile(char *FileName,int Verbose);
void Set256ColorMode(void);
void interrupt (*oldkb)();
void interrupt newkb();
int Gwrite(int x,int y,char *gtext,int inc)
{
```

```

WriteXYB(x,y,gtext,1,1,15,0,0,0);
return (x+inc+textwidth(gtext));
}
int GwriteDel(int x,int y,char *gtext,int inc)
{
WriteXYB(x,y,gtext,1,1,0,0,0,0);
return (x+inc+textwidth(gtext));
}
int Gwriteln(int x,int y,char *gtext)
{
WriteXYB(x,y,gtext,1,1,15,0,0,0);
return (y+10+textheight(gtext));
}
char *Gread(int x,int y)
{
char *gtext,text[2];
char ch;
gtext[0]=0x0;
do
{
ch=getch();
if (ch!=Enter)
{
if (ch!=BackSpace)
{
text[0]=ch;text[1]=^0;
x=Gwrite(x,y,text,0);
strcat(gtext,text);
}
else
if (strcmp(gtext,""))
{
text[0]=gtext[strlen(gtext)-1];text[1]=^0;
x=x-textwidth(text);
x=GwriteDel(x,y,"^",0);
x=x-textwidth(text);
gtext[strlen(gtext)-1]=^0;
}
}
} while (ch!=Enter);
return gtext;
}
void interrupt newkb()
{
if(inportb(0x60)==1) looping=0;
oldkb();
}
void Set256ColorMode(void)

```

```

{
union REGS regs;
regs.h.ah=0;
regs.h.al=0x13;
int86(VTDEBO,&regs,&regs);

int ReadPCXFileHdr(char *Filename,int Verbose)
{
    unsigned Index,i;
    int row=100,col=100;
    for(i=0;i<=strlen(Filename);i++) Filename[i]=toupper(Filename[i]);
    if(!strchr(Filename,'.')) strcat(Filename,".PCX");
    if((PCXFile=fopen(Filename,"rb"))==NULL)
    {
        Gwrite(col,row,"Không tìm thấy file ",0);
        return(EFileNotFoundException);
    }
    if(fread(&PCXData,sizeof(struct PCX_File),1,PCXFile)!=1)
    {
        Gwrite(col,row,"Lỗi đọc Header File ",0);
        return(EReadFileHeader);
    }
    if(PCXData.PCXHeader.Header!=PCXHdrTag)
    {
        Gwrite(col,row,"Không phải PCX File ",0);
        return(ENotPCXFile);
    }
    if(Verbose)
    {
        clrscr();
        printf("PCX Image Information for file: %s\n",Filename);
        printf("\nVersion %d\n",PCXData.PCXHeader.Version);
        printf("\nCompression %s\n",PCXData.PCXHeader.Encode==0?"none":"RLL");
        printf("\nBit per Pixel %d\n",PCXData.PCXHeader.BitPerPix);
        printf("\nX1: %d\n",PCXData.PCXHeader.X1);
        printf("\nY1: %d\n",PCXData.PCXHeader.Y1);
        printf("\nX2: %d\n",PCXData.PCXHeader.X2);
        printf("\nY2: %d\n",PCXData.PCXHeader.Y2);
        printf("\nHoriz Resolution: %d\n",PCXData.PCXHeader.Hres);
        printf("\nVert Resolution: %d\n",PCXData.PCXHeader.Vres);
        printf("\nV mode: %d\n",PCXData.Info.Vmode);
        printf("\nNumber of Planes: %d\n",PCXData.Info.NumOfPlanes);
        printf("\nByte per Scan Line of Plane :%d\n"
            ,PCXData.Info.BytesPerLine);
        printf("\n\nHit any key to proceed");
        getch();
        clrscr();
        if ((PCXData.PCXHeader.Hres==320)&&(PCXData.PCXHeader.Vres==200)&&

```

```
(PCXData.Info.NumOfPlanes==1))
Is256ColorFile=TRUE;
printf("\n\nColor Register Value of PCX file: %s\n",Filename);

if (Is256ColorFile)
for(Index=0;Index<MAX256PALETTECOLORS;Index++)
{
    printf("\nPalette[ %X ] : R=%6X G=%6X B=%6X
/n",Index,PCXData.Palette[Index].Red,
PCXData.Palette[Index].Green,PCXData.Palette[Index].Blue);
    if ((Index+1)%16==0)
    {
        getch();
        clrscr();
        if(Index!=0xFF)
        printf("\n\nColor Register Value of PCX file: %s\n", Filename);
    }
}
else
for (Index=0;Index<MAXPALETTECOLORS;Index++)
{
    printf("\nPalette[ %X ] : R=%6X G=%6X B=%6X
\n",Index,PCXData.Palette[Index].Red,
PCXData.Palette[Index].Green,PCXData.Palette[Index].Blue);
}
printf("\n\nHit any key to continue");
getch();
}
return(NoError);
}

static int ExpandScanLine(FILE *InFile)
{
register short BitNum;
register unsigned ByteNum;
register short CharRead;
unsigned InPtr,RepCount,PixelsData;
unsigned BytesToRead,PlaneNum;
unsigned ByteOffset,BitOffset;
BytesToRead=PCXData.Info.NumOfPlanes*PCXData.Info.BytesPerLine;
InPtr=0;

do
{
    CharRead=getc(InFile);
    if(CharRead==EOF) return(FALSE);
    if((CharRead & 0xC0)==0xC0)
    {
        RepCount=CharRead & ~0xC0;
```

```
CharRead=getc(InFile);
if(CharRead==EOF) return(FALSE);
while (RepCount--) ScanLine[InPtr++] = CharRead;
}
else
ScanLine[InPtr++]=CharRead;
} while (InPtr < BytesToRead);
return(TRUE);
}

unsigned InstallPCXFilePalette(void)
{
struct paletteType palette;
union REGS regs;
unsigned Index;
if(PCXData.PCXHeader.Version!=3)
{
if(!x256ColorFile)
{
for (Index=0;Index<MAX256PALETTECOLORS;Index++)
{
Color256Palette.Palette[Index].Red>>=2;
Color256Palette.Palette[Index].Green>>=2;
Color256Palette.Palette[Index].Blue>>=2;
}
regs.h.ah=0x10;
regs.h.al=0x12;
regs.x.bx=0;
regs.x.cx=MAX256PALETTECOLORS;
_ES=FP_SEG(&Color256Palette.Palette);
regs.x.dx=FP_OFF(&Color256Palette.Palette);
int86(VIDEO,&regs,&regs);
return(TRUE);
}
else
{
palette.size=MAXPALETTECOLORS;
for (Index=0;Index<MAXPALETTECOLORS;Index++)
{
palette.colors[Index]=Index;
PCXData.Palette[Index].Red>>=2;
PCXData.Palette[Index].Green>>=2;
PCXData.Palette[Index].Blue>>=2;
}
regs.h.ah=0x10;
regs.h.al=0x12;
regs.x.bx=0;
regs.x.cx=MAXPALETTECOLORS;
_ES=FP_SEG(&PCXData.Palette);
```

```
int86(VIDEO,&regs,&regs);
setallpalette(&palette);
return(TRUE);
}
}
else return(FALSE);
}
void DisplayPCXFile(char *FileName,int Verbose)
{
int i;
BYTE far *PtrScreen;
register unsigned ScanNum;
register unsigned ColNum;
unsigned OffsetDisplay[BytePerLine];
int PCXError,color,Plane;
long CurrentPos;
if((PCXError=ReadPCXFileHdr(FileName,Verbose))!=NoError)exit(PCXError);
if((PCXData.PCXHeader.X1!=0) || (PCXData.PCXHeader.Y1!=0))
{
printf("\nError POC file not PCX file\n");
exit(EPCCFile);
}

if(PCXData.PCXHeader.X2==319)
{
Set256ColorMode();
ImageWidth = 320;
ImageHeight= 200;
}
else
{
ImageWidth=640;
switch(PCXData.PCXHeader.Y2)
{
case 479: setgraphmode(VGAHI);
    ImageHeight=480;break;
case 349: setgraphmode(VGAMED);
    ImageHeight=350;break;
case 199: setgraphmode(VGALO);
    ImageHeight=200;break;
}

CurrentPos=felli(PCXFile);
fseek(PCXFile,-769,SEEK_END);
if(fread(&Color256Palette.ExtendedPalette,
sizeof(struct ExtendedPalette),1,PCXFile)==TRUE)
if(Color256Palette.ExtendedPalette==PCX256ColorTag)
Is256ColorFile=TRUE;
```

```

InstallPCXFilePalette();
fseek(PCXFile, CurrentPos, SEEK_SET);
for(ScanNum=0; ScanNum<ImageHeight; ScanNum++)
{
    if(ExpandScanLine(PCXFile) != TRUE)
    {
        closegraph();
        printf("\nScanLine corrupt in PCX file\n");
        exit(ECorrupt);
    }
    PtrScreen=MK_FP(0xA000,0);
    if(ImageWidth==320)
    {
        OffsetDisplay=ScanNum*ImageWidth;
        for (ColNum=0; ColNum<ImageWidth; ColNum++)
            PtrScreen[OffsetDisplay++]=ScanLine[ColNum];
    }
    else
    {
        BytePerLine=PCXData.Info.BytesPerLine;
        for(Plane=0; Plane<PCXData.Info.NumOfPlanes; Plane++)
        {
            outp(0x3C4,2);
            outp(0x3C5,1<<Plane);
            OffsetDisplay=ScanNum*BytePerLine;
            for (ColNum=0; ColNum<BytePerLine; ColNum++)
                PtrScreen[OffsetDisplay++]=ScanLine[Plane*BytePerLine+ColNum];
        }
        outp(0x3C4,2);
        outp(0x3C5,0x0F);
    }
}
putch(0x07);
fclose(PCXFile);
}

```

## 2. Nhóm chương trình xử lý ảnh (viết trên Turbo C).

Thực hiện các chức năng cải thiện ảnh, phát hiện biến:

- PROCESS.H
- MESSAGE.H

### • MESSAGE.H

```

// Nội dung tệp message.h — tạo giao diện trong môi trường DOS //
extern void button(title st,int centre,int x1,int y1,int x2,
                   int y2,int v,int miren,int mduoi,int mc,int mn);
extern char *fname;
/* ***** */
void manhhinh(void)

```

```

{ km mang;int i,int ch;
button("",0,0,0,getmaxx(),30,2,15,8,12,RED);
rectangle(0+2,0+2, getmaxx()-2,30-2);
button("Image Processing System ",200,0+2,0+2, getmaxx()-4,30-4, 18,15,0,WHITE);
button("",200,0,460, getmaxx(),getmaxy(),1,0,13,0,WHITE);
outtextxy(70,465,
"Chon: <ENTER> Di Chuyen: <Up><Down><Left><Right> Thoat:<ESC>");

/*
*****
void WriteText(int *x,int *y,char *s)
{ settextjustify(1,1);
outtextxy(*x,*y,s);
settextjustify(0,2);
*x+=textwidth(s); }

-----
void ReadText(int *x,int *y,char *s)
{ int i=0,char ch[2];
ch[1]=0;
while(1)
{ ch[0]=getch();
if((ch[0]==ENTER)||(ch[0]==ESCAPE)) break;
settextjustify(1,1);
WriteText(x,y,ch);
settextjustify(0,2);
s[i]=ch[0];++i;
} s[i]=0; }

-----
void Message(char *st,char *st1,int thongtin)
{ int x,y,y1=190;
void far *q; unsigned kt;
int trai=(getmaxx()/2)-(textwidth(st)/2)-10;
int cao= 100; int phai=(getmaxx()/2)+(textwidth(st)/2)+10;
kt= imagesize(trai,y1,phai,y1+cao);
if((q=(unsigned far*)farmalloc(kt))==NULL)
{ outtextxy(70,465,"Không còn vùng nhớ cho cài màn hình !");
exit(1);
}
getImage(trai,y1,phai,y1+cao,q);
button("",0,trai,y1,phai,y1+cao,2,15,15,0,7);
rectangle(trai+1,y1+1,phai-1,y1+cao-1);
rectangle(trai+3,y1+3,phai-3,y1+cao-3);
y= y1+18;x= getmaxx()/2;
WriteText(&x,&y,st1);
y= y+textheight(st1)+19;x= getmaxx()/2;
WriteText(&x,&y,st1);line(trai+3,y-13,phai-3,y-13);
if(thongtin==1)
{ y= y+textheight(st1)+15; x= getmaxx()/2-30;
ReadText(&x,&y,fname);
}
}

```

```

    }
getch();
putimage(trai,y1,q,COPY_PUT);
farfree((unsigned far*)q);
}
//-----
void Warning(void)
{
int x,y,y1=190;
char *st,*st1;
st=" *** Process Running - Please wait ! ***";
st1="WARNING !!!";
int trai=(getmaxx()/2)-(textwidth(st)/2)-10;
int cao=80; int phai=(getmaxx()/2)+(textwidth(st)/2)+10;
button("",0,trai,y1,phai,y1+cao,2,15,15,0,7);
rectangle(trai+1,y1+1,phai-1,y1+cao-1);
rectangle(trai+3,y1+3,phai-3,y1+cao-3);
y=y1+22;x=getmaxx()/2;
WriteText(&x,&y,st1);
y=y+textheight(st1)+25;x=getmaxx()/2;
WriteText(&x,&y,st);line(trai+3,y-17,phai-3,y-17);
}

```

#### • PROCESS.H

```

// Chủ các thao tác chính : Hiện ảnh, Lọc TB, Trung bình KG, Lọc Trung vị,
// Lọc Sobel, Lọc Hormomorphic
int Averaging(BYTE huge *Image,unsigned cotbd,unsigned hangbd,unsigned rong,
unsigned cao,unsigned avecol,
unsigned averow,double *matrac);
void DisplayImageInBuf(BYTE huge *Image,unsigned hang,
unsigned cot);
void DisplayPCXInBuf(BYTE huge *Image,unsigned hang,
unsigned cot);
BYTE ReadPixel(BYTE huge *Image,unsigned cot,unsigned hang);
void WritePixel(BYTE huge *Image,unsigned hang,unsigned cot,unsigned color);
void MedianFilter(BYTE huge *Image,unsigned hangbd,unsigned cotbd, unsigned
rong,unsigned cao);
int Averaging(BYTE huge *Image,unsigned cotbd,unsigned hangbd, unsigned rong,
unsigned cao,unsigned avecol, unsigned averow, double *matran);
int SpatialConvolution(BYTE huge *Image,unsigned cotbd,
unsigned hangbd, unsigned rong,unsigned cao, unsigned KernelCol, unsigned
KernelRow, short *Kernel);
extern void Error(unsigned OrderError);
extern void Message(char *st,char *st1,int thongtin);
void Threshold(BYTE huge *Image,unsigned col,unsigned row,
unsigned rong,unsigned cao);
//*****//
//Hiển thị ảnh BMP //
void DisplayImageInBuf(BYTE huge *Image,unsigned hang,unsigned cot)

```

```

    { int i,j;
    unsigned long p;
    p=0;
    for(i= cot;i>0;i--)
    for(j=0;j< hang;j++)
        {
        putpixel(j,i,Image[p++]);
        }
    }
//***** ****
//Đọc 1 pixel từ buffer
BYTE ReadPixel(BYTE huge *Image,unsigned cot,unsigned hang)
{unsigned long p;
 p= hang;
 p+= chieurong;
 p+= cot;
 return(Image[p]);}
//Ghi 1 pixel vào buffer
void WritePixel(BYTE huge *Image,unsigned hang,unsigned cot,unsigned color)
{ unsigned long p;
 if ((cot<=chieurong) && (hang<=chieucao))
 { p= hang;
   p+= chieurong;
   p+= cot;
   Image[p]= color;
 }
 else
 { Error(16);
   getch();
 }
}
void SortPixel(BYTE *a)
// Sắp xếp thứ tự hai pixels
{ BYTE i,j,tg;
 for(i= 0;i<8;i++)
 for(j=i;j<9;j++)
 if(a[i]<a[j])
 { tg= a[i];a[i]= a[j];a[j]= tg; }
}

// Lọc trung vị (medianfilter)-lọc phi tuyến //
void MedianFilter(BYTE huge *Image,unsigned hangbd,unsigned cotbd,      unsigned
rong,unsigned cao)
{ register unsigned i,j,k,l,pixelindex;
 BYTE *pixelvalues;
 pixelvalues=(BYTE *)calloc(9,(unsigned)sizeof(BYTE));
 if(pixelvalues==NULL)
 { Error(16);
 }
}

```

```

        getch(); exit(0);
    for(i=hangbd;i<cao-1;i++) //maxy
    {
        for(j=cothd;j<rong-1;j++) //maxx
        {
            pixelindex=0;
            for(k=0;k<3;k++)
                for(l=0;l<3;l++)
                    pixelvalues[pixelindex++]=ReadPixel(Image,j+k,i+l);
            SortPixel(pixelvalues); //sap xep*
            WritePixel(Image,i,j,pixelvalues[5]); //doc 9 ghi 1
        }
    } //end of for
    free(pixelvalues);
}

// ****
// Lọc trung bình trọng số (Spatial Averaging)
int Averaging(BYTE huge *Image,unsigned cothd,unsigned hangbd,unsigned rong,
unsigned cao,unsigned avecol, unsigned averow,double *matran)
{
    register unsigned i,j,k,l;/l=(avecol+1)/2;
    double *AvePir;
    double tong;
    for(i=hangbd;i<cao;i++)
        for(j=cothd;j<rong;j++)
        {
            tong=0;
            AvePir=matran;
            for(k=0;k<avecol;k++)
                for(l=0;l<averow;l++)
                    tong+=(ReadPixel(Image,j-k+2,i-l+2)*(*AvePir++));
            tong=fabs(tong); //10 là tham số chia các giá trị của lọc
            WritePixel(Image,i,j,(BYTE)tong); //doc 9 ghi 1
        } //end of for
    return(0);
}

// ****
// Lọc Spatial Convolution
int SpatialConvolution(BYTE huge *Image,unsigned cothd,unsigned hangbd, unsigned
rong,unsigned cao,unsigned KernelCol, unsigned KernelRow, short *Kernel)
{
    register unsigned i,j,k,l;
    short *KernelPir;
    long tong;
    for(i=hangbd;i<cao;i++)
        for(j=cothd;j<rong;j++)
        {
            tong=0L;
            KernelPir=Kernel;
            for(k=0;k<KernelCol;k++)

```

```

    for(l=0;l<KernelRow;l++)
        tong+=(ReadPixel(Image,j-k+2,l-l+2)*(*KernelPtr++));

    tong= labs(tong);
    tong= (tong<0) ? 0:tong;
    tong= (tong>64) ? 64:tong; //max 63 vi day la anh 16 mau
    WritePixel(Image,i,j,(BYTE)tong); //doc 9 ghi 1
} //end of for
return(0);
}

int Max(int value1, int value2)
{
    return ((value1 > value2) ? value1 : value2);
}

/* lọc Sobel làm tròn cạnh */
int EdgeDetect(BYTE huge *Image,unsigned cotbd,unsigned hangbd,
               unsigned rong,unsigned cao)
{
    register unsigned i,j;unsigned A,B,C,D,E,F,G,H,I;
    unsigned LineAEIAveAbove,LineAEIAveBelow,LineAEIMaxDif;
    unsigned LineBEHAveAbove,LineBEHAveBelow,
    LineBEHMaxDif;
    unsigned LineCEGAveAbove,LineCEGAveBelow,
    LineCEGMaxDif;
    unsigned LineDEFAveAbove,LineDEFAveBelow,
    LineDEFMaxDif;
    unsigned MaxDif;
    cotbd+= 1;hangbd+= 1;rong-= 2; cao-= 2;
    rong= rong+cotbd; cao= cao+hangbd;
    for(i=hangbd;i<cao;i++)
        for (j=cotbd;j<rong;j++)
        {
            A= ReadPixel(Image,j-1,i-1);
            B= ReadPixel(Image,j,i-1);
            C= ReadPixel(Image,j+1,i-1);
            D= ReadPixel(Image,j-1,i);
            E= ReadPixel(Image,j,i);
            F= ReadPixel(Image,j+1,i);
            G= ReadPixel(Image,j-1,i+1);
            H= ReadPixel(Image,j,i+1);
            I= ReadPixel(Image,j+1,i+1);
        }
    // Tính các giá trị sai phân
    LineAEIAveBelow= (D+G+H)/3;
    LineAEIAveAbove= (B+C+F)/3;
    LineAEIMaxDif= abs(LineAEIAveBelow- LineAEIAveAbove);
    LineBEHAveBelow= (A+D+G)/3;
    LineBEHAveAbove= (C+F+I)/3;
    LineBEHMaxDif= abs(LineBEHAveBelow- LineBEHAveAbove);
    LineCEGAveBelow= (F+H+I)/3;
    LineCEGAveAbove= (A+B+D)/3;
}

```

```

LineCEGMaxDif= abs(LineCEGAveBelow- LineCEGAveAbove);
LineDEFAveBelow= (G+H+I)/3;
LineDEFAveAbove= (A+B+C)/3;
LineDEFMaxDif= abs(LineDEFAveBelow- LineDEFAveAbove);
/*Tim giá trị lớn nhất*/
MaxDif= Max(LineAEIMaxDif,LineBEHMaxDif);
MaxDif= Max(LineCEGMaxDif,MaxDif);
MaxDif= Max(LineDEFMaxDif,MaxDif);
if (MaxDif>=16)
    WritePixel(Image,i,j,63);
else
    WritePixel(Image,i,j,E);
} //end of for
return(0);
}

// Lọc Homomorphic - Lọc đồng hình
//
void Homomorphic(BYTE huge *Image,unsigned hangbd,unsigned cotbd, unsigned rong,
unsigned cao, double *matran)
{
register unsigned i,j,k,l,i1,j1;
double *matranptr;
double tong;t=0;
// Lấy logarit cho tất cả các điểm ảnh //
for(i=hangbd;i<cao;i++)
    for (j=cotbd;j<rong;j++)
        {
        tong= ReadPixel(Image,j,i);
        if(tong<=0) tong=0.1;
        tong= log(tong);tong= fabs(tong);
        WritePixel(Image,i,j,(BYTE)tong);
        }
    for(i=hangbd+1;i<cao-1;i++)
        {
        tientrinh(i);i1=i-1;
        for (j=cotbd+1;j<rong-1;j++)
            {
            tong= 0.0;j1=j-1;
            matranptr= matran;
            for(k=0;k<3;k++)
                for(l=0;l<3;l++)
                    tong+=(ReadPixel(Image,j1+k,i1+l)*(*matranptr++));
            tong= exp(tong);
            tong=fabs(tong);
            tong/=(double)(l<<0);
            tong= (tong<0) ? 0:tong;
            tong= (tong>255) ? 255:tong; //max là 63 vì là ảnh 16 màu
            WritePixel(Image,i1,j1,(BYTE)tong);//đọc 9 ghi 1
            }
        }
}
}

```

```

//=====
void Addnoise(BYTE huge *thelimage)
{
    register i,r;
    for(i=0;i<1000;i++)
        [thelimage[(long)random(chieucao)*chieurong+random(chieurong)] = 200+random(20);
        putpixel(r+rand()%chieurong,r+59+rand()%chieucao,200+random(20));
    }
}

//      Tính Histogram của ảnh
void TinhHisto(BYTE huge *Image,unsigned rong,unsigned cao,
long HISTO[])
{
    register unsigned i,j;
    for(i = 0; i < 256; i++) HISTO[i] = 0;
    for(i = 0; i < cao; i++)
        for(j = 0; j < rong; j++)
            HISTO[Image[(long)i * rong + j]]++;
    return ;
}

//=====
//      Hiển Histogram
void VeHisto(long HISTO[], unsigned X, unsigned Y,BYTE New)
{
    long MaxPixel = 0;
    register unsigned i;
    for(i = 0; i < 235; i++)
        if(HISTO[i] > MaxPixel) MaxPixel = HISTO[i];
    setcolor(15);
    for(i = 0;i < 235; i++)
    {
        line(X+i,Y, X+i, Y-(int)400*HISTO[i]/(2.5*MaxPixel));
    }
    unsigned y1=397; unsigned x1=381;
    line(x1,y1,626,y1);
    line(626,2,626,y1);
    setcolor(0);
    line(x1,2,626,2);
    line(x1,2,x1,y1);
    if(New!= 1)
    {
        setcolor(15);line(x1,Y+26,626,Y+26);
        setcolor(0);line(x1,Y+27,626,Y+27);
        outtextxy(x1,Y+6," 0      127      255");
        outtextxy(x1,Y+16," P/I Histogram Origin");
    } else
    {
        outtextxy(x1,Y+6," 0      127      255");
        outtextxy(x1,Y+16," P/I Histogram New");
    }
}

```

```

}
return ;
}

// Biến đổi lược đồ
void HistoModify(BYTE huge *Image,unsigned rong,unsigned cao, long HISTO[])
{
register int i,j,l,k;
unsigned Limited,Tansuat;
long Heso,TotalPixel,HIS[256];
unsigned char LUT[256];
TinhHisto(Image,rong,cao,HISTO);
for (l = 0; l < 256; l++)
    LUT[l] = l;
Limited = 256;
TotalPixel = (long)rong*cao;
Tansuat = TotalPixel / Limited;
for(l = 0; l < 256; l++)
{
    Heso = 0;
    for(k = 0; k < l; k++)
        Heso += HISTO[k];
    Heso += HISTO[l]/2;
    LUT[l] = Heso/Tansuat;
}
for(l=0;l<256;l++)
    HIS[l] = 0;
for(l=0;l<256;l++)
    HIS[LUT[l]] += HISTO[l];
//Tạo bảng màu mực xám;
for(i = 0; i < cao; i++)
    for(j = 0; j < rong; j++)
    {
        if(chieurong>=640)putpixel(j/1.7, i/1.2, LUT[Image](long) i * rong + j]/16);
        else
            if((chieurong>=512)&&(chicucao>=480))
                putpixel(j/1.35,i/1.2,LUT[Image](long)i * chieurong + j]/16);
            else putpixel(j, i, LUT[Image](long) i * rong + j]/16);
    }
VeHisto(HIS,384,370,1);
return ;
}
/*=====

```

```

void CallHistoModify(BYTE huge *Image)
{
    long Xacsuat[256];
    unsigned i,j;
    TinhHisto(Image,chieurong,chieucao,Xacsuat);
    setviewport(5,60,getmaxx()-5,getmaxy()-22,1);
}
```

```

for(i=0;i<chieucao;i++)
for(j=0;j<chieurong;j++)
{
if(chieurong>=640) putpixel(j/1.7,i/1.2,Image|(long)i * chieurong + j]/16);
else
if((chieurong>=512)&&(chieucao>=480))
putpixel(j/1.35,i/1.2,Image|(long)i * chieurong + j]/16);
else
putpixel(j,i,Image|(long)i * chieurong + j]/16);
}
VeHisto(Xacsuat,384,173,0); getch();
HistoModify(Image,chieurong,chieucao,Xacsuat);
setviewport(0,0,getmaxx(),getmaxy()-20,1);
}

//******/
```

static unsigned Density[256];

BYTE Bayenmat[4][4]=

10,8,2,10,
12,4,14,6,
3,11,1,9,
15,7,13,5);

```

/*-----*/
void InitDensity(void)
{
    unsigned ColorEntry,ColorNum;
    unsigned Intensity;
    struct paletteType palette;
    static union REGS regs;
    getpalette(&palette);
    for(ColorEntry=0;ColorEntry<palette.size;ColorEntry++)
    {
        ColorNum= palette.colors[ColorEntry];
        regs.h.ah= 0x10; //get the color components
        regs.h.al= 0x15; //color register
        regs.x.bx= ColorNum;
        int86(0x10,&regs,&regs);
        if((regs.h.dh== regs.h.ch)&&(regs.h.ch== regs.h.cl))
            Intensity= regs.h.dh*100;
        else
        {
            Intensity= ((unsigned) regs.h.dh*30); //thanh phan red
            Intensity+= ((unsigned) regs.h.ch*59); //thanh phan green
            Intensity+= ((unsigned) regs.h.cl*11); //thanh phan blue
        }
        Density[ColorEntry]= (unsigned)((double)Intensity/100); //chu ý
    } //end for
    for(ColorEntry=0;ColorEntry<palette.size;ColorEntry++)
        setpalette(ColorEntry,Density[ColorEntry]);
} //end void
//******/
```

```

Use BayerMatrix for Dithering
***** */

void Dithering(BYTE huge *Image)
{ register unsigned currentrow,matrixrow;
register BYTE Intensity;
unsigned Col;
for (Col=0;Col<chieucao;Col++)
for (currentrow=0;currentrow<chieurong;currentrow++)
{
    matrixrow=currentrow % 4;
    Intensity= Density[ReadPixel(Image,Col,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][3])
        WritePixel(Image,Col,currentrow,128);
    Intensity= Density[ReadPixel(Image,Col-1,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][2])
        WritePixel(Image,Col,currentrow,64);
    Intensity= Density[ReadPixel(Image,Col-2,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][1])
        WritePixel(Image,Col,currentrow,32);
    Intensity= Density[ReadPixel(Image,Col-3,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][0])
        WritePixel(Image,Col,currentrow,16);
    Intensity= Density[ReadPixel(Image,Col-4,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][3])
        WritePixel(Image,Col,currentrow,8);
    Intensity= Density[ReadPixel(Image,Col-5,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][2])
        WritePixel(Image,Col,currentrow,4);
    Intensity= Density[ReadPixel(Image,Col-6,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][1])
        WritePixel(Image,Col,currentrow,2);
    Intensity= Density[ReadPixel(Image,Col-7,currentrow)]; /*brightness
    if (Intensity>Bayermat[matrixrow][0])
        WritePixel(Image,Col,currentrow,1);
}
}

void InitLut(BYTE *Table)
{ register unsigned i;
for(i=0;i<64;i++)
    Table[i]=i;
}

void Update(BYTE huge *Image,unsigned col,unsigned row,
           unsigned rong,unsigned cao,BYTE *Table)
{ register unsigned hang,cot;
for(hang= row;cot< cao-1;hang++)
    for(cot= col;cot< rong-1;cot++)

```

```

    WritePixel(Image,hang,cot,Table[ReadPixel(Image,hang,cot)]);
}
//*****
void Threshold(BYTE huge *Image,unsigned col,unsigned row,
               unsigned rong,unsigned cao)
{
    register unsigned i;
    BYTE Table[64];
    for(i=0;i<32;i++) Table[i]= 0;
    for(i=32;i<64;i++) Table[i]= 15;
    Update(Image,col,row,rong,cao,Table);
}
/* Toàn bộ các mảng trên được gọi bởi hàm main dưới đây */
void main()
{
    data();DoHoa();
    manhinh();
    menu();
    closegraph();
}
//*****

```

### 3. Nhóm chương trình biến đổi và nén ảnh (viết trên Visual C)

#### *Chương trình tạo giao diện*

- Modul Util.cpp

```

#include "StdAfx.h"
#include "Util.h"
#include "Resource.h"
#include <commctrl.h>
#include <string.h>
#include "ImgPro.h"
//----- Mở File ảnh -----
BOOL OpenDlg(HWND hWnd,LPSTR pFilter,LPSTR FileName)
{
    BOOL fGetName;
    OPENFILENAME ofn;
    FileName[0] = '\0';
    ofn.lStructSize = sizeof(OPENFILENAME);
    ofn.hwndOwner = hWnd;
    ofn.hInstance = NULL;
    ofn.lpstrFilter = pFilter;
    ofn.lpstrCustomFilter = NULL;
    ofn.nMaxCustFilter = 0;
    ofn.nFilterIndex = 1;
    ofn.lpstrFile = FileName;
    ofn.nMaxFile = 256;
    ofn.lpstrFileTitle = NULL;
    ofn.nMaxFileTitle = 0;
    ofn.lpstrInitialDir = NULL;

```

```

ofn.lpszTitle = (LPSTR)"Chọn file ảnh cần mở";
ofn.Flags = OFN_HIDEREADONLY | OFN_FILEMUSTEXIST;
ofn.nFileOffset = 0;
ofn.nFileExtension = 0;
ofn.lpszDefExt = "*";
ofn.lCustData = 0;
ofn.lpTemplateName = NULL;
fGetName = GetOpenFileName(&ofn);
if(!fGetName) return FALSE;
return TRUE;
|
//-----Lưu theo dạng -----//
BOOL SaveAsDlg(HWND hWnd,LPSTR pFilter,LPSTR FileName,LPSTR DefExt)
{
    FileName[0] = '0';
    BOOL fGetName;
    OPENFILENAME ofn;
    ofn.lStructSize = sizeof(OPENFILENAME);
    ofn.hwndOwner = hWnd;
    ofn.hInstance = NULL;
    ofn.lpstrFilter = pFilter;
    ofn.lpstrCustomFilter = NULL;
    ofn.nMaxCustFilter = 0;
    ofn.nFilterIndex = 1;
    ofn.lpszFile = FileName;
    ofn.nMaxFile = 256;
    ofn.lpszFileTitle = NULL;
    ofn.nMaxFileTitle = 0;
    ofn.lpszInitialDir = NULL;
    ofn.lpszTitle = (LPSTR)"Ghi ra File";
    ofn.Flags = OFN_HIDEREADONLY;
    ofn.nFileOffset = 0;
    ofn.nFileExtension = 0;
    ofn.lpszDefExt = DefExt;
    ofn.lCustData = 0;
    ofn.lpTemplateName = NULL;
    fGetName = GetSaveFileName(&ofn);
    if(!fGetName) return FALSE;
    FILE *Data;
    INT Replaced;
    Data = fopen(FileName,"rb");
    if(Data!=NULL)
    {
        Replaced = MsgBox(hWnd,111,MB_YESNO);
        if(Replaced==IDYES)
            fclose(Data);
        else
    }
}

```

```

        fclose(Data);
        return FALSE;
    }
}

return TRUE;
}

HPALETTE CrPal(UINT ColorsNum,PALETTEENTRY *PalEntries)
{
    HANDLE hMemPal;
    HPALETTE hPalette;
    LOGPALETTE far *LogPal;
    hMemPal = LocalAlloc(LMEM_MOVEABLE,sizeof(LOGPALETTE) +
    ColorsNum * sizeof(PALETTEENTRY));
    LogPal = (LOGPALETTE far *)LocalLock(hMemPal);
    LogPal->palNumEntries = ColorsNum ;
    LogPal->palVersion = 0x300;
    register int i;
    for(i=0;i<ColorsNum;i++)
        LogPal->palPalEntry[i] = PalEntries[i];
    hPalette = CreatePalette(LogPal);
    LocalUnlock(hMemPal);
    LocalFree(hMemPal);
    if (!hPalette) return NULL;
    return hPalette;
}

HANDLE UserCrPal(CHAR pCode,HPALETTE hOldPal,UINT ColorsNum)
{
/*      pCode  = 1 Red
           = 2 Green
           = 3 Blue
           = 11 Gray
           = 12 Inverse
-----*/
    INT i;
    HANDLE hOldPalEntry;
    LPPALETTEENTRY OldPal;
    hOldPalEntry = (PALETTEENTRY*)LocalAlloc(GMEM_MOVEABLE,
    ColorsNum*sizeof(PALETTEENTRY));
    OldPal = (PALETTEENTRY*)LocalLock(hOldPalEntry);
    GetPaletteEntries(hOldPal,0,ColorsNum,OldPal);
    switch(pCode)
    {
        case 1:
            for(i=0;i<ColorsNum;i++)
            {
                OldPal[i].peGreen = 0;
                OldPal[i].peBlue = 0;
            }
    }
}

```

```

        break;
    case 2:
        for(i=0;i<ColorsNum;i++)
        {
            OldPal[i].peRed = 0;
            OldPal[i].peBlue = 0;
        }
        break;
    case 3:
        for(i=0;i<ColorsNum;i++)
        {
            OldPal[i].peRed = 0;
            OldPal[i].peGreen = 0;
        }
        break;
    case 11:
        for(i=0;i<ColorsNum;i++)
        {
            OldPal[i].peRed = i;
            OldPal[i].peGreen = i;
            OldPal[i].peBlue = i;
        }
        break;
    case 12:
        for(i=0;i<ColorsNum;i++)
        {
            OldPal[i].peRed = ColorsNum - OldPal[i].peRed;
            OldPal[i].peGreen = ColorsNum - OldPal[i].peGreen;
            OldPal[i].peBlue = ColorsNum - OldPal[i].peBlue;
        }
        break;
    default:
        break;
    }
    LocalUnlock(hOldPalEntry);
    return hOldPalEntry;
}
/*-----Vẽ lại-----*/
BOOL Display(HWND hWnd,HBITMAP hBitmap,HPALETTE hPalette)
{
    HDC hDC,hMemDC;
    HANDLE OldBmp;
    RECT R;
    if(hBitmap == NULL) return FALSE;
    if(hPalette == NULL) return FALSE;
    hDC = GetDC(hWnd);
    hMemDC = CreateCompatibleDC(hDC);
    SelectPalette(hMemDC,hPalette,FALSE);
}

```

```

UnrealizeObject(hPalette);
RealizePalette(hDC);
SelectPalette(hMemDC,hPalette, FALSE);
OldBmp = SelectObject(hMemDC,hBitmap);
GetClientRect(hWnd,&R);
BitBlt(hDC,0,0,R.right,R.bottom,hMemDC,0,0,SRCOPY);
SelectObject(hMemDC,OldBmp);
DeleteDC(hMemDC);
ReleaseDC(hWnd,hDC);
return TRUE;
}

/*-----Hộp đối thoại-----*/
extern HINSTANCE hInst;
int MsgBox(HWND hWnd,int MsgID,int MsgStyle)
{
    int RetCode;
    const int Len = 128;
    TCHAR Str[Len];
    LoadString(hInst,MsgID,Str,Len);
    RetCode = MessageBox(hWnd,Str,"Xử lý ảnh",MsgStyle);
    return RetCode;
}

/* ===== Vẽ Menu =====*/
BOOL UpdateMenu(HWND hWnd,int UCode,int status)
{
/*
UCode = 1 : IDM_OPEN
UCode = 2 : IDM_FFT
UCode = 3 : IDM_FCT
*/
    HIMENU hMenu;
    hMenu = GetMenu(hWnd);
    int UPDSTATUS = MF_GRAYED;
    if(status == 1) UPDSTATUS = MF_ENABLED;
    switch(UCode)
    {
        case 1:
            EnableMenuItem(hMenu, IDM_RLE, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_LZW, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_HUFFMAN, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_JPEG, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_FFT, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_FCT, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_STDFILTER, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_REOPEN, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_NORMVIEW, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_REDVIEW, UPDSTATUS);
            EnableMenuItem(hMenu, IDM_GREENVIEW, UPDSTATUS);
}
}

```

```
EnableMenuItem(hMenu, IDM_BLUEVIEW, UPDSTATUS);
EnableMenuItem(hMenu, IDM_GRAYVIEW, UPDSTATUS);
EnableMenuItem(hMenu, IDM_INVERSEVIEW, UPDSTATUS);
EnableMenuItem(hMenu, IDM_IMGINFO, UPDSTATUS);
EnableMenuItem(hMenu, IDM_SAVEAS, UPDSTATUS);
EnableMenuItem(hMenu, IDM_FFTINV, UPDSTATUS);
EnableMenuItem(hMenu, IDM_FCTINV, UPDSTATUS);
break;
case 2:
    EnableMenuItem(hMenu, IDM_FFT, UPDSTATUS);
    UPDSTATUS = !UPDSTATUS;
    EnableMenuItem(hMenu, IDM_FFTINV, UPDSTATUS);
case 3:
    EnableMenuItem(hMenu, IDM_FCT, UPDSTATUS);
    UPDSTATUS = !UPDSTATUS;
    EnableMenuItem(hMenu, IDM_FCTINV, UPDSTATUS);
default:
    break;
}
return TRUE;
}

/*=====Others=====*/
int GetItemInt(HWND hDig,int IDItem)
{
    BOOL Succes;
    int Result;
    Result = GetDlgItemInt(hDig, IDItem, &Succes, TRUE);
    if(!Succes) return 0;
    return Result;
}

BOOL ImageDisk(HANDLE hImage,UINT Size, const char* FileName,BOOL Dir)
{
    UCHAR *Image;
    FILE *Data;
    Image = (UCHAR *) GlobalLock(hImage);
    if(Dir)
    {
        Data = fopen(FileName, "wb");
        if(Data == NULL) return FALSE;
        fwrite(Image,Size,1,Data);
    }
    else
    {
        Data = fopen(FileName, "rb");
        if(Data == NULL) return FALSE;
        fread(Image,Size,1,Data);
    }
    fclose(Data);
```

```

    GlobalUnlock(hImage);
    return TRUE;
}

int GetStdLen(int Len)
{
    int i,NLevel;
    i = Len - 1 ;
    NLevel = 0;
    while(i)
    {
        i >>= 1;
        NLevel++;
    }

    return 1<<NLevel;
}

```

***Biến đổi ánh: Biến đổi FFT xuôi-ngược*****• Modul FFT.cpp**

```

#include "StdAfx.h"
#include "FFT.h"
#include "Util.h"
#include <math.h>
typedef struct
{
    double real;
    double image;
}COMPLEX;
extern WORD mWidth,mHeighth,rWidth;
extern HANDLE hDImage;
BOOL BitRevFCT(INT Len);
HANDLE hSinCos;
COMPLEX *W;
WORD *BitLoc;
BOOL InitFFT(int Len);
BOOL FFT_DIR(COMPLEX * Array,int Len,BOOL Inv);
BOOL FFT2D()
{
    HANDLE hTmplImage;
    CHAR TempPath[256];
    UCHAR far* Image;
    COMPLEX *RowBuff,*ColBuff,far *TmplImage;
    int size,StdW,StdH,StdW2,StdH2;
    register int r,c,i;
    StdW = GetStdLen(mWidth);
    StdH = GetStdLen(mHeighth);
    StdW2 = StdW>>1;
    StdH2 = StdH>>1;

```

```
size = StdW * StdH;
hTmpImage = (COMPLEX far*)
GlobalAlloc(GMEM_MOVEABLE, size*sizeof(COMPLEX));
if(hTmpImage==NULL)
{
    MessageBox(NULL,114,MB_OK);
    return FALSE;
}
TmplImage = (COMPLEX far*)GlobalLock(hTmpImage);
Image = (UCHAR far*)GlobalLock(hDImage);
RowBuff = (COMPLEX*)LocalAlloc(GMEM_FIXED,
StdW*sizeof(COMPLEX));
if(RowBuff==NULL)
{
    MessageBox(NULL,114,MB_OK);
    return FALSE;
}
if(!InitFFT(StdW)) return FALSE;
W = (COMPLEX*)LocalLock(hSinCos);
BitRevFCT(StdW);
for(r = 0;r<mHeighth;r++)
{
    for(c=0;c<mWidth;c++)
    {
        RowBuff[c].real = (double)Image[r*rWidth + c];
        RowBuff[c].image = 0.0;
    }
    for(c=mWidth;c<StdW;c++)
    {
        RowBuff[c].real = 0.0;
        RowBuff[c].image = 0.0;
    }
    FFT_DIF(RowBuff,StdW,FALSE);
    for(c=0;c<StdW;c++)
        TmplImage[r*StdW + c] = RowBuff[c];
}
LocalFree(RowBuff);
GlobalUnlock(hDImage);
LocalUnlock(hSinCos);
LocalFree(hSinCos);
LocalFree(BitLoc);
ColBuff = (COMPLEX*)LocalAlloc(GMEM_FIXED, StdH*sizeof(COMPLEX));
if(!InitFFT(StdH)) return FALSE;
W = (COMPLEX*)LocalLock(hSinCos);
BitRevFCT(StdH);
for(c = 0;c<StdW;c++)
{
    for(r=0;r<StdH;r++)
        ColBuff[c].real = (double)Image[c*mWidth + r];
```

```

        ColBuff[StdH - r - 1] = TmplImage[r*StdW + c];
        FFT_DIF(ColBuff,StdH, FALSE);
        for(r=0;r<StdH;r++)
            TmplImage[r*StdW + c] = ColBuff[StdH - r - 1];
    }
    LocalFree(ColBuff);
    LocalUnlock(hSinCos);
    LocalFree(hSinCos);
    LocalFree(BitLoc);
    FILE *Data;
    GetTempPath(256,TempPath);
    strcat(TempPath,"\\fffdst.ipt");
    Data = fopen(TempPath,"wb");
    if(Data == NULL) return FALSE;
    fwrite(TmplImage,sizeof(COMPLEX),size,Data);
    fclose(Data);
    double max = 0.0;
    for(i=0;i<size;i++)
    {
        TmplImage[i].real = (double)sqrt(
            TmplImage[i].real*TmplImage[i].real +
            TmplImage[i].image*TmplImage[i].image );
        if(TmplImage[i].real>max) max = TmplImage[i].real;
    }
    Image = (UCHAR far*)GlobalLock(hDImage);
    double coef;
    coef = 255.0/log((double)(max + 1.0));
    int mW2,mH2;
    mW2 = (mWidth+1)>>1;
    mH2 = mHeight>>1;
    for(r=0;r<mH2;r++)
        for(c=0;c<mW2;c++)
        {
            Image[r*rWidth + c] = 255-(UCHAR)(0.5+
                log((double)TmplImage[(StdH-mH2+r)*StdW + (StdW-mW2+c)].real+
                    1.0)*coef);
            Image[r*rWidth + (mW2+c)] = 255-(UCHAR)(0.5+
                log((double)TmplImage[(StdH-mH2+r)*StdW + c].real+1.0)*coef);

            Image[(mH2+r)*rWidth + c] = 255-(UCHAR)(0.5+
                log((double)TmplImage[r*StdW + (StdW-mW2+c)].real+1.0)*coef);
            Image[(mH2+r)*rWidth + (mW2+c)] = 255
                -(UCHAR)(0.5+log((double)TmplImage[r*StdW + c].real+1.0)*coef);
        }
    WORD Line1,Line2;
    Line1 = mHeight-1;
    Line2 = mHeight-2;
    if(mHeight&1)

```

```
|  
|    for(c=0;c<mWidth;c++)  
|        Image[Line1*rWidth+c] = Image[Line2*rWidth+c];  
|    }  
|    GlobalUnlock(hTmplImage);  
|    GlobalFree(hTmplImage);  
|    GlobalUnlock(hDImage);  
|    return TRUE;  
|  
BOOL IFFT2D()  
{  
    HANDLE hTmplImage;  
    CHAR TempPath[256];  
    HANDLE hRatio;  
    UCHAR far* Image;  
    COMPLEX *RowBuff,*ColBuff,far *TmplImage;  
    double far *Ratio;  
    int size,StdW,StdH,StdW2,StdH2;  
    register int r,c,i;  
    FILE *Data;  
    rWidth = mWidth;  
    if(mWidth%4) rWidth = mWidth + 4 - mWidth%4;  
    StdW = GetStdLen(mWidth);  
    StdH = GetStdLen(mHeighth);  
    size = StdW * StdH;  
    GetTempPath(256,TempPath);  
    strcat(TempPath,"~fftdata.ipt");  
    Data = fopen(TempPath,"rb");  
    if(Data == NULL)  
    {  
        MsgBox(NULL,113,MB_OK);  
        return FALSE;  
    }  
    hTmplImage = (COMPLEX far*) GlobalAlloc (GMEM_MOVEABLE,  
    size*sizeof(COMPLEX));  
    if(hTmplImage==NULL)  
    {  
        MsgBox(NULL,114,MB_OK);  
        fclose(Data);  
        return FALSE;  
    }  
    TmplImage = (COMPLEX far*)GlobalLock(hTmplImage);  
    fread(TmplImage,sizeof(COMPLEX),size,Data);  
    fclose(Data);  
    remove(TempPath);  
    GetTempPath(256,TempPath);  
    strcat(TempPath,"~ratio.ipt");  
    Data = fopen(TempPath,"rb");
```

```

if(Data == NULL)
{
    MsgBox(NULL,113,MB_OK);
    return FALSE;
}
UINT RatioSize;
RatioSize = rWidth * mHeight;
hRatio = (double far*);
GlobalAlloc(GMEM_MOVEABLE,RatioSize*sizeof(double));
if(hRatio == NULL)
{
    MsgBox(NULL,114,MB_OK);
    return FALSE;
}
Ratio = (double far*)GlobalLock(hRatio);
freadd(Ratio,sizeof(double),RatioSize,Data);
fclose(Data);
remove(TempPath);
int mW2,mH2;
StdH2 = StdH>>1;
StdW2 = StdW>>1;
mW2 = (mWidth+1)>>1;
mH2 = mHeight>>1;
for(r=0;r<mH2;r++)
for(c=0;c<mW2;c++)
{
    TmplImage[(StdH-mH2+r)*StdW + (StdW-mW2+c)].real *=
    Ratio[r*rWidth + c];
    TmplImage[(StdH-mH2+r)*StdW + (StdW-mW2+c)].image *=
    Ratio[r*rWidth + c];
    TmplImage[(StdH-mH2+r)*StdW + c].real *= Ratio[r*rWidth +
    (mW2+c)];
    TmplImage[(StdH-mH2+r)*StdW + c].image *= Ratio[r*rWidth +
    (mW2+c)];
    TmplImage[r*StdW + (StdW-mW2+c)].real *= Ratio[(mH2+r)*rWidth +
    c];
    TmplImage[r*StdW + (StdW-mW2+c)].image *= Ratio
    [(mH2+r)*rWidth + c];
    TmplImage[r*StdW + c].real *= Ratio[(mH2+r)*rWidth +
    (mW2+c)];
    TmplImage[r*StdW + c].image *= Ratio[(mH2+r)*rWidth + (mW2+c)];
}
GlobalUnlock(hRatio);
GlobalFree(hRatio);
ColBuff = (COMPLEX*)LocalAlloc(GMEM_FIXED, H*sizeof(COMPLEX));
if(!InitFFT(StdH)) return FALSE;
W = (COMPLEX*)LocalLock(hSinCos);
BitRevPCT(StdH);

```

```

for(c = 0;c<StdW;c++)
{
    for(r=0;r<StdH;r++)
        ColBuff[StdH - r - 1] = TmplImage[r*StdW + c];
        FFT_DIF(ColBuff,StdH,TRUE);
    for(r=0;r<StdH;r++)
        TmplImage[r*StdW + c] = ColBuff[StdH - r - 1];
}
LocalFree(ColBuff);
LocalUnlock(hSinCos);
LocalFree(hSinCos);
LocalFree(BitLoc);
RowBuff = (COMPLEX*)LocalAlloc(GMEM_FIXED,
StdW*sizeof(COMPLEX));
if(!InitFFT(StdW)) return FALSE;
W = (COMPLEX*)LocalLock(hSinCos);
BitRevPCT(StdW);
Image = (UCHAR*)GlobalLock(hDImage);
for(r = 0;r<StdH;r++)
{
    for(c=0;c<StdW;c++)
        RowBuff[c] = TmplImage[r*StdW + c];
        FFT_DIF(RowBuff,StdW,TRUE);
    for(c=0;c<StdW;c++)
        TmplImage[r*StdW + c] = RowBuff[c];
}
LocalFree(RowBuff);
LocalUnlock(hSinCos);
LocalFree(hSinCos);
LocalFree(BitLoc);
double NIN2;
NIN2 = double(StdH * StdW);
for(i=0;i<size;i++)
{
    TmplImage[i].real = (double)sqrt((TmplImage[i].real * TmplImage[i].real
+TmplImage[i].image*TmplImage[i].image))/NIN2;
    if(TmplImage[i].real<0) TmplImage[i].real =0;
    if(TmplImage[i].real>255) TmplImage[i].real =255;
}
Image = (UCHAR far*)GlobalLock(hDImage);
for(r=0;r<mHeight;r++)
    for(c=0;c<mWidth;c++)
        Image[r*rWidth+c] = (UCHAR)(TmplImage[r*StdW + c].real);
GlobalUnlock(hTmplImage);
GlobalFree(hTmplImage);
GlobalUnlock(hDImage);
return TRUE;
}

```

```
BOOL FFT_DIF(COMPLEX * Array,int Len,BOOL Inv)
{
    int Sign,NLevel,BlkLen;
    register i,j,k;
    Sign = 1;
    if(Inv) Sign = -1;
    NLevel = 0;
    i = 1 ;
    while(i<Len)
    {
        i = 1<<NLevel;
        NLevel++;
    }
    if(i != Len)
    {
        MessageBox(NULL,"Error in FFT process!","System", MB_OK);
        return FALSE;
    }
    int HalfBlk,BlkCount,p,q,start,coef;
    COMPLEX Tmp,Wk;
    coef = 1;
    BlkLen = Len;
    for(i=0;i<NLevel;i++)
    {
        BlkCount = 1<<i;
        BlkLen = Len>>i;
        HalfBlk = BlkLen>> 1;
        start = 0;
        p = 0;
        for(j=0;j<BlkCount;j++)
        {
            for(k=0;k<HalfBlk;k++)
            {
                p = start + k;
                q = p + HalfBlk;
                Tmp.real = Array[p].real - Array[q].real;
                Tmp.image = Array[p].image - Array[q].image;
                Array[p].real = Array[p].real + Array[q].real;
                Array[p].image = Array[p].image + Array[q].image;
                Wk.real = W[k*coef].real;
                Wk.image = Sign * W[k*coef].image;
                Array[q].real = Tmp.real*Wk.real + Tmp.image*Wk.image;
                Array[q].image = -Tmp.real*Wk.image + Tmp.image*Wk.real;
            }
            start += BlkLen;
        }
    }
}
```

```

        coef <<= 1;
    }
    // Bits reversal
    for(i=0;i<(Len-1);i++)
    {
        j=BitLoc[i];
        if(j<=i)
        {
            Tmp.real = Array[i].real;
            Tmp.image = Array[i].image;
            Array[i].real = Array[j].real;
            Array[i].image = Array[j].image;
            Array[j].real = Tmp.real;
            Array[j].image = Tmp.image;
        }
    }
    return TRUE;
}
BOOL InitFFT(int BuffLen)
{
    double pi = 3.1415926;
    int Len;
    Len = BuffLen>>1;
    hSinCos = (COMPLEX*)LocalAlloc(GMEM_MOVEABLE,
    Len*sizeof(COMPLEX));
    if(hSinCos==NULL)
    {
        MsgBox(NULL,114,MB_OK);
        return FALSE;
    }
    W = (COMPLEX*)GlobalLock(hSinCos);
    W[0].real = 1.00;
    W[0].image = 0.00;
    W[1].real = cos(double(pi/Len));
    W[1].image= sin(double(pi/Len));
    for(int i=2;i<Len;i++)
    {
        W[i].real= W[i-1].real*W[1].real - W[i-1].image*W[1].image;
        W[i].image = W[i-1].image*W[1].real+ W[i-1].real*W[1].image;
    }
    LocalUnlock(hSinCos);
    return TRUE;
}
BOOL BitRevPCT(INT Len)
{
    int i,j,k;
    BitLoc = (WORD*)LocalAlloc(GMEM_FIXED,Len*sizeof(WORD));

```

```

if(BitLoc==NULL)
{
    MsgBox(NULL,114,MB_OK);
    exit(1);
}
for(i=0;i<Len;i++)
    BitLoc[i] = (WORD)i;
j = 0;
for(i=0;i<(Len-1);i++)
{
    if(i<=j)
    {
        BitLoc[i] = j;
        BitLoc[j] = i;
        k = Len>>1;
        while(k<=j)
        {
            j -= k;
            k >>= 1;
        }
    }
    else
    {
        k = Len>>1;
        while(k<=j)
        {
            j -= k;
            k >>= 1;
        }
        j += k;
    }
}
return TRUE;
}

```

*Biến đổi ảnh: Biến đổi Cosine xuôi - ngược*

\* Modul FCT.cpp

```

#include "StdAfx.h"
#include "FCT.h"
#include "Util.h"
#include <math.h>
extern WORD mWidth,mHeight,rWidth;
extern HANDLE hDImage;
HANDLE hCij;
double *Cij;
WORD *BitIdx;
INT NLevel;

```

```

BOOL BitRev(INT Len);
BOOL InitFCT(HANDLE *hCij,int Len);
BOOL InitIPCT(HANDLE *hCij,int Len);
BOOL FCT(double *Array,int Len);
BOOL IPCT(double *Array,int Len);
BOOL FCT2D(HANDLE hDImage,WORD mWidth,WORD mHeighth);
BOOL IPCT2D(HANDLE hDImage,WORD mWidth,WORD mHeighth);
BOOL FCT2D()
{
    HANDLE hTmpImage;
    UCHAR far* Image;
    double *RowBuff,*ColBuff,far *TemplImage;
    int size,StdW,StdH,StdW2,StdH2;
    register int r,c,i;
    CHAR FcName[256];
    StdW = GetStdLen(mWidth);
    StdH = GetStdLen(mHeighth);
    StdW2 = StdW>>1;
    StdH2 = StdH>>1;
    size = StdW * StdH;
    hTmpImage = (double far*)GlobalAlloc
    (GMEM_MOVEABLE,size*sizeof(double));
    if(hTmpImage==NULL)
    {
        MsgBox(NULL,114,MB_OK);
        return FALSE;
    }
    TemplImage = (double far*)GlobalLock(hTmpImage);
    Image = (UCHAR far*)GlobalLock(hDImage);
    RowBuff = (double*)LocalAlloc(GMEM_FIXED, StdW*sizeof(double));
    if(!InitFCT(&hCij,StdW)) return FALSE;
    Cij = (double*)LocalLock(hCij);
    BitRev(StdW);
    for(r = 0;r<mHeighth;r++)
    {
        for(c=0;c<mWidth;c++)
            RowBuff[c] = (double)Image[r*mWidth + c];
        for(c=mWidth;c<StdW;c++)
            RowBuff[c] = 0.0;
        FCT(RowBuff,StdW);
        for(c=0;c<StdW;c++)
            TemplImage[r*StdW + c] = RowBuff[c];
    }
    LocalFree(RowBuff);
    GlobalUnlock(hDImage);
    LocalUnlock(hCij);
    LocalFree(hCij);
    LocalFree(BitIdx);
}

```

```

ColBuff = (double*)LocalAlloc(GMEM_FIXED, StdH*sizeof(double));
if(!InitFCT(&hCij,StdH)) return FALSE;
Cij = (double*)LocalLock(hCij);
BitRev(StdH);
for(c = 0;c<StdW;c++)
{
    for(r=0;r<StdH;r++)
        ColBuff[StdH - r - 1] = TempImage[r*StdW + c];
    FCT(ColBuff,StdH);
    for(r=0;r<StdH;r++)
        TempImage[r*StdW + c] = ColBuff[StdH - r - 1];
}
LocalFree(ColBuff);
LocalUnlock(hCij);
LocalFree(hCij);
LocalFree(BitIdx);
FILE *Data;
GetTempPath(256,FcName);
strcat(FcName,"-fcdat.ipt");
Data = fopen(FcName,"wb");
if(Data == NULL) return FALSE;
fwrite(TempImage,sizeof(double),size,Data);
fclose(Data);
double max = 0.0;
for(i=0;i<size;i++)
{
    if(TempImage[i]>max) max = TempImage[i];
    if(TempImage[i]<0.0) TempImage[i]=0.0;
}
Image = (UCHAR far*)GlobalLock(hDImage);
double coef;
coef = 255.0/log((double)(max + 1.0));
for(r=0;r<mHeight;r++)
    for(c=0;c<mWidth;c++)
    {
        Image[r*mWidth + c] = 255-(UCHAR)(0.5+
            log((double)TempImage[r*StdW + c]+1.0)*coef);
    }
GlobalUnlock(hTmpImage);
GlobalFree(hTmpImage);
GlobalUnlock(hDImage);
return TRUE;
}

BOOL IPCT2D()
{
    HANDLE hTmpImage;
    UCHAR far* Image;

```

```

double *RowBuff,*ColBuff,for *TemplImage;
int size,StdW,StdH;
register int r,c;
FILE *Data;
CHAR FcName[256];
StdW = GetStdLen(mWidth);
StdH = GetStdLen(mHeight);
size = StdW * StdH;
hTmplImage = (double far*)GlobalAlloc(GMEM_MOVEABLE,
size*sizeof(double));
if(hTmplImage==NULL)
{
    MsgBox(NULL,114,MB_OK);
    return FALSE;
}
TmplImage = (double far*)GlobalLock(hTmplImage);
GetTempPath(256,FcName);
streat(FcName,"~fc1dat.ipt");
Data = fopen(FcName,"rb");
if(Data == NULL)
{
    MsgBox(NULL,117,MB_OK);
    GlobalUnlock(hTmplImage);
    GlobalFree(hTmplImage);
    return FALSE;
}
fread(TmplImage,sizeof(double),size,Data);
fclose(Data);
remove(FcName);
ColBuff = (double*)LocalAlloc(GMEM_FIXED, StdH*sizeof(double));
BitRev(StdH);
InitIPCT(&hCij,StdH);
Cij = (double*)LocalLock(hCij);
for(c = 0;c<StdW;c++)
{
    for(r=0;r<StdH;r++)
        ColBuff[StdH - r - 1] = TmplImage[r*StdW + c];
    IPCT(ColBuff,StdH);
    for(r=0;r<StdH;r++)
        TmplImage[r*StdW + c] = ColBuff[StdH - r - 1];
}
LocalFree(ColBuff);
LocalUnlock(hCij);
LocalFree(hCij);
LocalFree(BitIdx);
RowBuff = (double*)LocalAlloc(GMEM_FIXED, StdW*sizeof(double));
BitRev(StdW);
InitIPCT(&hCij,StdW);

```

```

Cij = (double*)LocalLock(hCij);
Image = (UCHAR *)GlobalLock(hDImage);
for(r=0;r<StdH;r++)
{
    for(c=0;c<StdW;c++)
        RowBuff[c] = TemplImage[r*StdW + c];
    IPCT(RowBuff,StdW);
    for(c=0;c<StdW;c++)
        TemplImage[r*StdW + c] = RowBuff[c];
}
LocalFree(RowBuff);
LocalUnlock(hCij);
LocalFree(hCij);
LocalFree(BitIdx);
Image = (UCHAR far*)GlobalLock(hDImage);
for(r=0;r<mHeight;r++)
    for(c=0;c<mWidth;c++)
        Image[r*rWidth+c] = (UCHAR)(TemplImage[r*StdW + c]);
GlobalUnlock(hTmplImage);
GlobalFree(hTmplImage);
GlobalUnlock(hDImage);
return TRUE;
}
BOOL PCT(double *Array,int Len)
{
    int i,j,k,kk,Len1,Len2;
    double *TmpArr;
    //sắp xếp lại
    TmpArr = (double*)GlobalAlloc(GMEM_FIXED, Len*sizeof(double));
    j = 0;
    for(i=0;i<Len;i += 2)
        TmpArr[j++] = Array[i];
    for(i=1;i<Len;i += 2 )
        TmpArr[j++] = Array[Len - i];
    for(i=0;i<Len;i++)
        Array[i] = TmpArr[i];
    GlobalFree(TmpArr);
    // Main Process
    int p,q,Dist,Next;
    double tmp;
    Len1 = Len>>1;
    Dist = Len1;
    Next = Len;
    kk = 0;
    for(i=0;i<NLevel;i++)
    {
        for(k=0;k<Dist;k++)
        {

```

```

        for(p=k;p<Len;p += Next)
        {
            q = p + Dist;
            tmp = Array[p] - Array[q];
            Array[p] += Array[q];
            Array[q] = tmp * Cij[kk];
        }
        kk++;
    }
    Dist >>= 1;
    Next >>= 1;
}
// Bits reversal
for(i=0;i<(Len-1);i++)
{
    j = BitIdx[i];
    if(i<=j)
    {
        tmp = Array[i];
        Array[i] = Array[j];
        Array[j] = tmp;
    }
}
// subtraction
for(i=1;i<Len1;i++) Array[i] *= 0.5;
Len2 = Len>>2;
k = 1;
for(i = 1;i < NLevel;i++)
{
    k <<= 1;
    j = k+1;
    for(int h = 1;h<=j;h++)
        for(int l=0; l<Len2;l++)
            Array[Len2 + h * Len1 + l] -=
                Array[Len2 + (h-1) * Len1 + l];
    Len1>>=1;
    Len2>>=1;
}
Array[0] *= 1.4142135624/double(Len);
for(i=1; i<Len;i++)
    Array[i] *= 2.0/double(Len);
return TRUE;
}
BOOL InitPCT(HANDLE *hCij,int Len)
{
    int i,j;
    double pi = 3.1415926535;
    i = Len - 1 ;
}

```

```

j = 0;
while(i)
{
    i >>= 1;
    j++;
}
NLevel = j;
*hCij = (double*)LocalAlloc(GMEM_MOVEABLE, Len*sizeof(double));
if(*hCij==NULL)
{
    MsgBox(NULL,114,MB_OK);
    return FALSE;
}
Cij = (double*)LocalLock(*hCij);
int Len1,Len2,k;
Len1 = Len>>1;
Len2 = Len<<1;
k=0;
for(i=0;i<NLevel;i++)
{
    for(j=0;j<Len1;j++)
    {
        Cij[k] = (double)cos(pi*double(4*j+1)/ double(Len2));
        k++;
    }
    Len1>>=1;
    Len2>>=1;
}
for(i=0;i<k;i++)
{
    Cij[i] *= 2.0;
}
LocalUnlock(*hCij);
return TRUE;
}
/*=====Inverse FCT=====*/
BOOL iPCT(double *Array,int Len)
{
    int i,j,k,Len1,Len2;
    double *TmpArr;
    Array[0] *= double(Len)/1.4142135624;
    for(i=1;i<Len;i++)
        Array[i] *= double(Len)/2.0;
    // addition
    Len1 = 1;
    Len2 = 2;
    k = 1<<NLevel;
    for(int h = 1;h<NLevel;h++)

```

```

{
    k >= 1;
    j = k-1;
    for(int l = j;l>=1;l--)
        for(i=0;i<Len1;i++)
            Array[Len1 + l*Len2+i] +=
                Array[Len1 + (l-1)*Len2 + i];
    Len1<<=1;
    Len2<<=1;
}
Len1 = Len>>1;
for(i=1;i<Len1;i++)
    Array[i] *= 2.0;
// Bits reversal
double tmp;
for(i=1;i<(Len-1);i++)
{
    j=BitIdx[i];
    if(i>=j)
    {
        tmp = Array[i];
        Array[i] = Array[j];
        Array[j] = tmp;
    }
}

// Main Process
int p,q,kk,Dist,Next;
Dist = 1;
Next = 2;
kk = 0;
for(i=0;i<NLevel;i++)
{
    for(k=0;k<Dist;k++)
    {
        for(p=k;p<Len;p += Next)
        {
            q = p + Dist;
            tmp = Array[p]*0.5;
            Array[q] *= Cij[kk];
            Array[p] = tmp + Array[q];
            Array[q] = tmp - Array[q];
        }
        kk++;
    }
    Dist <<= 1;
    Next <<= 1;
}
}

```

```

//Rearrange
TmpArr = (double*)GlobalAlloc(GMEM_FIXED, Len*sizeof(double));
j = 0;
Len1 = Len>>1;
for(i=0;i<Len1;i++)
{
    TmpArr[j] = Array[i];
    j += 2;
}
j = 1;
for(i=Len1;i<Len;i++)
{
    TmpArr[Len - j] = Array[i];
    j += 2;
}
for(i=0;i<Len;i++)
    Array[i] = TmpArr[i];
GlobalFree(TmpArr);
return TRUE;
}

BOOL InitIPCT(HANDLE *hCij,int N)
{
    double pi = 3.1415926535;
    int i,j,k,Len1,Len2;
    *hCij = (double*)LocalAlloc(GMEM_MOVEABLE, N*sizeof(double));
    if(*hCij==NULL)
    {
        MsgBox(NULL,114,MB_OK);
        return FALSE;
    }
    Cij = (double*)LocalLock(*hCij);
    i = N - 1 ;
    j = 0;
    while(i)
    {
        i >>= 1;
        j++;
    }
    NLevel = j;
    Len1 = 4;
    Len2 = 1;
    k = 0;
    for(i=1;i<=NLevel;i++)
    {
        for(j=0;j<Len2;j++)
        {
            Cij[k] = 1.0/(double)cos(pi*(double(4*j+1) /double(Len1)));
            k++;
        }
    }
}

```

```

    }
    Len1<<=1;
    Len2<<=1;
}
for(i=0;i<N;i++)
    Cij[i] /= 4.0;
LocalUnlock(*hCij);
return TRUE;
}

/*=====
Biến đổi FCT for JPEG
HANDLE FCTB(WORD *Width, WORD *Heighth)
{
    HANDLE hTmpImage;
    UCHAR far *Image;
    double far *TemplImage;
    double Buff[8];
    int size,bc,br;
    register int r,c;
    int NV,NH,Rest;
    *Heighth = ((mHeighth - 1)/8 + 1)*8;
    *Width = ((mWidth - 1)/8 + 1)*8;
    NV = *Heighth/8;
    NH = *Width/8;
    Rest = 8 + mWidth - (*Width);
    size = (*Heighth) * (*Width);
    hTmpImage = (double far*)GlobalAlloc (GMEM_MOVEABLE,
    size*sizeof(double));
    if(hTmpImage==NULL)
    {
        MsgBox(NULL,114,MB_OK);
        return NULL;
    }
    TemplImage = (double far*)GlobalLock(hTmpImage);
    BitRev(8);
    InitFCT(&hCij,8);
    Cij = (double*)LocalLock(hCij);
    Image = (UCHAR far*)GlobalLock(hDImage);
    for(r = 0;r<mHeighth;r++)
    {
        for(bc=0;bc<(NH-1);bc++)
        {
            for(c=0;c<8;c++)
                Buff[c] = (double)Image[r*rWidth + bc*8 + c];
            FCT(Buff,8);
            for(c=0;c<8;c++)
                TemplImage[r*(*Width)+bc*8+c] = Buff[c];
        }
    }
}
```

```

    }
    for(c=0;c<Rest;c++)
        Buff[c] = (double)Image[r*rWidth + (NH-1)*8 + c];
    for(c=Rest;c<8;c++)      Buff[c] = 0.0;
    PCT(Buff,8);
    for(c=0;c<8;c++)
        TempImage[r*(*Width) + (NH-1)*8 + c] = Buff[c];
}
GlobalUnlock(hDImage);
for(c = 0;c<(*Width);c++)
{
    for(br=0;br<NV;br++)
    {
        for(r=0;r<8;r++)
            Buff[r] = TempImage[(br*8 + r)*(*Width) + c];
        PCT(Buff,8);
        for(r=0;r<8;r++)
            TempImage[(br*8 + r)*(*Width) + c] = Buff[r];
    }
}
LocalUnlock(hCij);
LocalFree(hCij);
LocalFree(BitIdx);
GlobalUnlock(hTempImage);
return hTempImage;
}

```

BOOL IPCTB(HANDLE hTempImage,WORD Width,WORD Height)

```

{
    UCHAR far *Image;
    double far *TempImage;
    double Buff[8];
    int bc,br;
    register int r,c;
    int NV,NH;
    NV = Height/8;
    NH = Width/8;
    TempImage = (double far*)GlobalLock(hTempImage);
    InitIPCT(&hCij,8);
    Cij = (double*)LocalLock(hCij);
    BitRev(8);
    for(c = 0;c<Width;c++)
    {
        for(br=0;br<NV;br++)
        {
            for(r=0;r<8;r++)
                Buff[r] = (double)TempImage[(br*8 + r)*Width + c];
            IPCT(Buff,8);
        }
    }
}
```

```

        for(r=0;r<8;r++)
            TemplImage[(br*8+r)*Width + c] = Buff[r];
    }

    for(r = 0;r<Height;r++)
    {
        for(bc=0;bc<NH;bc++)
        {
            for(c=0;c<8;c++)
                Buff[c] = (double)TemplImage[r*Width + bc*8 + c];
            IFCT(Buff,8);
            for(c=0;c<8;c++)
                TemplImage[r*Width + bc*8 + c] = Buff[c];
        }
    }

    Image = (UCHAR far*)GlobalLock(hDImage);
    for(r = 0;r<mHeight;r++)
    {
        for(c=0;c<mWidth;c++)
            Image[r*rWidth+c] = (UCHAR)TemplImage[r*Width+c];
    }
    LocalUnlock(hCij);
    LocalFree(hCij);
    LocalFree(BitIdx);
    GlobalUnlock(hTmplImage);
    GlobalFree(hTmplImage);
    GlobalUnlock(hDImage);
    return TRUE;
}

BOOL BitRev(INT Len)
{
    int i,j,k;
    BitIdx = (WORD*)LocalAlloc(GMEM_FIXED, Len*sizeof(WORD));
    for(i=0;i<Len;i++)
        BitIdx[i] = (WORD)i;
    j = 0;
    for(i=0;i<(Len-1);i++)
    {
        if(i<=j)
        {
            BitIdx[i] = j;
            BitIdx[j] = i;
            k = Len>>1;
            while(k<=j)
            {
                j := k;
                k >>= 1;
            }
        }
    }
}

```

```

    }
else
{
    k = Len>>1;
    while(k<=j)
    {
        j -= k;
        k >>= 1;
    }
    j += k;
}

return TRUE;
}

```

**Nén ảnh RLE, HUFFMAN, LZW, JPEG**

```

#include "StdAfx.h"
#include "Compress.h"
#include "Globals.h"
#include "RLE.h"
#include "Huffman.h"
#include "LZW.h"
#include "JPEG.h"
#include "Util.h"
#include "Resource.h"
#include <string.h>
struct ZIPINFO
{
    UINT ImageSize;
    CHAR ZPname[100];
    UINT ZSize;
    CHAR Ratio[20];
    CHAR Methode[50];
};

ZIPINFO ZInfo;
LRESULT CALLBACK ZipShow(HWND, UINT, WPARAM, LPARAM);
BOOL Compress(HWND hWnd, INT Methode)
{
    /*===== Các phương pháp ======
    1: RLC
    2: Huffman
    3: LZW
    4: JPEG
    =====*/
    HANDLE hZip, hZPal;
    UCHAR far *Zip;

```

```

UINT OSize;
INT ColorsNum;
CHAR ZipFile[256],Type[5],MetName[50];
FILE *Data;
LPPALETTEENTRY ZPal;
ColorsNum = 1<<mBitCount;
switch(Methode)
{
    case 1: /* RLE */
        strcpy(Type,"SRLE");
        strcpy(MetName,"Phương pháp RLE");
        RLE(hDImage,mSize,&hZip,&OSize);
        if(!SaveAsDwg(hWnd,"RLE (*.rle)\\0*.rlc",
                        (LPSTR)ZipFile,"rlc"))
            return FALSE;
        Zip = (UCHAR far*)GlobalLock(hZip);
        Data = fopen(ZipFile,"wb");
        if(Data==NULL) return FALSE;
        fwrite(&Type,sizeof(CHAR),1,Data);
        fwrite(&mWidth,sizeof(WORD),1,Data);
        fwrite(&mHeigh,sizeof(WORD),1,Data);
        fwrite(&mBitCount,sizeof(BYTE),1,Data);
        hZPal = (PALETTEENTRY*)LocalAlloc
                (GMEM_MOVEABLE,ColorsNum*sizeof(PALETTEENTRY));
        ZPal = (PALETTEENTRY*) LocalLock(hZPal);
        GetPaletteEntries(hCPal,0,ColorsNum,ZPal);
        fwrite(ZPal,sizeof(PALETTEENTRY), ColorsNum,Data);
        LocalUnlock(hZPal);
        LocalFree(hZPal);
        fwrite(&OSize,sizeof(UINT),1,Data);
        fwrite(Zip,sizeof(UCHAR),OSize,Data);
        fclose(Data);
        break;
    case 2: /* HUFFMAN */
        strcpy(Type,"SHUF");
        strcpy(MetName,"Phương pháp Huffman");
        if(!SaveAsDwg(NULL,"Huffman (*.huf)\\0*.huf",
                        (LPSTR)ZipFile,"huf")) return FALSE;
        Data = fopen(ZipFile,"wb");
        if(Data==NULL) return FALSE;
        fwrite(&Type,sizeof(CHAR),1,Data);
        fwrite(&mWidth,sizeof(WORD),1,Data);
        fwrite(&mHeigh,sizeof(WORD),1,Data);
        fwrite(&mBitCount,sizeof(BYTE),1,Data);
        hZPal = (PALETTEENTRY*)LocalAlloc
                (GMEM_MOVEABLE,
                 ColorsNum*sizeof(PALETTEENTRY));
        ZPal = (PALETTEENTRY*) LocalLock(hZPal);

```

```

GetPaletteEntries(hCPal,0,ColorsNum,ZPal);
fwrite(ZPal,sizeof(PALETTEENTRY),
ColorsNum,Data);
LocalUnlock(hZPal);
LocalFree(hZPal);
HUFF(hDImage,mSize,&hZip,&OSize,Data);
fclose(Data);
break;
case 3: /* LZW */
strcpy(Type,"SLZW");
strcpy(MetName,"Phương pháp LZW");
LZW(hDImage,mSize,&hZip,&OSize,256,4096);
if(!SaveAsDlg(hWind,"LZW (*.lzw)\\0*.lzw",
(LPSTR)ZipFile,"lzw")) return FALSE;
Zip = (UCHAR far*)GlobalLock(hZip);
Data = fopen(ZipFile,"wb");
if(Data==NULL) return FALSE;
fwrite(&Type,sizeof(CHAR),4,Data);
fwrite(&mWidth,sizeof(WORD),1,Data);
fwrite(&mHeigh,sizeof(WORD),1,Data);
fwrite(&mBitCount,sizeof(BYTE),1,Data);
hZPal = (PALETTEENTRY*)LocalAlloc
(GMEM_MOVEABLE,
ColorsNum*sizeof(PALETTEENTRY));
ZPal = (PALETTEENTRY*) LocalLock(hZPal);
GetPaletteEntries(hCPal,0,ColorsNum,ZPal);
fwrite(ZPal,sizeof(PALETTEENTRY),
ColorsNum,Data);
LocalUnlock(hZPal);
LocalFree(hZPal);
fwrite(&OSize,sizeof(UINT),1,Data);
fwrite(Zip,sizeof(UCHAR),OSize,Data);
fclose(Data);
break;
case 4: /* JPEG */
strcpy(Type,"SJPG");
strcpy(MetName,"Phương pháp JPEG");
if(!SaveAsDlg(NULL,"Pseudo JPEG (*.mjp)\\0*.mjp",
(LPSTR)ZipFile,"mjp")) return FALSE;
SetCursor(LoadCursor(NULL, IDC_WAIT));
Data = fopen(ZipFile,"wb");
if(Data==NULL) return FALSE;
fwrite(&Type,sizeof(CHAR),4,Data);
fwrite(&mWidth,sizeof(WORD),1,Data);
fwrite(&mHeigh,sizeof(WORD),1,Data);
fwrite(&mBitCount,sizeof(BYTE),1,Data);

```

```

hZPal = (PALETTEENTRY*) LocalAlloc
(GMEM_MOVEABLE,
ColorsNum*sizeof(PALETTEENTRY));
ZPal = (PALETTEENTRY*) LocalLock(hZPal);
GetPaletteEntries(hCPal,0,ColorsNum,ZPal);
fwrite(ZPal,sizeof(PALETTEENTRY),
ColorsNum,Data);
LocalUnlock(hZPal);
LocalFree(hZPal);
JPEG(Data,&OSize);
fclose(Data);
break;
default:
    return TRUE;
}

GlobalUnlock(hZip);
GlobalFree(hZip);
CHAR RStr[20];
RStr[0]=^0';
double RTemp;
strcpy(ZInfo.ZFname,ZipFile);
strcpy(ZInfo.Methode,MetName);
RTemp = double((double)mSize/(double)OSize);
gvt(RTemp,5,RStr);
strcpy(ZInfo.Ratio,RStr);
ZInfo.ImageSize = mSize;
ZInfo.ZSize = OSize;
DialogBox(NULL, (LPCTSTR)IDD_COMPRESS, hWnd, (DLGPROC)ZipShow);
return TRUE;
LRESULT CALLBACK ZipShow(HWND hDlg,
UINT message, WPARAM wParam,LPARAM lParam)
{
switch (message)
{
case WM_INITDIALOG:
    SetDlgItemInt(hDlg, IDC_IMGSIZE,ZInfo.ImageSize, FALSE);
    SetDlgItemInt(hDlg, IDC_ZSIZE,ZInfo.ZSize, FALSE);
    SetDlgItemText(hDlg, IDC_ZFILE,(LPSTR)ZInfo.ZFname);
    SetDlgItemText(hDlg, IDC_ZMODE,(LPSTR)ZInfo.Methode);
    SetDlgItemText(hDlg, IDC_ZRATIO,(LPSTR)ZInfo.Ratio);
    return TRUE;
case WM_COMMAND:
    if (LOWORD(wParam) == IDOK )
    {
        EndDialog(hDlg, LOWORD(wParam));
        return TRUE;
    }
}

```

```

        break;
    }
    return FALSE;
}

```

**Giải nén**

```

#include "StdAfx.h"
#include "DeComp.h"
#include "Globals.h"
#include "DeRLE.h"
#include "DeHuff.h"
#include "DeLZW.h"
#include "Jpeg.h"
#include "Util.h"
#include <string.h>
BOOL DeCompress(HWND hWnd, INT Methode)
{
    /*===== Các phương pháp ======
    1: RLC
    2: Huffman
    3: LZW
    4: JPEG
    =====*/
    HANDLE hZImage,hPalEntry,hCode,hClen,hGray;
    PALETTEENTRY *PalEntry;
    UCHAR far *ZImage,*CodeLen,*Gray,Clen;
    UINT ZSize,ColorsNum,*Code,far *HuffImage;
    CHAR ZipFile[256],Type[5];
    FILE *Data;
    switch(Methode)
    {
        case 1: /* RLE */
            if(!OpenDig(hWnd,"RLE (*.rlc)\\0*.rlc",ZipFile))
                return FALSE;
            Data = fopen(ZipFile,"rb");
            if(Data==NULL) return FALSE;
            fread(&Type,sizeof(CHAR),4,Data);
            if((Type[0]!='S')||(Type[1]!='R')||(Type[2]!='L')
            ||(Type[3]!='E'))
            {
                MessageBox(hWnd,"File không hợp lệ!","Xử lý ảnh", MB_OK);
                break;
            }
            fread(&mWidth,sizeof(WORD),1,Data);
            fread(&mHeight,sizeof(WORD),1,Data);
            fread(&mBitCount,sizeof(BYTE),1,Data);
            ColorsNum = 1<<mBitCount;

```

```

hPalEntry = (PALETTEENTRY*)
LocalAlloc(GMEM_MOVEABLE,
ColorsNum*sizeof(PALETTEENTRY));
PalEntry = (PALETTEENTRY*) LocalLock(hPalEntry);
fread(PalEntry,sizeof(PALETTEENTRY),
ColorsNum,Data);
hPal = CrPal(ColorsNum,PalEntry);
hCPal = hPal;
LocalUnlock(hPalEntry);
LocalFree(hPalEntry);
fread(&ZSize,sizeof(UINT),1,Data);
hZImage = (UCHAR far*)GlobalAlloc(GMEM_MOVEABLE,
ZSize*sizeof(UCHAR));
if(hZImage==NULL) return FALSE;
ZImage = (UCHAR far*)GlobalLock(hZImage);
fread(ZImage,sizeof(UCHAR),ZSize,Data);
fclose(Data);
GlobalUnlock(hZImage);
rWidth = mWidth;
if(mWidth%4)
    rWidth = mWidth+4 - mWidth%4;
mSize = rWidth * mHeighth;
if(hDImage!=NULL) GlobalFree(hDImage);
hDImage = (UCHAR* far)GlobalAlloc(GMEM_MOVEABLE,
mSize*sizeof(UCHAR));
if(hDImage==NULL) return FALSE;
DeRLE(hZImage,ZSize,hDImage);
GlobalFree(hZImage);
break;
case 2: /* HUFFMAN */
if(!OpenDlgItem(hWnd,"Huffman(*.huf)NO*.huf", ZipFile))
    return FALSE;
Data = fopen(ZipFile,"rb");
if(Data==NULL) return FALSE;
fread(&Type,sizeof(CHAR),4,Data);
if((Type[0]!='S')||(Type[1]!='H')||(Type[2]!='U')
||(Type[3]!='F'))
{
    MessageBox(hWnd,"File không hợp lệ!","Xử lý ảnh",MB_OK);
    break;
}
fread(&mWidth,sizeof(WORD),1,Data);
fread(&mHeighth,sizeof(WORD),1,Data);
fread(&mBitCount,sizeof(BYTE),1,Data);
ColorsNum = 1<<mBitCount;
hPalEntry = (PALETTEENTRY*) LocalAlloc
(GMEM_MOVEABLE,
ColorsNum*sizeof(PALETTEENTRY));

```

```
PalEntry = (PALETTEENTRY*) LocalLock(hPalEntry);
fread(PalEntry,sizeof(PALETTEENTRY),
ColorsNum,Data);
hPal = CrPal(ColorsNum,PalEntry);
hCPal = hPal;
LocalUnlock(hPalEntry);
LocalFree(hPalEntry);
fread(&CLen,sizeof(UCHAR),I,Data);
hCode = (UINT*)LocalAlloc(GMEM_MOVEABLE,
CLen*sizeof(UINT));
if(hCode==NULL) return FALSE;
Code = (UINT*)LocalLock(hCode);
fread(Code,sizeof(UINT),CLen,Data);
LocalUnlock(hCode);
hCLen = (UCHAR*)LocalAlloc(GMEM_MOVEABLE,
CLen*sizeof(UCHAR));
if(hCLen==NULL) return FALSE;
CodeLen = (UCHAR*)LocalLock(hCLen);
fread(CodeLen,sizeof(UCHAR),CLen,Data);
LocalUnlock(hCLen);
hGray = (UCHAR*)LocalAlloc(GMEM_MOVEABLE,
CLen*sizeof(UCHAR));
if(hGray==NULL) return FALSE;
Gray = (UCHAR*)LocalLock(hGray);
fread(Gray,sizeof(UCHAR),CLen,Data);
LocalUnlock(hGray);
fread(&ZSize,sizeof(UINT),I,Data);
hZImage = (UINT far*)GlobalAlloc(GMEM_MOVEABLE,
ZSize*sizeof(UINT));
if(hZImage==NULL) return FALSE;
HuffImage = (UINT far*)GlobalLock(hZImage);
fread(HuffImage,sizeof(UINT),ZSize,Data);
fclose(Data);
GlobalUnlock(hZImage);
rWidth = mWidth;
if(mWidth%4)
    rWidth = mWidth+4 - mWidth%4;
mSize = rWidth * mHeight;
if(hDImage!=NULL) GlobalFree(hDImage);
hDImage = (UCHAR* far)GlobalAlloc(GMEM_MOVEABLE,
mSize*sizeof(UCHAR));
if(hDImage==NULL) return FALSE;
DeHUFF(hZImage,ZSize,hCode,CLen,hClen,hGray, hDImage);
GlobalFree(hZImage);
break;
case 3: /* LZW */
    if(!OpenDlg(hWnd,"LZW (*.lzw)\0*.lzw", ZipFile))
        return FALSE;
```

```

Data = fopen(ZipFile,"rb");
if(Data==NULL) return FALSE;
fread(&Type,sizeof(CHAR),4,Data);
if((Type[0]!='S')||(Type[1]!='L')||(Type[2]!='Z')
||(Type[3]!='W'))
{
    MessageBox(hWnd,"File không hợp lệ!","Xử lý ảnh", MB_OK);
    return FALSE;
}
fread(&mWidth,sizeof(WORD),1,Data);
fread(&mHeight,sizeof(WORD),1,Data);
fread(&mBitCount,sizeof(BYTE),1,Data);
ColorsNum = 1<<mBitCount;
hPalEntry = (PALETTEENTRY*)
LocalAlloc(GMEM_MOVEABLE,
ColorsNum*sizeof(PALETTEENTRY));
PalEntry = (PALETTEENTRY*) LocalLock(hPalEntry);
fread(PalEntry,sizeof(PALETTEENTRY),
ColorsNum,Data);
hPal = CrPal(ColorsNum,PalEntry);
hCPal = hPal;
LocalUnlock(hPalEntry);
LocalFree(hPalEntry);
fread(&ZSize,sizeof(UINT),1,Data);
hZImage = (UCHAR far*)GlobalAlloc (GMEM_MOVEABLE,
ZSize*sizeof(UCHAR));
if(hZImage==NULL) return FALSE;
ZImage = (UCHAR far*)GlobalLock(hZImage);
fread(ZImage,sizeof(UCHAR),ZSize,Data);
fclose(Data);
GlobalUnlock(hZImage);
rWidth = mWidth;
if(mWidth%4)
    rWidth = mWidth+4 - mWidth%4;
mSize = rWidth * mHeight;
if(hDImage!=NULL) GlobalFree(hDImage);
hDImage = (UCHAR far*)GlobalAlloc (GMEM_MOVEABLE,
mSize*sizeof(UCHAR));
if(hDImage==NULL) return FALSE;
SetDIBits(hZImage,hDImage,4096);
GlobalFree(hZImage);
break;
case 4/* JPEG */
if(!OpenDlg(hWnd,"Pseudo JPEG(*.mjp)\0*.mjp",ZipFile))
    return FALSE;
Data = fopen(ZipFile,"rb");
if(Data==NULL) return FALSE;
fread(&Type,sizeof(CHAR),4,Data);

```

```

        if((Type[0]!='S')||(Type[1]!='I')||(Type[2]!='P')||(Type[3]!='G'))
        {
            MessageBox(hWnd,"File không hợp lệ!","Xử lý ảnh", MB_OK);
            break;
        }
        fread(&mWidth,sizeof(WORD),1,Data);
        fread(&mHeight,sizeof(WORD),1,Data);
        fread(&mBitCount,sizeof(BYTE),1,Data);
        ColorsNum = 1<<mBitCount;
        hPalEntry = (PALETTEENTRY*)LocalAlloc
        (GMEM_MOVEABLE,
        ColorsNum*sizeof(PALETTEENTRY));
        PalEntry = (PALETTEENTRY*) LocalLock(hPalEntry);
        fread(PalEntry,sizeof(PALETTEENTRY),
        ColorsNum, Data);
        hPal = CrPal(ColorsNum,PalEntry);
        hCPal = hPal;
        LocalUnlock(hPalEntry);
        LocalFree(hPalEntry);
        rWidth = mWidth;
        if(mWidth%4)
            rWidth = mWidth+4 - mWidth%4;
        mSize = rWidth * mHeight;
        DeJPEG(Data);
        break;
    default:
        return TRUE;
    }
    GetTempPath(256,ZipFile);
    strcat(ZipFile,"~simage.ipt");
    ImageDisk(hDImage,mSize,ZipFile,TRUE);
    return TRUE;
}

```

**Nội dung tệp Image.CPP & Xử lý định dạng GIF**

```

#include "StdAfx.h"
#include "Image.h"
#include "Globals.h"
#include "DeLZW.h"
#include "Util.h"
#include <fcntl.h>
typedef struct tagCOLORENTY
{
    BYTE Red;
    BYTE Green;
    BYTE Blue;
} COLORENTY;
BMPOpen(HWND hWnd,FILE *bFile);

```

```

BOOL GIPOpen(HWND hWnd,FILE *gFile);
BOOL OpenFiles(HWND hWnd)
{
    CHAR FileName[256],*Ext;
    FILE *Data;
    if(!OpenDig(hWnd,"Bitmap (*.bmp)\0*.bmp\0 GIF (*.gif)\0*.gif",FileName))
        return NULL;
    Data = fopen(FileName,"rb");
    if(Data==NULL) return FALSE;
    Ext = strchr(FileName,'.');
    if(!strcmp(Ext,".bmp"))
    {
        if(!BMPOpen(hWnd,Data)) return FALSE;
    }
    else
        if(!strcmp(Ext,".gif"))
        {
            if(!GIPOpen(hWnd,Data)) return FALSE;
        }
    return TRUE;
}
/*=====GIF Routine=====*/
BOOL GIFHeader(HWND hWnd,FILE *gFile)
{
// Local Variables
    BYTE             Sign[6];
    WORD            ScrWidth;
    WORD            ScrHeight;
    BYTE            ScrFlags;
    BYTE            BkColor;
    BYTE            Reserved;
    BYTE            ImgFlags;
    INT             ColorsNum;
    HANDLE          hGMap,hLMap;
    COLORENTRY *GMap,*LMap;
    BOOL            LPAL,GPAL;
    UCHAR           next;
    HANDLE          hPalEntry;
    LPPALETTEENTRY PalEntry;
    HDC hDC;
    LPAL = GPAL = FALSE;
// Read Header
    fread(&Sign,sizeof(BYTE),6,gFile);
    mVersion = 100;
    if(Sign[0] != 'G' || Sign[1] != 'I' || Sign[2] != 'F')
        return FALSE;
    if(Sign[3] == '8' && Sign[4] == '7' && Sign[5] == 'a')

```

```
mVersion = 0;
if(Sign[3] == '8' && Sign[4] == '9' && Sign[5] == 'a')
    mVersion = 1;
if (mVersion > 2) return NULL;
mType = 1;
fread(&ScrWidth,sizeof(WORD),1,gFile);
fread(&ScrHeight,sizeof(WORD),1,gFile);
fread(&ScrFlags,sizeof(BYTE),1,gFile);
fread(&BkColor,sizeof(BYTE),1,gFile);
fread(&Reserved,sizeof(BYTE),1,gFile);
mBitCount = (ScrFlags & 0x0007) + 1;
ColorsNum = 1<<mBitCount;
if(ScrFlags & 0x0080 != 0)
{
    // Have Global Color Map
    GPAL = TRUE;
    hGMap
    =(COLORENTRY*)LocalAlloc(GMEM_MOVEABLE,ColorsNum *
    sizeof(COLORENTRY));
    GMap =(COLORENTRY*)LocalLock(hGMap);
    fread(GMap,sizeof(COLORENTRY),ColorsNum,gFile);
}
mInterlaced = 0;
if(ScrFlags & 0x0040 != 0) mInterlaced = 1;
next = '@';
while(next != ',')
{
    fread(&next,sizeof(UCHAR),1,gFile);
    fread(&mLeft,sizeof(WORD),1,gFile);
    fread(&mTop,sizeof(WORD),1,gFile);
    fread(&mWidth,sizeof(WORD),1,gFile);
    fread(&mHeight,sizeof(WORD),1,gFile);
    fread(&ImgFlags,sizeof(BYTE),1,gFile);
    if(ImgFlags & 0x0080)
    {
        //Use Local Color Map
        hLMap
        =(COLORENTRY*)LocalAlloc(GMEM_MOVEABLE,ColorsNum *
        sizeof(COLORENTRY));
        LMap =(COLORENTRY*)LocalLock(hLMap);
        fread(LMap,sizeof(COLORENTRY),ColorsNum,gFile);
        LPAL = TRUE;
    }
    hPalEntry = (PALETTEENTRY*)LocalAlloc (GMEM_MOVEABLE,
    ColorsNum*sizeof(PALETTEENTRY));
    PalEntry = (PALETTEENTRY*)LocalLock(hPalEntry);
    int i;
    if(!GPAL && !LPAL)
    {
```

```

hDC = GetDC(hWnd);
GetSystemPaletteEntries(hDC,0,ColorsNum,PalEntry);
ReleaseDC(hWnd,hDC);
}

if(LPAL)
    for(i=0;i<ColorsNum;i++)
    {
        PalEntry[i].peRed = LMap[i].Red;
        PalEntry[i].peGreen = LMap[i].Green;
        PalEntry[i].peBlue = LMap[i].Blue;
        PalEntry[i].peFlags = PC_RESERVED;
    }
else
    if(GPAL)
        for(i=0;i<ColorsNum;i++)
        {
            PalEntry[i].peRed = GMap[i].Red;
            PalEntry[i].peGreen = GMap[i].Green;
            PalEntry[i].peBlue = GMap[i].Blue;
            PalEntry[i].peFlags = PC_RESERVED;
        }

hPal = CrPal(ColorsNum,PalEntry);
if(!hPal)
{
    MsgBox(NULL,122,MB_OK);
    return FALSE;
}
LocalUnlock(hGMap);
LocalFree(hGMap);
LocalUnlock(hLMap);
LocalFree(hLMap);
LocalUnlock(hPalEntry);
LocalFree(hPalEntry);
return TRUE;
}

/*-----GIFOpen-----*/
BOOL GIFOpen(HWND hWnd,FILE *gFile)
{
    INT ColorsNum;
    HANDLE hTmpImage;
    UCHAR far *TempImage;
    if(!GIFHeader(hWnd,gFile)) return FALSE;
    SetCursor(LoadCursor(NULL, IDC_WAIT));
    ColorsNum = 1<<mBitCount;
    rWidth = mWidth;
    if(mWidth%4)
        rWidth = mWidth+4 - mWidth%4;
}

```

```

mSize = rWidth * mHeighth;
if(hDImage !=NULL) GlobalFree(hDImage);
hDImage = (UCHAR far*)GlobalAlloc(GMEM_MOVEABLE,
mSize*sizeof(UCHAR));
if(hDImage == NULL) return FALSE;
DeGIF(gFile,4096);
if(mBitCount<8)
{
    MsgBox(NULL,123,MB_OK);
    return FALSE;
}
return TRUE;
}

/*-----GIFCreate-----*/
BOOL GIFCreate(HWND hWnd,HANDLE hImage,int pCode)
{
    HANDLE          hPalEntry;
    PALETTEENTRY    *PalEntry;
    HDC             hDC;
    register int    i;
    BITMAPINFO*    bInfo;
    UCHAR far*     Image;
    INT             ColorsNum;
    ColorsNum = 1<<mBitCount;
    if(hImage == NULL)      return FALSE;
    Image = (UCHAR far*) GlobalLock(hImage);
    hPalEntry = UserCrPal(pCode,hPal,ColorsNum);
    PalEntry = (PALETTEENTRY*)GlobalLock(hPalEntry);
    hCPal = CrPal(ColorsNum,PalEntry);
    hDC = GetDC(hWnd);
    SelectPalette(hDC,hCPal,TRUE);
    RealizePaleite(hDC);
    hBInfo = (BITMAPINFO*)LocalAlloc(GMEM_MOVEABLE,
sizeof(BITMAPINFOHEADER)+ ColorsNum*sizeof(RGBQUAD));
    bInfo = (BITMAPINFO*)LocalLock(hBInfo);
    bInfo->bmiHeader.biSize=40;
    bInfo->bmiHeader.biWidth = mWidth;
    bInfo->bmiHeader.biHeight = mHeighth;
    bInfo->bmiHeader.biPlanes = 1;
    bInfo->bmiHeader.biBitCount=mBitCount;
    bInfo->bmiHeader.biCompression = BI_RGB;
    bInfo->bmiHeader.biSizeImage=0;
    bInfo->bmiHeader.biXPelsPerMeter=0;
    bInfo->bmiHeader.biYPelsPerMeter=0;
    bInfo->bmiHeader.biCirUsed=0;
    bInfo->bmiHeader.biCirImportant=0;
}

```

```

for(i=0;i<ColorsNum;i++)
{
    bInfo->bmiColors[i].rgbRed = PalEntry[i].peRed;
    bInfo->bmiColors[i].rgbGreen = PalEntry[i].peGreen;
    bInfo->bmiColors[i].rgbBlue = PalEntry[i].peBlue;
}
hBmp = CreateDIBitmap(hDC,(BITMAPINFOHEADER FAR *) bInfo,
CBM_INIT,      Image,(BITMAPINFO FAR *) bInfo,DIB_RGB_COLORS);
if (!hBmp) return FALSE;
LocalUnlock(hbInfo);
LocalFree(hbInfo);
LocalUnlock(hPalEntry);
LocalFree(hPalEntry);
ReleaseDC(hWnd,hDC);
GlobalUnlock(hImage);
return TRUE;
}

/*=====Bitmap Routine=====*/
BOOL BMPOpen(HWND hWnd,FILE *bFile)
{
    BITMAPFILEHEADER  bHeader;
    BITMAPINFO          *bInfo;
    LOGPALETTE        far *LogPal;
    HANDLE            hMemPal,hBInfo,hMTemp;
    INT               NumColor;
    UINT              i, BPerLine, RealSize;
    UCHAR far *Image, far *MTemp;
    mType = 2;
    if(hDImage != NULL) GlobalFree(hDImage);
    fread(&bHeader,sizeof(BITMAPFILEHEADER),1,bFile);
    if (bHeader.bfType != 0x4d42)
    {
        MessageBox(hWnd,"File không hợp lệ","Thông báo", MB_OK);
        return FALSE;
    }
    // Bitmap Information
    hBInfo = (BITMAPINFO *)LocalAlloc (LMEM_MOVEABLE, bHeader.biOffBits
    - sizeof(BITMAPFILEHEADER));
    if(hBInfo == NULL) return FALSE;
    bInfo = ( BITMAPINFO * ) LocalLock(hBInfo);
    fread(bInfo,bHeader.biOffBits - sizeof(BITMAPFILEHEADER) ,1,bFile);
    if (bInfo->bmiHeader.biCompression != BI_RGB)
    {
        MessageBox(hWnd,"Lỗi giải nén file","Thông báo", MB_OK);
        return FALSE;
    }
    // Number of color in palette of the Bitmap File

```

```
if(bInfo->bmiHeader.biBitCount == 24)
{
    MessageBox(hWnd, "Không đọc được các loại ảnh Bitmap 24
bits", "Thông báo", MB_OK);
    return FALSE;
}
mWidth = bInfo->bmiHeader.biWidth;
mHeight= bInfo->bmiHeader.biHeight;
mBitCount = bInfo->bmiHeader.biBitCount;
rWidth = mWidth;
if(mWidth%4)
    rWidth = mWidth+4 - mWidth%4;
mSize = rWidth * mHeight;
NumColor = 1<< bInfo->bmiHeader.biBitCount;
// Number of Bytes per line
BPerLine = (bHeader.bfSize - bHeader.bfOffBits)/bInfo->bmiHeader.biHeight;
bInfo->bmiHeader.biSizeImage = bHeader.bfSize - bHeader.bfOffBits;
RealSize = bInfo->bmiHeader.biSizeImage;
// Create Palette
hMemPal =(LOGPALETTE far *)
LocalAlloc(LMEM_MOVEABLE,sizeof(LOGPALETTE) + 256 *
sizeof(PALETTEENTRY));
LogPal = (LOGPALETTE far *)LocalLock(hMemPal);
LogPal->palNumEntries = 256 ;
LogPal->palVersion = 0x300;
for (i = 0; i < NumColor ; i++)
{
    LogPal->palPalEntry[i].peRed = bInfo->bmiColors[i].rgbRed;
    LogPal->palPalEntry[i].peGreen = bInfo->
bmiColors[i].rgbGreen;
    LogPal->palPalEntry[i].peBlue = bInfo->bmiColors[i].rgbBlue ;
    LogPal->palPalEntry[i].peFlags = PC_RESERVED;
}
hPal = CreatePalette(LogPal);
LocalUnlock(hMemPal);
LocalFree(hMemPal);
if (!hPal) return FALSE;
LocalUnlock(hBInfo);
LocalFree(hBInfo);
// Read content
if(hDImage!=NULL) GlobalFree(hDImage);
hDImage = (UCHAR far*)GlobalAlloc(GMEM_MOVEABLE,
mSize*sizeof(UCHAR));
if (hDImage == NULL) return FALSE;
UCHAR Item,mByte,count,rest;
INT c,r,cc,MaxBytes;
if(mBitCount<8)
|
```

```
mType += mBitCount;
hMTemp = GlobalAlloc(GMEM_MOVEABLE,
RealSize*sizeof(UCHAR));
if(hMTemp==NULL) return FALSE;
MTemp = (UCHAR far*)GlobalLock(hMTemp);
fread(MTemp,RealSize,1,bFile);
Image = (UCHAR far*)GlobalLock(hDImage);
count = 8/mBitCount;
MaxBytes = (mWidth-1)/count;
rest = mWidth%count;
if(rest==0) rest = count;
for(r=0;r<mHeight;r++)
{
    cc=0;
    for(c=0;c<MaxBytes;c++)
    {
        mByte = MTemp[r*BPerLine+c];
        for(i=0;i<count;i++)
        {
            Item = mByte>>(8-mBitCount);
            Image[r*tWidth+cc] = Item;
            mByte <<= mBitCount;
            cc++;
        }
    }
    mByte = MTemp[r*BPerLine+c];
    for(i=0;i<rest;i++)
    {
        Item = mByte>>(8-mBitCount);
        Image[r*tWidth+cc] = Item;
        mByte <<= mBitCount;
        cc++;
    }
}
mBitCount = 8;
GlobalUnlock(hMTemp);
GlobalFree(hMTemp);
}

else
{
    Image = (UCHAR far*)GlobalLock(hDImage);
    fread(Image,RealSize,1,bFile);
}
fclose(bFile);
GlobalUnlock(hDImage);
return TRUE;
}
```

*Nội dung tệp ImgPro.CPP (chương trình chính)*

```

#include "StdAfx.h"
#include "Resource.h"
#include "Process.h"
#include "ImgPro.h"
#include "Util.h"
#define MAX_LOADSTRING 100
// Global Variables:
extern HINSTANCE      hInst;
extern HBITMAP         hBmp;
extern HPALETTE        hCPal;
extern HANDLE          hDImage;
extern BYTE             mType;
extern WORD             mWidth;
extern WORD             mHeight;
extern BYTE             mBitCount;
TCHAR szTitle[MAX_LOADSTRING];
// The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];
// The title bar text
// Forward declarations of functions included in this code module:
ATOM      MyRegisterClass(HINSTANCE hInstance);
BOOL      InitInstance(HINSTANCE, int);
LRESULT CALLBACK Starting(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK About(HWND, UINT, WPARAM, LPARAM);
LRESULT CALLBACK ImgInfo(HWND, UINT, WPARAM, LPARAM);
BOOL Start(HINSTANCE hInst, UINT message, WPARAM wParam, LPARAM lParam);
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow)

{
    MSG msg;
    HACCEL hAccelTable;
    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_IMGPRO, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }
    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_IMGPRO);
    // Main message loop:
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

```

```

        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}

// FUNCTION: MyRegisterClass()
// PURPOSE: Registers the window class.
// Các hàm này là cần thiết khi sử dụng Win32
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc   = (WNDPROC)WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, (LPCTSTR)IDI_IMGPROMO);
    wcex.hCursor         = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(COLOR_GRAYTEXT);
    wcex.lpszMenuName   = (LPCSTR)IDC_IMGPROMO;
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm         = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);
    return RegisterClassEx(&wcex);
}

// FUNCTION: InitInstance(HANDLE, int)
// PURPOSE: Saves instance handle and creates main window
// COMMENTS:
// In this function, we save the instance handle in a global variable and create and
// display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;
    hInst = hInstance; // Store instance handle in our global variable
    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
    if (!hWnd)
    {
        return FALSE;
    }
    //ShowWindow(hWnd, nCmdShow);
    ShowWindow(hWnd, SW_SHOWMAXIMIZED);
    UpdateWindow(hWnd);
}

```

```
return TRUE;
}

// FUNCTION: WndProc(HWND, unsigned, WORD, LONG)
// PURPOSE: Processes messages for the main window.
// WM_COMMAND - process the application menu
// WM_PAINT - Paint the main window
// WM_DESTROY - post a quit message and return
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hDC;
    switch (message)
    {
        case WM_CREATE:
            DialogBox(hInst, (LPCTSTR)IDD_START, hWnd,
(DLGPROC)Starting);
            break;
        case WM_PAINT:
            hDC = BeginPaint(hWnd,&ps);
            Display(hWnd,hBmp,hCPal);
            EndPaint(hWnd,&ps);
            break;
        case WM_COMMAND:
            wmId = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
                case IDM_ABOUT:
                    DialogBox(hInst, (LPCTSTR)IDD_ABOUTBOX,
hWnd, (DLGPROC)About);
                    break;
                case IDM_IMGINFO:
                    DialogBox(hInst, (LPCTSTR)IDD_IMGINFO, hWnd,
(DLGPROC)ImgInfo);
                    break;
                case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;
            default:
                DoCommand(hWnd,wmId,wParam,
lParam);
                return
DefWindowProc(hWnd,message,wParam,lParam);
            }
    }
}
```

```
        }
        break;
    case WM_DESTROY:
        RemoveTemp();
        ClearReSource();
        PostQuitMessage(0);
        break;
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
}
return 0;
}

// Message handler for about box.
LRESULT CALLBACK About(HWND hDlg, UINT message, WPARAM wParam,
LPARAM lParam)
{
switch (message)
{
    case WM_INITDIALOG:
        return TRUE;
    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) ==
IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return TRUE;
        }
        break;
}
return FALSE;
}

LRESULT CALLBACK ImgInfo(HWND hDlg, UINT message, WPARAM wParam,
LPARAM lParam)
{
    INT ColorsNum;
    switch(mType)
    {
        case 0:
            ColorsNum = 256;
            break;
        case 3:
            ColorsNum = 2;
            break;
        case 6:
            ColorsNum = 16;
            break;
        default:
            ColorsNum = 1<<mBitCount;
    }
}
```

```

    }
    switch (message)
    {
        case WM_INITDIALOG:
            SetDlgItemInt(hDlg, IDC_INPOWIDTH,(int)mWidth, FALSE);
            SetDlgItemInt(hDlg, IDC_INFOHEIGHT,(int)mHeighth, FALSE);
            SetDlgItemInt(hDlg, IDC_INFOCOLOR,ColorsNum, FALSE);
            break;
        case WM_COMMAND:
            if (LOWORD(wParam) == IDOK)
            {
                EndDialog(hDlg, LOWORD(wParam));
                break;
            }
            break;
    }
    return FALSE;
}
LRESULT CALLBACK Starting(HWND hDlg, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
        case WM_INITDIALOG:
            return TRUE;
        case WM_MOUSEMOVE:
            EndDialog(hDlg, LOWORD(wParam));
            return TRUE;
        case WM_MBUTTONDOWN:
            EndDialog(hDlg, LOWORD(wParam));
            return TRUE;
        case WM_KEYDOWN:
            EndDialog(hDlg, LOWORD(wParam));
            return TRUE;
    }
    return FALSE;
}

```

### Modul mô phỏng các bộ lọc

```

#include "StdAfx.h"
#include "Filters.h"
#include <limits.h>
//Local SubFunction
int GetCurPixel(UCHAR far* IMatrix,int mWidth,int mHeighth,
    int mCol,int mRow,int mBitCount);

```

```
FILTER* CreateFilter(LPSTR Name,int M00,int M01,int M02,int M10, int M11,
                     int M12, int M20,int M21,int M22,int COEF)
```

```
{
    FILTER *RetF;
    RetF = (FILTER*)LocalAlloc(GMEM_FIXED,sizeof(FILTER));
    RetF->Name= Name;
    RetF->M00 = M00;
    RetF->M01 = M01;
    RetF->M02 = M02;
    RetF->M10 = M10;
    RetF->M11 = M11;
    RetF->M12 = M12;
    RetF->M20 = M20;
    RetF->M21 = M21;
    RetF->M22 = M22;
    RetF->COEF = COEF;
```

```
return RetF;
```

```
}
```

```
BOOL InitFilters(FILTER *F[Nmax])
```

```
{
```

```
    F[0] = CreateFilter((LPSTR) "Lọc Laplace",
                        0,-1,0,-1,4,-1,0,-1,0, 1);
    F[1] = CreateFilter((LPSTR) "Lọc Trung vị",
                        1,1,1,1,1,1,1,1,1,9);
    F[2] = CreateFilter((LPSTR) "Lọc Thông thấp",
                        1,1,1,1,2,1,1,1,1,10);
    F[3] = CreateFilter((LPSTR) "Lọc Thông cao",
                        -1,-1,-1,-1,8,-1,-1,-1,-1,1);
    F[4] = CreateFilter((LPSTR) "Lọc Binomial-Gausse",
                        1,2,1,2,4,2,1,2,1,16);
```

```
return TRUE;
```

```
}
```

```
// SubFunction of FilIn()
```

```
BOOL FilIn(HANDLE hMatrix,int mWidth,int mHeigth,
           int mBitCount,FILTER F)
{
    HANDLE hTemp;
    UCHAR far* IMatrix;
    int CurFilter[9], CurMask[9],far* TempMatrix,size,coef;
    register int mCol,mRow,OPixel,i;
    int NLevel;
    OPixel = 0;
    NLevel = (1<<mBitCount)-1;
    if(mWidth%4) mWidth += (4 - mWidth%4);
```

```
size = mWidth * mHeight;
hTemp = (int far*)GlobalAlloc (GMEM_MOVEABLE, size * sizeof(int));
TempMatrix = (int far*)GlobalLock(hTemp);
IMatrix = (UCHAR far*)GlobalLock(hIMatrix);
CurFilter[0] = F.M22;
CurFilter[1] = F.M21;
CurFilter[2] = F.M20;
CurFilter[3] = F.M12;
CurFilter[4] = F.M11;
CurFilter[5] = F.M10;
CurFilter[6] = F.M02;
CurFilter[7] = F.M01;
CurFilter[8] = F.M00;
Coef      = F.COEF;
for(mRow=0;mRow<mHeight;mRow++)
for(mCol=0;mCol<mWidth;mCol++)
{
    CurMask[0] = GetCurPixel(IMatrix,mWidth,mHeight,mCol-1,mRow+1,mBitCount);
    CurMask[1] = GetCurPixel(IMatrix,mWidth,mHeight,mCol,mRow+1,mBitCount);
    CurMask[2] = GetCurPixel(IMatrix,mWidth,mHeight,mCol+1,mRow+1,mBitCount);
    CurMask[3] = GetCurPixel(IMatrix,mWidth,mHeight,mCol-1,mRow, mBitCount);
    CurMask[4] = GetCurPixel(IMatrix,mWidth,mHeight,mCol,mRow, mBitCount);
    CurMask[5] = GetCurPixel(IMatrix,mWidth,mHeight,mCol+1,mRow, mBitCount);
    CurMask[6] = GetCurPixel(IMatrix,mWidth,mHeight,mCol-1,mRow-1, mBitCount);
    CurMask[7] = GetCurPixel(IMatrix,mWidth,mHeight,mCol,mRow-1, mBitCount);
    CurMask[8] = GetCurPixel(IMatrix,mWidth,mHeight,mCol+1,mRow-1, mBitCount);
    OPixel = 0;
    for(i=0;i<9;i++)
        OPixel += CurMask[i] * CurFilter[i];
    TempMatrix[mRow*mWidth + mCol] = OPixel/coef;
}
for(i=0;i<size;i++)
{
    if(TempMatrix[i]>NLevel) TempMatrix[i] = NLevel;
    if(TempMatrix[i]<0) TempMatrix[i] = 0;
    IMatrix[i] = (UCHAR)TempMatrix[i];
}
GlobalUnlock(hIMatrix);
```

```
GlobalUnlock(hTemp);
GlobalFree(hTemp);
return TRUE;
}

int GetCurPixel(UCHAR far* IMatrix,int mWidth,int mHeighth,
                int mCol,int mRow,int mBitCount)
{
    UCHAR ret;
    if ((mCol <0)|| (mRow <0)|| (mCol >mWidth)|| (mRow >mHeighth)) return
0;
    ret = IMatrix[mRow * mWidth + mCol];
    return (int) ret;
}

int SetCurPixel(UCHAR Val,UCHAR far* IMatrix,int mWidth,int mHeighth,
int mCol,int mRow,int mBitCount)
{
if ((mCol <0)|| (mRow <0)|| (mCol >mWidth)|| (mRow >mHeighth)) return -1;
IMatrix[mRow * mWidth + mCol] = Val;
return 0;
}
```

## MỤC LỤC

	Trang
Lời tựa (cho lần xuất bản thứ ba) .....	3
Lời mở đầu .....	5
<b>Chương 1: NHẬP MÔN XỬ LÝ ẢNH</b>	
1.1. Tổng quan về một hệ thống xử lý ảnh .....	7
1.2. Các vấn đề cơ bản trong xử lý ảnh .....	9
1.2.1. Một số khái niệm .....	9
1.2.2. Biểu diễn ảnh .....	11
1.2.3. Tăng cường ảnh - khôi phục ảnh .....	11
1.2.4. Biến đổi ảnh .....	12
1.2.5. Phân tích ảnh .....	13
1.2.6. Nhận dạng ảnh .....	14
1.2.7. Nén ảnh .....	14
Bài tập chương I .....	15
<b>Chương 2: THU NHẬN ẢNH</b>	
2.1. Các thiết bị thu nhận ảnh và kỹ thuật phân tích màu .....	17
2.1.1. Thiết bị thu nhận ảnh .....	17
2.1.2. Biểu diễn màu .....	18
2.1.3. Tổng hợp màu và xách màu .....	20
2.1.4. Hệ toạ độ màu .....	21
2.2. Lấy mẫu và lượng tử hóa (Image Sampling and quantization) .....	22
2.2.1. Lấy mẫu (Image sampling) .....	23
2.2.2. Lượng tử hóa ảnh (Image Quantization) .....	25
2.2.2.1. Khái niệm và nguyên tắc lượng tử hóa ảnh .....	25
2.2.2.2. Kỹ thuật lượng tử hóa trung bình bình phương cực tiểu .....	26

<b>2.3. Một số phương pháp biểu diễn ảnh (image representation).....</b>	<b>29</b>
2.3.1. Mô hình dải .....	29
2.3.2. Mã xích .....	30
2.3.3. Mã tử phân .....	31
<b>2.4. Các định dạng ảnh cơ bản trong xử lý ảnh .....</b>	<b>31</b>
2.4.1. Khái niệm chung .....	31
2.4.2. Qui trình đọc 1 tệp ảnh .....	32
<b>2.5. Các kỹ thuật tái hiện ảnh .....</b>	<b>32</b>
2.5.1. Kỹ thuật chụp ảnh (photography hardcopy techniques).....	32
2.5.2. Kỹ thuật in ảnh (Printer Hardcopy techniques).....	32
<b>2.6. Khái niệm ảnh đen trắng, đa cấp xám, ảnh màu .....</b>	<b>38</b>
2.6.1. Ảnh đen trắng .....	38
2.6.2. Ảnh màu .....	38
Bài tập chương 2 .....	39

### **Chương 3: CÔNG CỤ TRỢ GIÚP XỬ LÝ ẢNH SỐ**

<b>3.1. Tổng quan về xử lý ảnh trong không gian .....</b>	<b>41</b>
3.1.1. Tín hiệu số và biểu diễn ảnh số .....	41
3.1.2. Khái quát về hệ thống xử lý tín hiệu số .....	42
<b>3.2. Các toán tử không gian (Spatial operators).....</b>	<b>43</b>
a) Toán tử tuyến tính .....	43
b) Tích chập .....	44
c) Kỹ thuật lọc số .....	49
<b>3.3. Các biến đổi không Gian: Biến đổi Fourier và biến đổi KL (spatial transforms).....</b>	<b>54</b>
3.3.1. Biến đổi Fourier .....	56
3.3.1.1. Biến đổi Fourier-FT: khái niệm và công thức .....	56
3.3.1.2. Biến đổi Fourier rời rạc - DFT .....	57

3.3.1.3. Một số tính chất và áp dụng .....	59
3.3.2 Biến đổi KL.....	61
3.3.2.1 Một số định nghĩa và khái niệm.....	61
3.3.2.2 Cơ sở lý thuyết của biến đổi KL.....	62
3.3.2.3. Biến đổi KL .....	63
3.4. Toán tử xử lý điểm ảnh.....	64
3.4.1. Xử lý điểm ảnh bằng ảnh xạ biến đổi .....	65
3.4.2 Lược đồ mức xám (histogram).....	66
3.4.3. Biến đổi lược đồ xám .....	67
3.5. Mô hình thống kê .....	72
3.5.1. Mô hình 1 chiều nhân quả (one dimensional causal model).....	73
3.5.2. Mô hình nhân quả hai chiều.....	74
Bài tập chương 3.....	75

## **Chương 4: XỬ LÝ VÀ NÂNG CAO CHẤT LƯỢNG ẢNH**

4.1.1. Cải thiện ảnh dùng toán tử điểm .....	78
4.1.1.1. Tăng độ tương phản (stretching contrast).....	78
4.1.1.2. Tách nhiễu và phân ngưỡng .....	79
4.1.1.3 Biến đổi âm bản (Digital Negative).....	81
4.1.1.5. Trích chọn bit (Bit Extraction).....	83
4.1.1.6. Trữ ảnh.....	83
4.1.1.7. Nén dài độ sáng.....	83
4.1.1.8. Mô hình hóa và biến đổi lược đồ xám.....	84
4.1.2. Cải thiện ảnh dùng toán tử không gian .....	85
4.1.2.1. Làm tròn nhiễu bằng lọc tuyến tính: lọc trung bình và lọc dài thông thấp .....	85
4.1.2.2. Làm tròn nhiễu bằng lọc phi tuyến .....	89
4.1.2.3. Mật nén giữ sai phân và làm nhẵn (Unsharp Masking and Crispering) .....	92

4.1.2.5. Khuếch đại và nội suy ảnh .....	94
4.1.3. Một số kỹ thuật cải thiện ảnh nhị phân .....	96
4.1.3.1. Dân ảnh .....	96
4.1.3.2. Cờ ảnh .....	97
4.2. Khôi phục ảnh (Image restoration).....	97
4.2.1. Các mô hình quan sát và tạo ảnh.....	98
4.2.2. Kỹ thuật lọc tuyến tính.....	102
4.2.2.1. Kỹ thuật lọc ngược (Inverse filter).....	102
4.2.2.2. Lọc giả ngược (Pseudoinverse Filter).....	103
4.2.2.3. Lọc Wiener.....	103
4.2.2.4. Lọc Wiener với đáp ứng xung hữu hạn FIR (Finite Impulse Response).....	105
4.2.2.5. Kỹ thuật làm tròn spline và nội suy.....	106
4.2.3. Kỹ thuật lọc phi tuyến trong khôi phục ảnh .....	108
4.2.3.1. Lọc nhiễu dốm .....	109
4.2.3.2. Kỹ thuật entropy cực đại .....	110
4.2.3.3. Phương pháp Bayesian .....	111
4.2.3.4. Giải chập mù (Blind deconvolution) .....	112
Bài tập chương 4.....	113

## *Chương 5: CÁC PHƯƠNG PHÁP PHÁT HIỆN BIÊN*

5.1. Tổng quan về phân tích ảnh (Image analysis).....	115
5.2. Khái quát về biên và phân loại các kỹ thuật dò biên .....	115
5.2.1. Khái niệm về biên .....	115
5.2.2. Phân loại các kỹ thuật phát hiện biên .....	117
5.2.3. Quá trình phát hiện biên .....	117
5.3. Phương pháp phát hiện biên cục bộ.....	118
5.3.1. Phương pháp gradient .....	118

5.3.1.1. Kỹ thuật gradient.....	119
5.3.1.2. Toán tử la bàn.....	122
5.3.2. Kỹ thuật Laplace .....	123
5.3.3. Kỹ thuật đạo hàm tích chập - phương pháp Canny .....	125
<b>5.4. Dò biên theo quy hoạch động.....</b>	<b>127</b>
5.4.1. Mô tả phương pháp .....	127
5.4.2. Thuật toán .....	130
<b>5.5. Một số phương pháp khác .....</b>	<b>133</b>
5.5.1. Tiếp cận theo mô hình mịt .....	133
5.5.2. Tiếp cận tối ưu hoá.....	134
<b>Bài tập chương 5 .....</b>	<b>135</b>

## *Chương 6: PHÂN VÙNG ẢNH*

<b>6.1. Tổng quan về phân vùng ảnh.....</b>	<b>137</b>
<b>6.2. Phân vùng ảnh dựa theo ngưỡng biên độ.....</b>	<b>137</b>
<b>6.3. Phân vùng theo miền đóng nhất .....</b>	<b>140</b>
6.3.1. Phương pháp tách cây từ phân.....	142
6.3.2. Phương pháp cục bộ hay phân vùng bồi hợp.....	144
6.3.2.1. Thuật toán tô màu .....	146
6.3.2.2. Thuật toán đếm quy cục bộ .....	146
6.3.3. Phương pháp tổng hợp.....	147
<b>6.4. Phân vùng dựa theo đường biên.....</b>	<b>148</b>
6.4.1. Làm mảnh biên .....	149
6.4.2. Nhị phân hoá đường biên .....	151
6.4.3. Miêu tả đường biên .....	151
6.4.3.1. Mã hoá theo toạ độ Décác .....	153
6.4.3.2. Mã hoá Freeman .....	154
6.4.3.3. Xấp xỉ bởi đoạn thẳng .....	156

---

6.4.3.4. Xấp xỉ đa thức .....	159
6.4.4. Nới lỏng thuật toán phân vùng (Relaxation in Algorithms in region analysis) .....	160
6.5. Phân đoạn dựa theo kết cấu bề mặt (Texture).....	162
6.5.1. Tiếp cận thống kê .....	162
6.5.2. Cách tiếp cận cấu trúc .....	165
6.5.3. Phân đoạn theo tính kết cấu .....	166
Bài tập chương 6 .....	166

## **Chương 7: NHẬN DẠNG ẢNH**

7.1. Tổng quan về nhận dạng .....	169
7.1.1. Không gian biểu diễn đối tượng, không gian diễn dịch .....	170
7.1.2. Mô hình và bản chất của quá trình nhận dạng .....	171
7.1.2.1. Mô hình .....	171
7.1.2.2. Bản chất của quá trình nhận dạng .....	173
7.2. Nhận dạng dựa trên phân hoạch không gian.....	174
7.2.1. Phân hoạch không gian .....	174
7.2.2. Hàm phân lớp hay hàm ra quyết định .....	174
7.2.3. Nhận dạng thống kê .....	175
7.2.4. Một số thuật toán nhận dạng tiêu biểu trong tự học .....	177
7.2.4.1. Thuật toán dựa vào khoảng cách lớn nhất .....	177
7.2.4.2. Thuật toán K trung bình (giả sử có K lớp) .....	178
7.2.4.3. Thuật toán ISODATA .....	178
7.3. Nhận dạng theo cấu trúc .....	179
7.3.1. Biểu diễn định tĩnh .....	179
7.3.2. Phương pháp ra quyết định dựa vào cấu trúc .....	179
7.3.2.1. Một số khái niệm .....	179
7.3.2.2. Phương pháp nhận dạng .....	180

<b>7.4. Mạng nơron nhân tạo và nhận dạng theo mạng nơron .....</b>	<b>181</b>
7.4.1. Bộ não và nơron sinh học .....	181
7.4.2. Mô hình mạng nơron nhân tạo .....	184
7.4.2.1. Mô hình nơron nhân tạo .....	184
7.4.2.2. Mạng nơron .....	185
7.4.3. Các mạng nơron một lớp .....	190
7.4.3.1. Mạng Hopfield .....	190
7.4.3.2. Mạng kiểu bộ nhớ 2 chiều kết hợp thích nghi ( <u>Adaptive Bidirectional Associative Memory Neural Network</u> ) .....	195
7.4.3.3. Mạng Kohonen .....	201
7.4.3.4. Mạng Perceptron .....	205
7.4.4. Các mạng nơron nhiều lớp (Multi-layer Neural Network) .....	207
7.4.4.1 Mạng nơron nhiều lớp lan truyền ngược sai số (Back-propagation Neural Network) .....	207
7.4.4.2. Mạng nơron nhiều lớp ngược hướng (Counter-propagation Neural Network) .....	212
7.4.5. Ứng dụng mạng nơron lan truyền ngược hướng cho nhận dạng ký tự .....	213
7.4.5.1. Mở đầu .....	213
7.4.5.2. Nhận dạng bằng mạng nơron lan truyền ngược hướng .....	213
7.4.5.3. Cài đặt mạng lan truyền ngược hướng cho nhận dạng ký tự .....	215
7.4.5.4. Nhận dạng 37 ký tự sử dụng mạng lan truyền ngược hướng .....	216
7.5.1. Sơ đồ tổng quát của một hệ nhận dạng chữ .....	218
7.5.2. Giai đoạn xử lý sơ bộ .....	218
7.5.3. Giai đoạn tách chữ .....	221
7.5.4. Một số thuật toán nhận dạng chữ .....	222
7.5.4.1. Kỹ thuật nhận dạng chữ in .....	222
7.5.4.2 Kỹ thuật nhận dạng chữ viết tay có ràng buộc .....	224
7.5.4.3. Thuật toán nhận dạng chữ tổng quát .....	225

---

7.6. Kết luận.....	228
--------------------	-----

## **Chương 8: NÉN DỮ LIỆU ẢNH**

8.1. Tổng quan về nén dữ liệu ảnh.....	229
8.1.1. Một số khái niệm .....	229
8.1.2. Các loại dữ thừa dữ liệu.....	230
8.1.3. Phân loại các phương pháp nén .....	231
8.2. Các phương pháp nén thế hệ thứ nhất.....	233
8.2.1. Phương pháp mã hoá loạt dài .....	233
8.2.2. Phương pháp mã hoá Huffman.....	234
8.2.3. Phương pháp LZW .....	236
8.2.4. Phương pháp mã hoá khối (Block Coding) .....	242
8.2.5. Phương pháp thích nghi.....	243
8.3. Phương pháp mã hoá dựa vào biến đổi thế hệ thứ nhất.....	244
8.3.1. Nguyên tắc chung .....	244
8.3.2. Thuật toán mã hoá dùng biến đổi 2 chiều .....	245
8.3.3. Mã hoá dùng biến đổi Cosin và chuẩn JPEG .....	246
8.3.3.1. Phép biến đổi Cosin một chiều .....	243
8.3.3.2. Phép biến đổi Cosin rời rạc hai chiều .....	252
8.3.3.3. Biến đổi Cosin và chuẩn nén JPEG .....	253
8.4. Phương pháp mã hoá thế hệ thứ hai.....	262
8.4.1. Phương pháp Kim tự tháp Laplace (Pyramide Laplace) .....	263
8.4.2. Phương pháp mã hoá dựa vào biểu diễn ảnh .....	263
8.4.2.1 Mã hoá dựa vào vùng giá tăng .....	264
8.4.2.2 Phương pháp tách - hợp .....	264
8.5. Kết luận.....	265
Bài tập chương 8.....	266

1. Định dạng ảnh IMG .....	267
2. Định dạng ảnh PCX .....	268
3. Định dạng ảnh TIFF .....	269
4. Định dạng ảnh GIF(Graphics Interchanger Format) .....	271
5. Định dạng ảnh JPEG(Joint Photograph Expert Group) .....	274
5.1. Sơ lược lịch sử JPEG .....	274
5.2. Hệ thống vạch dấu và cách tổ chức thông tin trong tệp JPG .....	275
5.3. Định dạng JPG/JFIF ( Jpeg Format Interchange File) .....	277
5.4. Mẫu dữ liệu một tệp ảnh JPEG .....	282
<i>Phụ lục B</i>	
1. Nhóm chương trình nạp và lưu ảnh .....	283
2. Nhóm chương trình xử lý ảnh (viết trên Turbo C) .....	294
3. Nhóm chương trình biến đổi và nén ảnh (viết trên Visual C) .....	305
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>355</b>



Khoa CNTT

Các sách Công nghệ Thông tin đã xuất bản tại NXB KH và Khoa CNTT

» **Bài tập lập trình ngôn ngữ C -1999, tái bản 2001, 2003**

Nguyễn Thanh Thuỷ, Nguyễn Quang Huy.

» **Kỹ thuật đồ họa - 2000, tái bản 2002, 2003**

Lê Tấn Hùng, Huỳnh Quyết Thắng.

» **Công nghệ phần mềm - 2002**

Lê Đức Trung.

» **Nhập môn lập trình ngôn ngữ C - 1999, tái bản 2001, 2003**

Lê Đăng Hưng, Trần Việt Linh, Lê Đức Trung, Nguyễn Thanh Thuỷ.

» **Lập trình hướng đối tượng với C++ - 2000, tái bản 2003**

Tạ Tuán Anh, Nguyễn Hữu Đức, Lê Đăng Hưng, Nguyễn Thanh Thuỷ.

» **Bài tập lập trình hướng đối tượng với C++ - 2000, tái bản 2003**

Tạ Tuán Anh, Nguyễn Hữu Đức, Nguyễn Quang Huy,

Nguyễn Thanh Thuỷ.

» **Nhập môn hệ điều hành LINUX- 2000**

Đinh Lan Anh, Nguyễn Hữu Đức, Nguyễn Quang Huy,

Nguyễn Thanh Thuỷ

» **Quản trị hệ thống LINUX- 2000**

Lê Quân, Ngô Hồng Sơn, Trương Diệu Linh, Nguyễn Thanh Thuỷ

» **Ngôn ngữ hình thức - 2002**

Nguyễn Văn Ba.

» **STL - Lập trình khái lược trong C++ - 2003**

Nguyễn Hữu Đức, Đặng Công Kiên,

Doãn Trung Tùng, Nguyễn Thanh Thuỷ

2 0 7 0 3 3



GIÁ: 57.000đ

Nhập môn xử lý ảnh số