MỤC LỤC

LỜI CẢM ƠN

Trước tiên em xin gởi lời cảm ơn sâu sắc đến Thầy **Ngô Quốc Tạo** và các thầy cô giáo bộ môn nghành công nghệ thông tin đã tạo mọi điều kiện về cơ sở vật chất và tinh thần giúp đở hướng dẫn em trong trong thời gian làm đồ án tốt nghiệp.

Em xin cảm ơn các thầy giáo, cô giáo Khoa Công Nghệ Thông Tin Trường Đại học Dân Lập Hải Phòng đã trang bị kiến thức cho em những kiến thức cần thiết và bổ ích để hoàn thành đồ án này.

Do thời gian và kiến thức còn hạn chế nên đồ án không tránh khỏi những sai sót. Em mong nhận được sự đóng góp bổ sung của thầy cô giáo và các bạn.

Cuối cùng xin chân thành cảm ơn tất cả các bạn đã đóng góp ý kiến và hổ trợ em trong quá trình thực hiện thành đồ án này.

Hải Phòng , Tháng 7-2010

Nguyễn Tiến Mạnh

LỜI NÓI ĐẦU

Ngày nay với sự phát triển như vũ bảo của công nghệ thông tin. Nó đã đem lại những ứng dụng to lớn trong nhiều lĩnh vực khác nhau. Công nghệ thông tin đã trở thành ngành công nghiệp mũi nhọn của nhiều nước trên thế giới. Sự tồn tại và phát triển của một doanh nghiệp, cơ quan, tổ chức nhà nước...Không thể thiếu sự trợ giúp của máy tính.

Trong việc quản lý, thu nhận và xử lý thông tin với khối lượng ngày càng lớn, nhiều lúc với những phần mềm thủ công không đem lại hiệu quả mong muốn, tốn nhiều công sức và thời gian.

Nhằm đem lại sự nhanh chóng và chính xác, đở tốn công sức của con người. Trong những thập niên gần đây nhiều nhà nghiên cứu đã phát triển mạnh mẽ bài toán nhập liệu tự động.

Nhập liệu tự động là việc nạp thông tin vào máy không thông qua những tác động thủ công của con người.

Tuy nhiên trong thực tế để cài một hệ nhập liệu tự động cụ thể gặp khá nhiều khó khăn.

Để phần nào khắc phục các nhược điểm trên. Đồ án tiến hành nghiên cứu một số thuật toán hiệu chỉnh những nhược điểm của nhập liệu tự động, và bước đầu cài đặt thử nghiệm bài toán nhập liệu tự động(nhận dạng phiếu điều tra).

Cấu trúc luận văn gồm 3 chương:

Chương I: Tổng quan về xử lý ảnh

Trong chương này luận văn nghiên cứu phần tổng quan của xử lý ảnh, phạm vi ứng dụng của xử lý ảnh, các tệp trong xử lý ảnh và sự cần thiết sự phát hiện độ dich chuyển của phiếu điều tra so với phiếu mẫu

Chương II: Nghiên cứu các kỹ thuật phát hiện độ dịch chuyển của phiếu điều tra và bài tóan ứng dụng

Trong chương này nghiên cứu các thuật toán nhằm giải quyết các khó khăn đã được nêu trong chương I. Ở đây đưa ra các phương pháp xác định độ dịch chuyển trang

văn bản và sau đó chọn phương pháp so sánh Histogram để đi sâu nghiên cứu và cài đặt thử nghiệm chương trình.

Chương III: Cài đặt chương trình và hướng nâng cao.

Chương cuối cùng này đồ án đưa ra kết quả chương trình và hướng nâng cao của luận.

Do thời gian và kiến thức còn hạn chế nên luận văn không tránh khỏi những sai sót mong các thầy cô giáo và các bạn đóng góp ý kiến.

Hải Phòng, Tháng 7/2010

CHƯƠNG I

TỔNG QUAN VỀ XỬ LÝ ẢNH

1.1 Tổng quan về xử lý ảnh

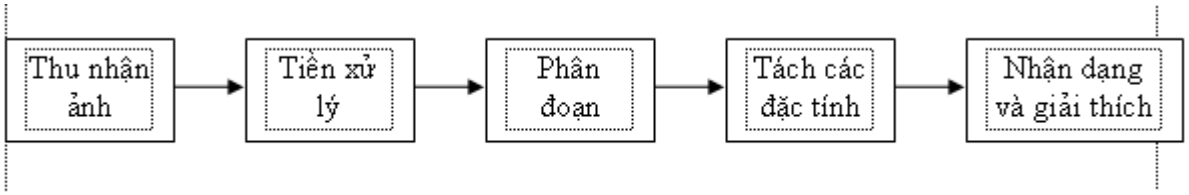
Xử lý ảnh (XLA) là đối tượng nghiên cứu của lĩnh vực thị giác máy, là quá trình biến đổi từ một ảnh ban đầu sang một ảnh mới với các đặc tính và tuân theo ý muốn của người sử dụng. Xử lý ảnh có thể gồm quá trình phân tích, phân lớp các đối tượng, làm tăng chất lượng, phân đoạn và tách cạnh, gán nhãn cho vùng hay quá trình biên dịch các thông tin hình ảnh của ảnh.

Cũng như xử lý dữ liệu bằng đồ hoạ, xử lý ảnh số là một lĩnh vực của tin học ứng dụng. Xử lý dữ liệu bằng đồ họa đề cập đến những ảnh nhân tạo, các ảnh này được xem xét như là một cấu trúc dữ liệu và được tạo bởi các chương trình. Xử lý ảnh số bao gồm các phương pháp và kỹ thuật biến đổi, để truyền tải hoặc mã hoá các ảnh tự nhiên. Mục đích của xử lý ảnh gồm:

- Biến đổi ảnh làm tăng chất lượng ảnh.
- Tự động nhận dạng ảnh, đoán nhận ảnh, đánh giá các nội dung của ảnh.

Nhận biết và đánh giá các nội dung của ảnh là sự phân tích một hình ảnh thành những phần có ý nghĩa để phân biệt đối tượng này với đối tượng khác, dựa vào đó ta có thể mô tả cấu trúc của hình ảnh ban đầu. Có thể liệt kê một số phương pháp nhận dạng cơ bản như nhận dạng ảnh của các đối tượng trên ảnh, tách cạnh, phân đoạn hình ảnh,… Kỹ thuật này được dùng nhiều trong y học (xử lý tế bào, nhiễm sắc thể), nhận dạng chữ trong văn bản.

1.2 Các quá trình xử lý ảnh



Hinh 1.1 Các giai đoạn chính trong xử lý ảnh

Thu nhận ảnh: Đây là công đoạn đầu tiên mang tính quyết định đối với quá trình XLA. Ảnh đầu vào sẽ được thu nhận qua các thiết bị như camera, sensor, máy scanner,v.v… và sau đó các tín hiệu này sẽ được số hóa. Việc lựa chọn các thiết bị thu nhận ảnh sẽ phụ thuộc vào đặc tính của các đối tượng cần xử lý. Các thông số quan trọng ở bước này là độ phân giải, chất lượng màu, dung lượng bộ nhớ và tốc độ thu nhận ảnh của các thiết bị.

Tiền xử lý: Ở bước này, ảnh sẽ được cải thiện về độ tương phản, khử nhiễu, khử bóng, khử độ lệch,v.v… với mục đích làm cho chất lượng ảnh trở lên tốt hơn nữa, chuẩn bị cho các bước xử lý phức tạp hơn về sau trong quá trình XLA. Quá trình này thường được thực hiện bởi các bộ lọc.

Phân đoạn ảnh: phân đoạn ảnh là bước then chốt trong XLA. Giai đoạn này 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 màu, cùng mức xám v.v… Mục đích của phân đoạn ảnh là để có một miêu tả tổng hợp về nhiều phần tử khác nhau cấu tạo lên ảnh thô. Vì lượng thông tin chứa trong ảnh rất lớn, trong khi đa số các ứng dụng chúng ta chỉ cần trích một vài đặc trưng nào đó, do vậy cần có một quá trình để giảm lượng thông tin khổng lồ đó. Quá trình này bao gồm phân vùng ảnh và trích chọn đặc tính chủ yếu.

Tách các đặc tính: Kết quả của bước phân đoạn ảnh thường được cho dưới dạng dữ liệu điểm ảnh thô, trong đó hàm chứa biên của một vùng ảnh, hoặc tập hợp tất cả các điểm ảnh thuộc về chính vùng ảnh đó. Trong cả hai trường hợp, sự chuyển đổi dữ liệu thô này thành một dạng thích hợp hơn cho việc xử lý trong máy tính là rất cần thiết. Để chuyển đổi chúng, câu hỏi đầu tiên cần phải trả lời là nên biểu diễn một vùng ảnh dưới dạng biên hay dưới dạng một vùng hoàn chỉnh gồm tất cả những điểm ảnh thuộc về nó. Biểu diễn dạng biên cho một vùng phù hợp với những ứng dụng chỉ quan tâm chủ yếu đến các đặc trưng hình dạng bên ngoài của đối tượng, ví dụ như các góc cạnh và điểm uốn trên biên chẳng hạn. Biểu diễn dạng vùng lại thích hợp cho những ứng dụng khai thác các tính chất bên trong của đối tượng, ví dụ như vân ảnh hoặc cấu trúc xương của nó. Sự chọn lựa cách biểu diễn thích hợp cho một vùng ảnh chỉ mới là

một phần trong việc chuyển đổi dữ liệu ảnh thô sang một dạng thích hợp hơn cho các xử lý về sau. Chúng ta còn phải đưa ra một phương pháp mô tả dữ liệu đã được chuyển đổi đó sao cho những tính chất cần quan tâm đến sẽ được làm nổi bật lên, thuận tiện cho việc xử lý chúng.

Nhận dạng và giải thích: Đây là bước cuối cùng trong quá trình XLA. Nhận dạng ảnh có thể được nhìn nhận một cách đơn giản là việc gán nhãn cho các đối tượng trong ảnh. Ví dụ đối với nhận dạng chữ viết, các đối tượng trong ảnh cần nhận dạng là các mẫu chữ, ta cần tách riêng các mẫu chữ đó ra và tìm cách gán đúng các ký tự của bảng chữ cái tương ứng cho các mẫu chữ thu được trong ảnh. Giải thích là công đoạn gán nghĩa cho một tập các đối tượng đã được nhận biết.

Chúng ta cũng có thể thấy rằng, không phải bất kỳ một ứng dụng XLA nào cũng bắt buộc phải tuân theo tất cả các bước xử lý đã nêu ở trên, ví dụ như các ứng dụng chỉnh sửa ảnh nghệ thuật chỉ dừng lại ở bước tiền xử lý. Một cách tổng quát thì những chức năng xử lý bao gồm cả nhận dạng và giải thích thường chỉ có mặt trong hệ thống phân tích ảnh tự động hoặc bán tự động, được dùng để rút trích ra những thông tin quan trọng từ ảnh, ví dụ như các ứng dụng nhận dạng ký tự quang học, nhận dạng chữ viết tay v.v…

1.3. Ảnh và biểu diễn ảnh

Ảnh trong thực tế là một ảnh liên tục cả về không gian và giá trị độ sáng. Để có thể xử lý ảnh bằng máy tính thì cần thiết phải tiến hành số hóa ảnh. Quá trình số hóa biến đổi các tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng tử hóa các thành phần giá trị mà về nguyên tắc bằng mắt thường không thể phân biệt được hai điểm liền kề nhau. Các điểm như vậy được gọi là các pixel (Picture Element) hay các phần tử ảnh hoặc điểm ảnh. Ở đây cần phân biệt khái niệm pixel hay đề cập đến trong các hệ thống đồ họa máy tính. Để tránh nhầm lẫn ta gọi khái niệm pixel này là pixel thiết bị. Khái niệm pixel thiết bị có thể

xém xét như sau: khi ta quan sát màn hình (trong chế độ đồ họa), màn hình không liên tục mà gồm các điểm nhỏ, gọi là pixel. Mỗi pixel gồm một tập tọa độ (x, y) và màu.

Như vậy mỗi ảnh là tập hợp các điểm ảnh. Khi được số hóa nó thường được biểu diễn bởi mảng 2 chiều I(n,p): n là dòng và p là cột.

Về mặt toán học có thể xem ảnh là một hàm hai biến f(x,y) với x, y là các biến tọa độ. Giá trị số ở điểm (x,y) tương ứng với giá trị xám hoặc độ sáng của ảnh (x là các cột còn y là các hàng). Giá trị của hàm ảnh f(x,y) được hạn chế trong phạm vi của các số nguyên dương.

$$0 \leq f(x,y) \leq f_{max.}$$

Với ảnh đen trắng mức xám của ảnh có thể được biểu diễn bởi một số như sau:

$$f = k \int_{\lambda=0}^{\infty} c(\lambda) S_{BW}(\lambda) d\lambda$$

Trong đó $S_{BW}(\lambda)$ là đặc tính phổ của cảm biến được sử dụng và k là hệ số tỷ lệ xích. Vì sự cảm nhận độ sáng có tầm quan trọng hàng đầu đối với ảnh đen trắng nên $S_{BW}(\lambda)$ được chọn giống như là hiệu suất sáng tương đối. Vì *f* biểu diễn công suất trên đơn vị diện tích, nên nó bao giờ cũng không âm và hữu hạn.

$$0 \leq f \leq f_{max}$$

Trong đó f_{max} là giá trị lớn nhất mà *f* đạt được. Trong xử lý ảnh, *f* được chia thang sao cho nó nằm trong một phạm vi thuận lợi nào đó.

Thông thường đối với ảnh xám, giá trị f_{max} là 255 (2^8=256) bởi vì mỗi phần tử ảnh được mã hóa bởi một byte. Khi quan tâm đến ảnh màu ta có thể mô tả màu qua ba hàm số: thành phần màu đỏ qua *R(x,y)*, thành phần màu lục qua *G(x,y)* và thành phần màu lam qua *B(x,y)*. Bộ ba giá trị *R*, *G*, và *B* nhận được từ:

$$R = k \int_{\lambda=0}^{\infty} c(\lambda) S_R(\lambda) d\lambda$$

$$G = k \int_{\lambda=0}^{\infty} c(\lambda) S_G(\lambda) d\lambda$$

$$B = k \int_{\lambda=0}^{\infty} c(\lambda) S_B(\lambda) d\lambda$$

Ở đó $S_R(\lambda)$,$S_G(\lambda)$ *và* $S_B(\lambda)$ theo thứ tự là những đặc tính phổ của các cảm biến (bộ lọc) đỏ, lục và lam. *R, G, B* cũng không âm và hữu hạn.

Ảnh có thể được biểu diễn theo một trong hai mô hình: mô hình Vector hoặc mô hình Raster.

Mô hình Vector: Ngoài mục đích tiết kiệm không gian lưu trữ, dễ dàng hiển thị và in ấn, các ảnh biểu diễn theo mô hình vector còn có ưu điểm cho phép dễ dàng lựa chọn, sao chép, di chuyển, tìm kiếm…Theo những yêu cầu này thì kỹ thuật biểu diễn vector tỏ ra ưu việt hơn. Trong mô hình này, người ta sử dụng hướng vector của các điểm ảnh lân cận để mã hóa và tái tạo lại hình ảnh ban đầu. Các ảnh vector được thu nhận trực tiếp từ các thiết bị số hóa như Digitalize hoặc được chuyển đổi từ các ảnh Raster thông qua các chương trình vector hóa.

Mô hình Raster: là mô hình biểu diễn ảnh thông dụng nhất hiện nay. Ảnh được biểu diễn dưới dạng ma trận các điểm ảnh. Tùy theo nhu cầu thực tế mà mỗi điểm ảnh có thể được biểu diễn bởi một hay nhiều bit. Mô hình Raster thuận lợi cho việc thu nhận, hiển thị và in ấn. Các ảnh được sử dụng trong phạm vi của đề tài này cũng là các ảnh được biểu diễn theo mô hình Raster.

Khi xử lý các ảnh Raster chúng ta có thể quan tâm đến mối quan hệ trong vùng lân cận của các điểm ảnh. Các điểm ảnh có thể xếp hàng trên một lưới (raster) hình vuông, lưới hình lục giác hoặc theo một cách hoàn toàn ngẫu nhiên với nhau.

8 láng giềng 4 láng giềng

Hình 1.2 Quan hệ trong vùng lân cận giữa các điểm ảnh.

Cách sắp xếp theo hình vuông là được quan tâm đến nhiều nhất và có hai loại: điểm 4 láng giềng (4 liền kề) hoặc 8 láng giềng (8 liền kề). Với điểm 4 láng giềng, một điểm ảnh I(i, j) sẽ có điểm kế cận theo 2 hướng i và j; trong khi đó với điểm 8 láng giềng, điểm ảnh I(i, j) sẽ có 4 điểm kế cận theo 2 hướng i, j và 4 điểm kế cận theo hướng chéo 45^{o} (Xem hình 1.2)

1.4. Phạm vi ứng dụng của xử lý ảnh

Xử lý ảnh đã đem lại nhiều ứng dụng trong nhiều lĩnh vực khác nhau: y học, khoa học hình hình sự, khí tượng thuỷ văn, quản lý, ...

Quản lý là là một trong những ứng dụng quan trọng của xử lý ảnh. Cùng với sự bùng nổ của kinh tế thị trường. Khối lượng quản lý càng lớn, như quản lý hồ sơ, quản lý phiếu điều tra trong công tác thống kê, các câu hỏi trắc nghiệm. Để thực hiện các công việc trên một cách chính xác, nhanh chóng và hiệu quả. Xử lý ảnh và nhận dạng đã nghiên cứu và phát triển mạnh mẽ bài toán nhập liệu tự động.

1.5. Các loại tệp cơ bản trong xử lý ảnh

Ảnh thu được sau quá trình số hoá có nhiều loại khác nhau phụ thuộc vào kỹ thuật số hoá ảnh và các ảnh thu nhận được có thể lưu trữ trên tệp để dùng cho việc xử lý các bước tiếp theo. Sau đây là một số loại tệp cơ bản và thông dụng nhất hiện nay.

1.5.1. File ảnh IMG

Ảnh IMG là ảnh đen trắng, phần đầu file IMG có 16 bytes chứa các thông tin cần thiết:

+ 6 bytes đầu dùng để đánh dấu nhận dạng file IMG. Giá trị của 6 bytes đầu này viết dưới dạng hexa: 0x0001 0x0008 0x0001.

+ 2 bytes chứa độ dài các mẫu tin. Đó là độ dài của một dãy các bytes lặp lại một số lần nào đó, số lần lặp này sẽ được lưu trong một file đếm. Nhiều dãy giống nhau được lưu trong một bytes. Đó chính là cách lưu trữ nén

+ 4 bytes tiếp theo mô tả kích cỡ của pixel

+ 2 bytes tiếp mô tả số pixel trên một dòng

+2 bytes cuối cho biết số dòng trong ảnh

Các dòng giống nhau được nén thành một pack. Có 4 loại pack sau:

- Loại 1: Gói các dòng giống nhau.Quy cách gói tin này 0x00 0x00 0xFF Count. 3 bytes đầu cho biết số các dãy giống nhau ,bytes cuối cho biết số các dòng giống nhau.
- Loại 2: Gói các dãy giống nhau. Quy cách gói này 0x00 Count. Bytes thứ hai cho số các dãy giống nhau được nén trong gói. Độ dài cử dãy được ghi đầu file.
- Loại 3:Dãy các pixel không giống nhau, không lặp lại và không nén được. Quy cách như sau: 0x80 Count. Bytes thứ hai cho biết độ dài dãy các pixel không giống nhau, không nén được.
- Loại 4:Dãy các pixel giống nhau. Tuỳ theo các bit cao của bytes 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 bytes được nén được tính bởi 7 bit thấp còn lại. Nếu bit cao tắt (giá tri 0) thì đay là gói nén các bytes toàn bit 1. Số các bytes được nén được tính bởi 7 bit thấp còn lại.

Các cảu file IMG phong phú như vậy là do ảnh IMG là ảnh đen trắng nên chỉ cần 1 bit cho một pixel thay vì 4 hoặc 8 bit như đã nói ở trên toàn bộ ảnh chỉ có điểm sáng hoặc tối tương ứng với 1 hoặc 0. Tỷ lệ nén của file này là khá cao.

1.5.2 File ảnh PCX

Định dạng ảnh PCX là một trong những định dạng ảnh cổ điển nhất. Nó sử dụng phương pháp mã loạt dài RLE (Run-Length-Encoded) 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 liệuảnh (image data) và bảng màu mở rộng.

Header của tệp PCX có kích thước cố định gồm 128 byte và được phân bố như sau:

+ 1 byte : chỉ ra kiểu định dạng. Nếu là 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: toạ độ góc trái trên của ảnh. Với kiểu PCX nó có giá trị là (0,0); còn PCC thì khác (0,0).

+ 1 word: toạ độ góc phải dưới.

+ 1 word: kích thước bề rộng và bề 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.

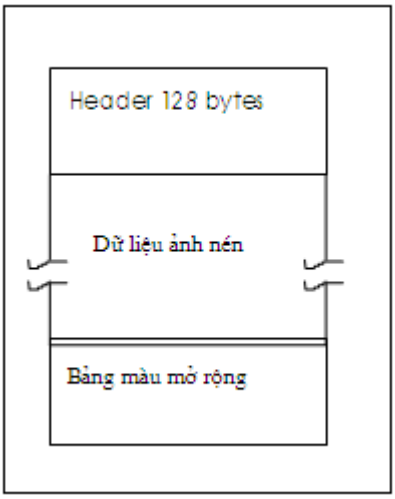
+1 byte: số bit plane mà ảnh sử dụng. Với ảnh 16 màu, giá trị này là 4, với ảnh 256 màu (1 pixel/8 bit) thì số bit plane lại là 1.

+ 1 byte: số bytes cho một dòng quét ảnh.

+ 1 word: kiểu bảng màu.

+ 58 byte: không dùng.

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 (Targed Image File Format) sẽ mô tả dưới đây.



Hình 1.3 Cấu trúc tệp ảnh dạng PCX.

1.5.2.1 Kỹ thuật nén ảnh PCX

a) Kiểu nén: Thông tin về giá trị điểm xám cho mỗi điểm ảnh PCX được lưu trữ theo kiểu nén, khi được lưu trữ theo kiểu nén các file phải tuân theo quy luật nhất định: là một ma trận hai chiều để lưu trữ thông tin liên quan về các giá trị mức xám. Kỹ thuật dùng để nén ảnh PCX là kỹ thuật Run Length Encode (RLE), phần tử thông tin cần nén là 1 bytes.

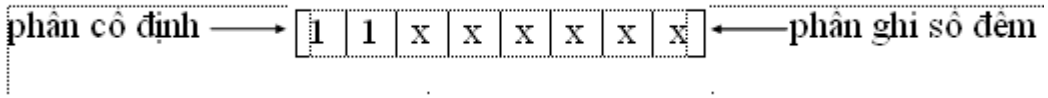
b) Tỷ số nén: Trong kỹ thuật nén ảnh người ta quan tâm nhiều đến tỷ số nén. Tỷ số nén của ảnh được tính bởi tỷ số giữa kích thước lưu trữ ảnh sau khi nén trên kích thước cần thiết để lưu trữ ảnh không nén. Giá trị của tỷ số này phụ thuộc vào mỗi file ảnh, ảnh pcx có thể là 1,4 hoặc 8 bits, nếu xét yếu tố này ảnh hưởng đến tỷ số nén ta thấy:

- Ảnh 1 bits (hay ảnh nhị phân) thì một bytes lưu trữ 8 bits khả năng xuất hiện mỗi mức xám là lớn (50% cho mỗi mức xám) làm cho tần xuất lặp bits là lớn, yếu tố này làm tăng khả năng nén. Nhưng phải ít nhất 3 bytes liên tiếp giống nhau trong một dòng quét thì mới có hiệu quả cho việc nén tức là tần xuất lặp ở đây không phải cho từng pixel mà là cả gói 8 pixel cùng lặp giống nhau, yếu tố này làm giảm khả năng nén. Vậy việc nén ảnh nhị phân chỉ có ý nghĩa đối với ảnh có nền, còn đối với một số ảnh nhị phân khác việc nén không có ý nghĩa có khi càn làm tăng thêm kích thuóc của ảnh.
- Ảnh 4 bits (hay 16 màu) tương ứng với 4 bits mã hoá một pixel, ảnh này có 2 pixel được chứa trong một bytes. Khả năng xuất hiện cho mỗi mức màu là 1/16. Yếu tố này làm giảm đi khả năng nén so với ảnh nhị phân. Cần có ít nhất 3 bytes liên tiếp giống nhau cùng trong một dòng quét thì mới có hiệu quả nén, tần số lặp pixel ở đây là lặp gói gồm hai pixel, yếu tố này làm tăng khả năng nén hơn so với ảnh nhị phân.
- Ảnh 8 bits (hay ảnh 256 màu) tương ứng với 8 bits hay 1 bytes mã hoá một pixel. Khả năng xuất hiện cho mỗi mức màu là 1/256, yếu tố này làm giảm khả năng nén so với ảnh nhị phân và ảnh 4 bits. chỉ cần ít nhất 3 bytes (hay 3 pixel) liên tiếp giống nhau mà cùng nằm trong một dòng quét thì có hiệu quả nén.

Như vậy đối với mỗi ảnh Pcx 1,4,8 bits màu thì mỗi loại đều có các yếu tăng hoặc giảm khả năng nén. nếu ảnh nào sử dụng nền hoặc chỉ dùng một số mức màu nhất định trong bảng màu thì có khả năng nén cao.

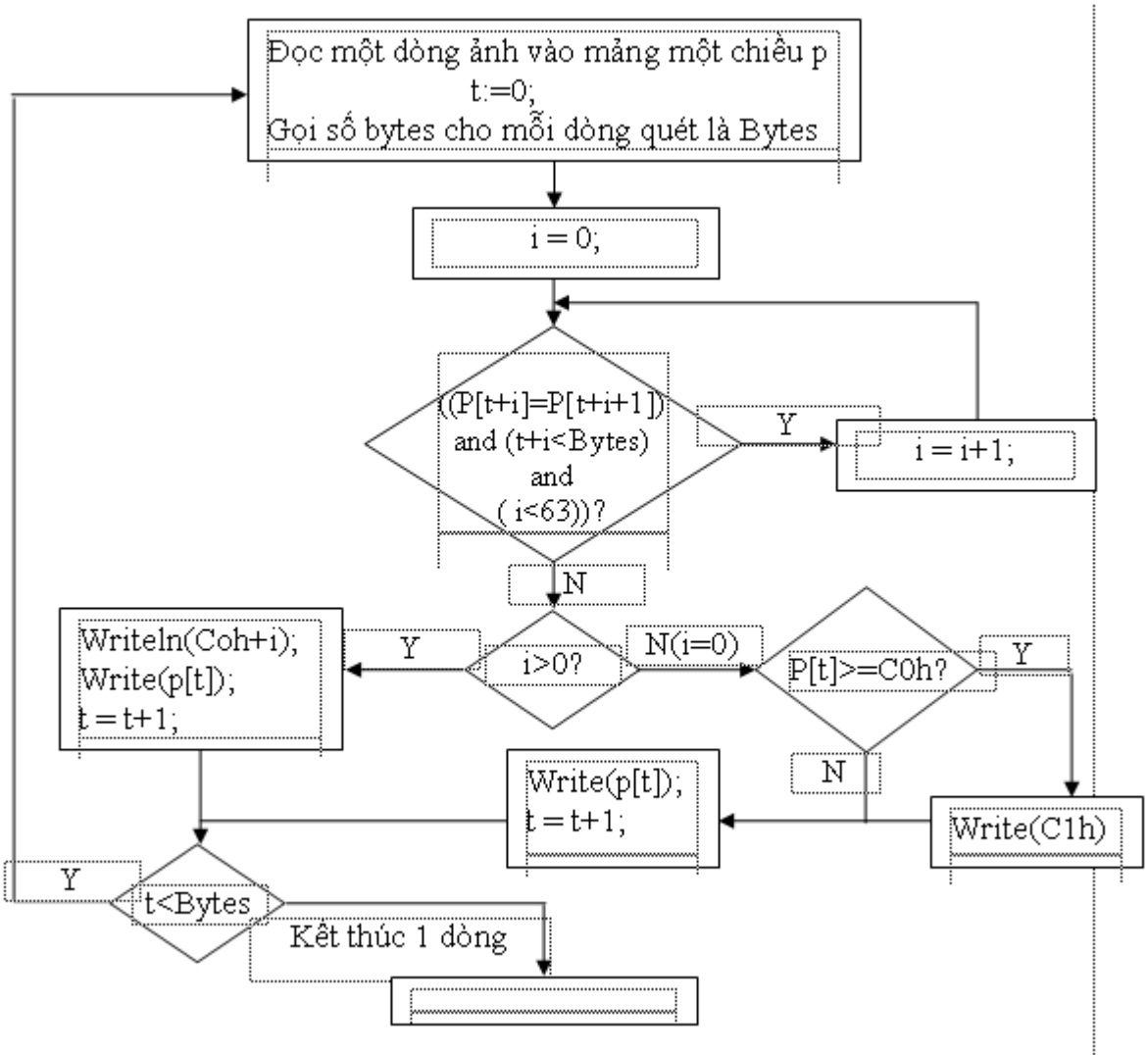
c) Dấu hiệu nén trong file trong ảnh PCX: Cấu trúc nén trong một dòng ảnh bao gồm hai bytes, bytes đầu là dấu hiệu nén và số bytes được nén, bytes tiếp theo chứa

chỉ số màu của các bytes đó. Bytes dùng làm dấu hiệu nén là một bytes đặc biệt nó được chia làm hai phần như hình vẽ sau:



Hình 1.4 Cấu trúc của bytes dấu hiệu

Phần cố định là C0h (1100 0000b), có 2 bits cao nhất là 1, số bits thấp hơn còn lại (gồm 6 bits) dùng để chỉ số bytes giống nhau liên tiếp. Như vậy mỗi cấu trúc chỉ có thể ghi được tối đa là 63 bytes giống nhau.



Hình 1.5 Sơ đồ giải thuật nén một dòng ảnh cho file PCX

1.5.2.2 Giải nén ảnh PCX

Quá trình nén được tiến hành theo từng dòng như sau:

+ Thứ tự đầu tiên trong file ảnh PCX là dòng đầu tiên của ảnh.

+ Việc nén file ảnh PCX phải bắt đầu từ dòng đầu tiên của ảnh.

+ Kết thúc khi tất cả các dòng đều được nén.

+ Mỗi một dòng nén phải tuân theo cùng một giải thuật nén của file PCX.

1.5.3 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 trường 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 trỏ 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 nhớ hơn.

1.5.4 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 đến7 lần. Với ảnh 16 màu, kích thước ảnh nhỏ hơn ảnh gốc 2-3 lần, có trường hợp có thể 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 loạt 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.

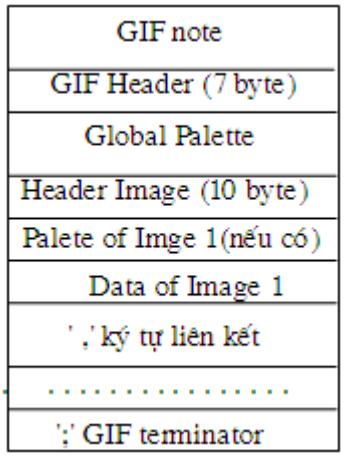
Định dạng ảnh GIF do hãng ComputServer Incorporated (Mỹ) đề xuất lần đầu tiên vào năm 1990. Với đing dạng GIF, những vướng mắc mà các định dạng khác gặp phải khi số màu trong ảnh tăng lên không còn nữa. Khi số màu càng tăng thì ưu thế của định dạng GIF càng nổi trội. Những ưu thế này có được là do GIF 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. Dạng ảnh GIF cho chất lượng cao, độ phân giải đồ hoạ cũng đạt cao, cho phép hiển thị trên hầu hết các phần cứng đồ hoạ.

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
  - Dấu phân cách
  - Bộ mô tả ảnh
  - Bản đồ màu cục bộ
  - Dữ liệu ảnh .

Phần mô tả này lặp n lần nếu ảnh chứa n đối tượng.

- Phần đầu cuối ảnh GIF(terminator)



- Chứ ký của ảnh GIF có giá trị là GIF87a. Nó gồm 6 ký tự, 3 kí 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;

+ Phần chưa dùng: 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 thự 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 lơ theo index 0
màu Red	4	giá trị màu đỏ theo index 1
màu Green	5	giá trị màu xanh lục theo index1
màu Blue	6	giá trị màu xanh lơ theo index 0
. . . . .	. . . . .	

- Bộ mô tả ảnh: định nghĩa vị trí thực tế và phần mở rộng của ảnh trong phạm vi không gian ảnh đã có trong phần mô tả hình hiển thị. Nếu ảnh biểu diễn theo ánh xạ bản đồ màu cục bộ thì cờ định nh\ghĩa 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 phảy). 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 thự byte	Mô tả
00101100	1	Ký tự liên kết ảnh (')
căn trái ảnh	2,3	Pixel bắt đầu ảnh tính từ trái hình hiển thị
căn đỉnh trên	4,5	Pixel cuối ảnh bắt đầu tính từ đỉnh trên hình hiển thị
độ rộng ảnh	6,7	chiều rộng ảnh tính theo pixel
độ cao ảnh	8,9	chiều cao ảnh tính theo pixel
MI000pixel	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 chiếu 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 hoạ (Graphic), còn JPEG dùng cho ảnh chụp (Photographic).

1.5.5 File ảnh BMP (BITMAP)

1.5.5.1. Khái niệm về ảnh đen trắng, ảnh màu, ảnh cấp xám.

- *Ảnh đen trắng.*

Đó là những bức ảnh mà mỗi điểm ảnh chỉ là những điểm đen hoặc trắng, được quy định bằng một bit. Nếu bit mang giá trị là 0 thì điểm ảnh là điểm đen, còn nếu mang giá trị là 1 thì điểm ảnh là điểm trắng. Do đó để biểu diễn một điểm ảnh đen trắng ta có thể dùng một ma trận nhị phân, là ma trận mà mỗi phần tử chỉ nhận một trong hai giá trị là 0 hoặc 1.

- *Ảnh màu*

Quá trình giấu tin vào ảnh màu cũng tương tự như với ảnh đen trắng nhưng trước hết ta phải chọn từ mỗi điểm ảnh ra bit có trọng số thấp nhất (LSB) để tạo thành một ảnh nhị phân gọi là ảnh thứ cấp. Sử dụng ảnh thứ cấp này như ảnh môi trường để giấu tin, sau khi biến đổi ảnh thứ cấp ta trả nó lại ảnh ban đầu để thu được ảnh kết quả.

- *Ảnh đa cấp xám*

Đối với ảnh đa cấp xám bảng màu của nó đã có sẵn, tức là những cặp màu trong bảng màu có chỉ số chênh lệch càng ít thì càng giống nhau. Vì vậy đối với ảnh đa cấp xám bit LSB của mỗi điểm ảnh là bit cuối cùng của mỗi điểm ảnh.

Quá trình tách bit LSB của ảnh đa cấp xám và thay đổi các bit này bằng thuật toán giấu tin trong ảnh đen trắng sẽ làm chỉ số của điểm màu bị thay đổi tăng hoặc giảm 1 đơn vị, do đó điểm ảnh mới sẽ có độ sáng tối của ô màu liền trước hoặc liền sau ô màu của điểm ảnh cũ. Bằng mắt thường rất khó có thể nhận thấy sự thay đổi về độ sáng tối này.

- *Ảnh nhỏ hơn hoặc bằng 8 màu*

Những ảnh thuộc loại này gồm có 16 màu (4 bit màu) và ảnh 256 màu (8 bit màu). Khác với ảnh màu, ảnh xám với số bit nhỏ hơn hoặc bằng 8 bit không phải luôn luôn được sắp xếp màu bảng màu.

Những màu ở liền kề nhau trong bảng màu có thể rất khác nhau chẳng hạn như màu đen với màu trắng vẫn có thể được xếp cạnh nhau.

Vì vậy việc xác định bit LSB của ảnh loại này rất khó. Nếu ta chỉ làm như đối với ảnh xám, tức là vẫn lấy bit cuối cùng của mỗi điểm ảnh để tạo thành ảnh thứ cấp thì mỗi thay đổi 0 -> 1 hoặc 1 ->0 trên ảnh thứ cấp có thể làm cho ảnh màu của điểm ảnh cũ và mới tương đương ứng thay đổi rất nhiều dù chỉ số màu của chúng cũng tăng hoặc giảm 1 mà thôi.

- *Ảnh hightcolor (16 bit màu)*

Ảnh 16 bit màu thực tế chỉ sử dụng 15 bit cho mỗi điểm ảnh trong đó 5 bit biểu diễn cường độ tương đối của màu đỏ, 5 bit biểu diễn cường độ tương đối của màu xanh lam, 5 bit biểu diễn cường độ tương đối của màu xanh lơ. Còn lại một bit không dùng đến là bit cao nhất của byte thứ hai trong mỗi cặp thứ hai byte biểu diễn một điểm ảnh, đó chính là bit LSB của ảnh 16 bit màu. Việc thay đổi giá trị của những bit này sẽ không hề ảnh hưởng tới màu sắc của từng điểm ảnh trong môi trường.

- *Ảnh true color (24 bit màu)*

Ảnh true color sử dụng 3 byte cho mỗi điểm ảnh, mỗi byte biểu diễn một thành phần trong cấu trúc RGB. Trong mỗi byte các bit cuối cùng của mỗi byte trong phần dữ liệu ảnh là các bit LSB của ảnh true color.

Để tăng lượng thông tin giấu được vào ảnh môi trường, từ mỗi byte của ảnh true color ra sẽ lấy nhiều hơn một bit để tạo thành ảnh thứ cấp. Thông thường cũng chỉ nên lấy nhiều nhất 4 bit cuối cùng của mỗi byte để ảnh kết quả không bị nhiễu đáng kể, khi đó lượng thông tin tối đa có thể giấu trong ảnh cũng tăng lên gấp bốn lần so với lượng thông tin tối đa giấu được trong ảnh đó nếu chỉ lấy 1 bit cuối cùng ở từng byte.

1.5.5.2. Cấu trúc ảnh BMP

Để thực hiện việc giấu tin trong ảnh, trước hết ta phải nghiên cứu cấu trúc của ảnh và có khả năng xử lý được ảnh tức là phải số hoá ảnh. Quá trình số hoá các dạng ảnh khác nhau và không như nhau. Có nhiều loại ảnh đã được chuẩn hoá như: JPEG, PCX, BMP… Sau đây là cấu trúc ảnh *.BMP.

Mỗi file ảnh BMP gồm 3 phần:

- ✓ BitmapHeader (54 byte)
- ✓ Palette màu (bảng màu)
- ✓ BitmapData (thông tin ảnh)

Cấu trúc cụ thể của ảnh:

- Palette màu (bảng màu): bảng màu của ảnh, chỉ những ảnh lớn hơn hoặc bằng 8 bit màu mới có Palette màu.

- BitmapData (thông tin ảnh): phần này nằm ngay sau phần palette màu của ảnh BMP. Đây là phần chứa giá trị màu của điểm ảnh trong ảnh BMP, các dòng ảnh được lưu từ dưới lên trên, các điểm ảnh được lưu từ trái sang phải. Giá trị của mỗi điểm ảnh là một chỉ số trỏ tới phần tử màu tương ứng của palette màu.

BitmapHeader (54 byte)

Byte	Đặt tên	Ý nghĩa	Giá trị
1 - 2	ID	Nhận dạng file	'BMP' hay 19778
3 – 6	File_Size	Kích thước File	Kiểu Long trong turbo C
7 – 10	Reserved	Dành riêng	Mang giá trị 0
11 – 14	OffsetBit	Byte bắt đầu vùng dữ liệu	Offset của byte bắt đầu vùng dữ liệu
15 -18	Isize	Số byte cho vùng info	40 byte
19 – 22	Width	Chiều rộng của ảnh BMP	Tính bằng pixel
23 – 26	Height	Chiều cao của ảnh BMP	Tính bằng pixel
27 – 28	Planes	Số planes màu	Cố định là 1
29 – 30	bitCount	Số bít cho một pixel	Có thể là 1,4,6,16,24
31-34	Compression	Kiểu nén dữ liệu	0: Không nén 1: Nén runlength 8bits/pixel 2: Nén runlength 4bits/pixel
35 -38	ImageSize	Kích thước ảnh	Tính bằng byte
39 – 42	XpelsPerMeter	Độ phân giải ngang	Tính bằng pixel/metr
43 – 46	YpelsPerMeter	Độ phân giải dọc	Tính bằng pixel/metr
47 – 50	ColorsUsed	Số màu sử dụng trong ảnh	
51 – 54	ColorsImportant	Số màu được sử dụng khi hiện ảnh	

- Thành phần BitCount của cấu trúc BitmapHeader cho biết số bit dành cho mỗi điểm ảnh và số lượng màu lớn nhất của ảnh. BitCount có thể nhận các giá trị sau:

1: Bitmap là ảnh đen trắng, mỗi bit biểu diễn 1 điểm ảnh. Nếu bit mang giá trị 0 thì điểm ảnh là đen, bit mang giá trị 1 điểm ảnh là điểm trắng.

4: Bitmap là ảnh 16 màu, mỗi điểm ảnh được biểu diễn bởi 4 bit.

8: Bitmap là ảnh 256 màu, mỗi điểm ảnh biểu diễn bởi 1 byte.

16: Bitmap là ảnh highcolor, mỗi dãy 2 byte liên tiếp trong bitmap biểu diễn cường độ tương đối của màu đỏ, xanh lá cây, xanh lơ của một điểm ảnh.

24: Bitmap là ảnh true color (2^{24} màu), mỗi dãy 3 byte liên tiếp trong bitmap biểu diễn cường độ tương đối của màu đỏ, xanh lá cây, xanh lơ (RGB) của một điểm ảnh.

- Thành phần ColorUsed của cấu trúc BitmapHeader xác định số lượng màu của palette màu thực sự được sử dụng để hiển thị bitmap. Nếu thành phần này được đặt là 0, bitmap sử dụng số màu lớn nhất tương ứng với giá trị của BitCount.

1.6. Cấu trúc ảnh PNG

Là một dạng hình ảnh sử dụng phương pháp nén dữ liệu mới – không làm mất đi dữ liệu gốc. PNG được tạo ra nhằm cải thiện và thay thế định dạng ảnh GIF với một định dạng hình ảnh không đòi hỏi phải có giấy phép sáng chế sử dụng. PNG được hỗ trợ bởi thư viện tham chiếu libpng, một thư viện nền độc lập bao gồm các hàm của C để quản lý các hình ảnh PNG.

Những tập tin PNG thường có phần mở rộng là PNG và đã được gán kiểu chuẩn MIME là image/png.

Một tập tin PNG bao gồm 8 – byte kí hiệu (89 50 4E 47 0D 0A 1A) được viết trong hệ thống có cơ số 16, chứa các chữ "PNG" và 2 dấu xuống dòng, ở giữa là xếp theo số lượng của các thành phần, mỗi thành phần đều chứa thông tin về hình ảnh. Cấu trúc dựa trên các thành phần được thiết kế cho phép định dạng PNG có thể tương thích với các phiên bản cũ khi sử dụng. Các "thành phần" trong tập tin.

PNG là cấu trúc như một chuỗi các thành phần, mỗi thành phần chứa kích thước, kiểu, dữ liệu, và mã sửa lỗi CRC ngay trong nó.

Chuỗi được gán tên bằng 4 chữ cái phân biệt chữ hoa chữ thường. Sự phân biệt này giúp bộ giải mã phát hiện bản chất của chuỗi khi nó không nhận dạng được.

Với chữ cái đầu, viết hoa thể hiện chuỗi này là thiết yếu, nếu không thì ít cần thiết hơn ancillary. Chuỗi thiết yếu chứa thông tin cần thiết để đọc được tệp và nếu bộ giải mã không nhận dạng được chuỗi thiết yếu,việc đọc tệp phải được hủy.

Về cơ bản, định dạng PNG đem lại cho ta những ưu thế vượt trội hơn so với các định dạng phổ thông khác hiện nay như JPG, GIF, BMP…Những ưu thế tỏ rõ sức mạnh hơn khi được sử dụng trong môi trường đồ họa web.

✓ Giảm thiểu dung lượng: Trong tất cả các định dạng ảnh phổ thông hiện nay thì hình ảnh PNG có thể coi là dung lượng nhỏ nhất. Điều này rất quan trọng khi sử dụng PNG trong môi trường web.

✓ Độ sâu của màu: Ảnh PNG hỗ trợ đến true color 48bit màu. Trong khi đó ảnh gif chỉ ở mức 256 màu.

1.7 Sự cần thiết phát hiện độ dịch chuyển của phiếu điều tra so với phiếu mẫu.

Trong các cuộc khảo sát thị trường, điều tra xã hội, các kỳ thi trắc nghiệm... trên giấy, việc xử lý các phiếu kết quả và hoàn chỉnh báo cáo thống kê là những công việc tiêu tốn khá nhiều thời gian và nhân công. Làm thế nào để bớt được gánh nặng này, đồng thời hạn chế sự can thiệp của con người để đảm bảo được độ chính xác, tính khách quan và nhanh chóng của việc điều tra hay chấm điểm? Chính vì vậy các phần mềm nhập và xử lý dữ liệu tự động sẽ giúp con người có thể xử lý nhanh, chính xác và đỡ tốn thời gian .Các phiếu điều tra, bài thi... chứa các ô đánh dấu trong hình chữ nhật, hình tròn hoặc hình e-lip... được quét bằng máy quét (scanner) và lưu dưới dạng file ảnh (ở hầu hết các định dạng thông thường như TIF, GIF, PCX, BMP, JPG...) nhiều trang, tương ứng mỗi trang là một phiếu. Ảnh được nhận dạng và xử lý, kết quả xử lý được thể hiện dưới dạng CSDL như DBF (Foxpro), XLS (Excel), MDB (Microsoft Access), TXT (dạng text file)... nhưng trong thực tế việc scan các phiếu điều tra thường xảy các sai sót như ảnh bị nhiễu, bị nghiêng một góc nào đó, hay ảnh bị dich chuyển Bản thân việc in phiếu, giấy in, máy photocopy, máy quét... đều ẩn chứa các nguyên nhân kỹ thuật khiến ảnh thu được từ các phiếu khác nhau có độ lệch (ví dụ: nghiêng), độ dịch chuyển (dịch lên hoặc dịch xuống) khác nhau mà ta cần loại bỏ.

Để loại bỏ những khó khăn này thì việc dịch chuyển ảnh đã scan cho chuẩn với ảnh mẫu là rất cần thiết. Nó giúp tăng độ chuẩn xác khi chấm các bài thi chắc nghiệm hoặc trong các phiếu điều tra.

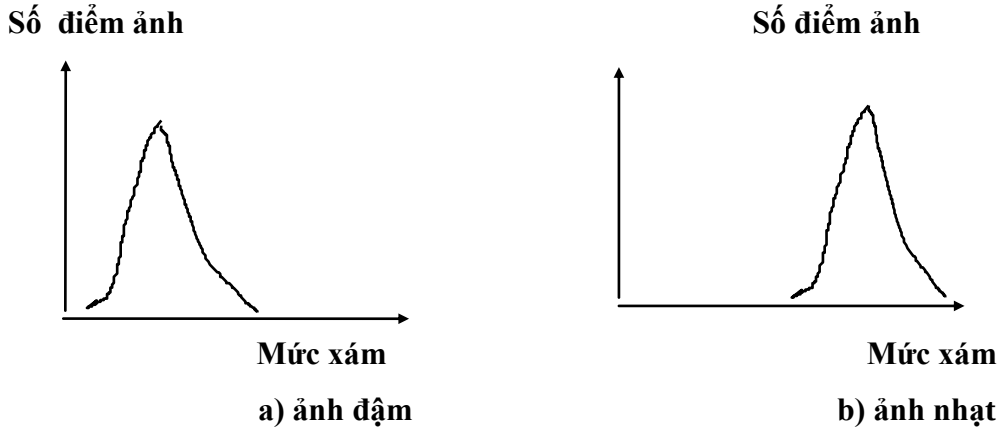
CHƯƠNG II
CÁC KỸ THUẬT PHÁT HIỆN ĐỘ DỊCH CHUYỂN PHIẾU ĐIỀU TRA VÀ BÀI TOÁN ỨNG DỤNG

2.1 Các định nghĩa cơ bản về Histogram

2.1.1 Định nghĩa histogram là gì?

Lược đồ mức xám (histogram) 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.



Hình 2.1 Lược đồ xám của ảnh

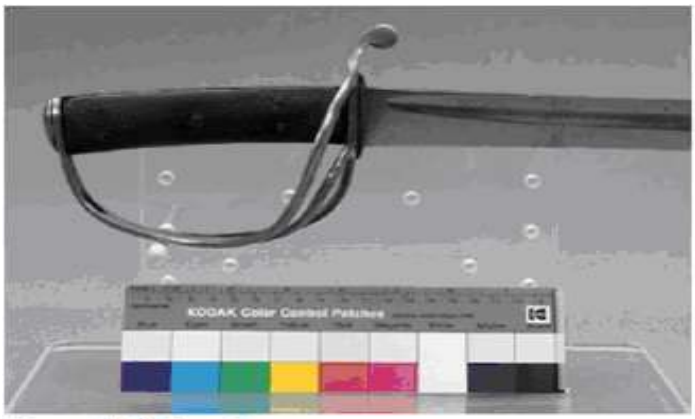


Figure 1. Object Image

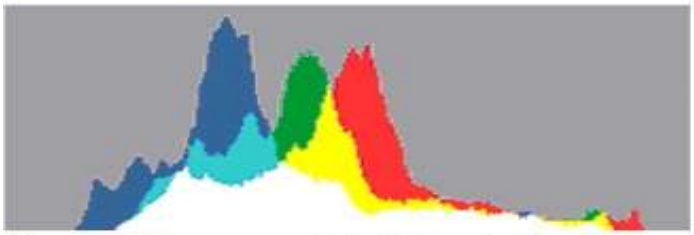
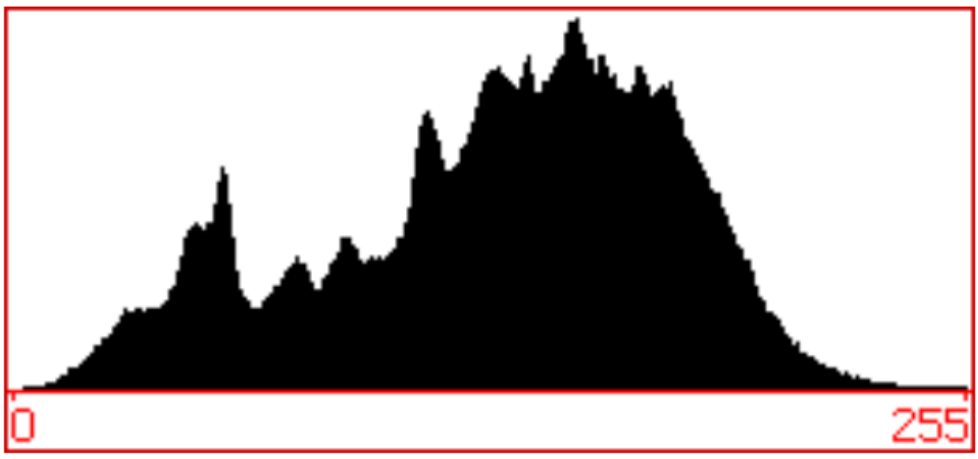


Figure 2. Histogram of the Object in Figure 1

Hình 2.2: Một ví dụ về biểu đồ tần suất histogram

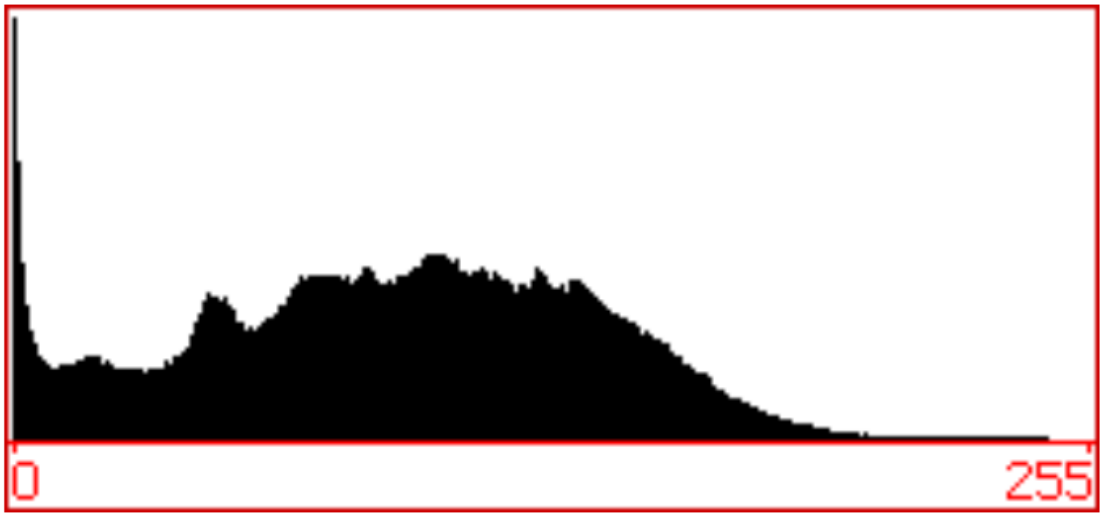
Histogram cung cấp cho những thông cơ bản, như độ sáng và độ tương phản (contrast) của ảnh. Độ tương phản đặc trưng cho sự thay đổi độ sáng của đối tượng so với nền. Có thể nói, độ tương phản là độ nổi của điểm ảnh hay vùng ảnh so với nền. Ta có một vài nhận xét về histogram:

+ NX1. Histogram tốt có hình ngọn núi với độ cao tăng dần từ trái, cao nhất ở giữa và thấp nhất ở bên phải. Điều đó chứng tỏ số lượng điểm ảnh nhiều nhất là ở độ sáng trung bình. (Xem Hình 2.3).



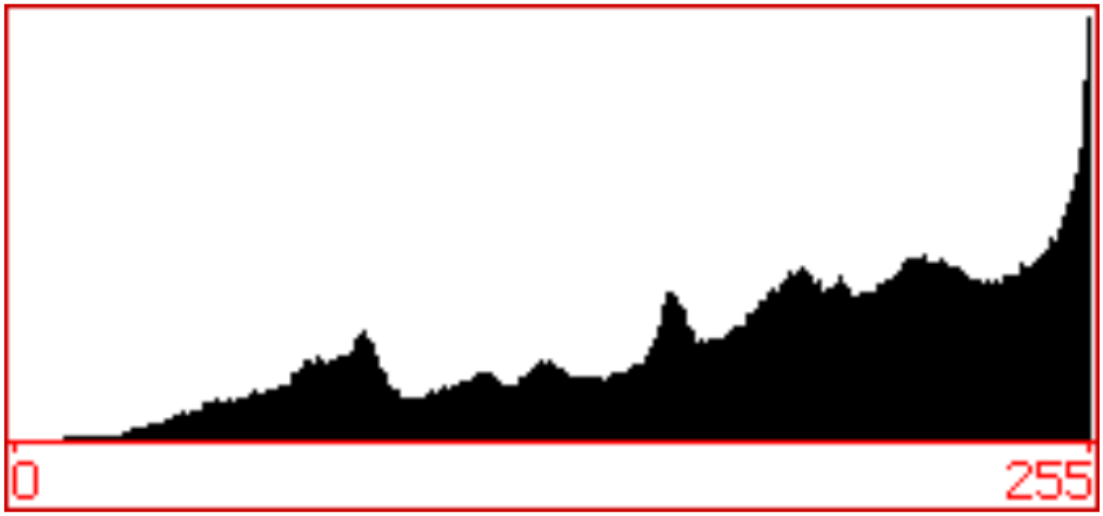
Hình 2.3: Histogram tốt

+ NX2. Ảnh quá tối: histogram bị nghiêng về bên trái, có một cái cột gần như thẳng đứng sát trái (Xem Hình 2.4).



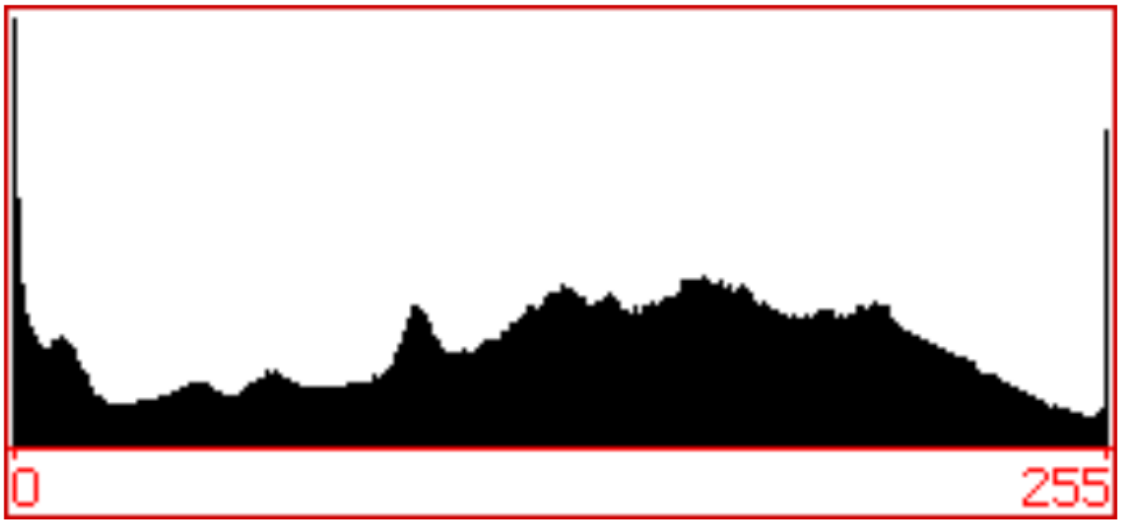
Hình 2.4: Histogram của ảnh quá tối

+ NX3. Ảnh quá sáng: histogram bị nghiêng về bên phải, có một cái cột gần như thẳng đứng sát phải (Xem Hình 2.5).



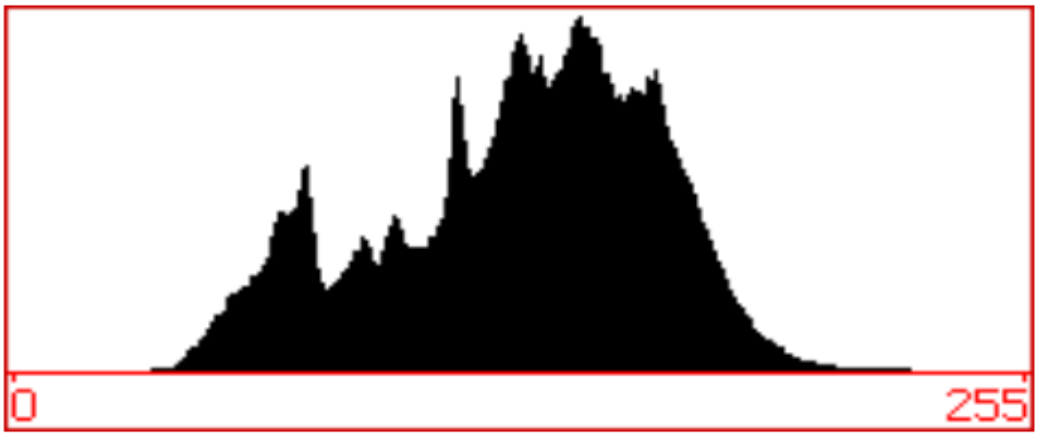
Hình 2.5: Histogram của ảnh quá sáng

+ NX4. Ảnh quá tương phản: có hai cái cột nằm ở 2 đầu trái phải (Xem Hình 2.6)



Hình 2.6: Histogram của ảnh quá tương phản

+ NX5. Ảnh kém tương phản: dải màu bị dồn vào giữa, hai đầu không có gì. (Xem Hình 2.7)



Hình 2.7: Histogram của ảnh kém tương phản

Từ lược đồ xám ta có thể suy diễn ra các tính chất quan trọng của ảnh như giá trị xám trung bình hoặc độ tản mạn. Qua cách tác động lên điểm ảnh, sự phân bố của biểu đồ cột được thay đổi theo mục đích. Dựa vào lược đồ xám chúng ta có thể xác định được ngưỡng thích hợp cho quá trình phân đoạn hoặc tính được các đại lượng đặc trưng của một ảnh.

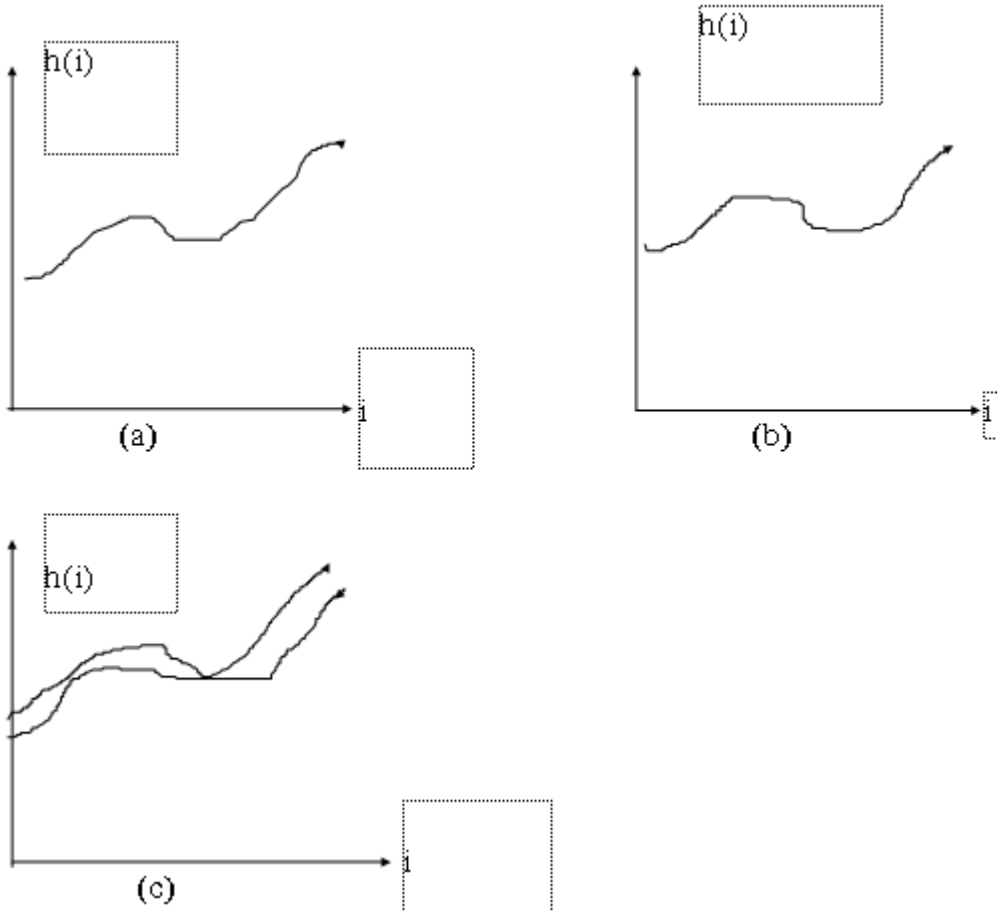
2.2 Các kỹ thuật phát hiện độ dịch chuyển văn bản

2.2.1 Kỹ thuật so sánh theo histogram

Việc đánh giá độ dịch chuyển của văn bản so với văn bằn mẫu sẽ được tiến hành thông qua việc xây dựng Histogram ngang và dọc của 2 văn bản. Đây cũng là một hướng tiếp cận dựa trên kỹ thuật đo độ tương tự, xét vị trí tương đối giữa các vùng thay đổi. Độ dịch chuyển của văn bản so với mẫu sẽ được đánh giá dựa trên sự tương đồng của Histogram văn bản so với Histogram của văn bản mẫu tương ứng.

Phương pháp này được trình bày như sau:

Giả sử Histogram dọc của ảnh mẫu và ảnh cần cần nhận dạng như sau:



Hình 2.8 Mô hình Histogram dọc của ảnh mẫu và ảnh cần nhận dạng

(a) ảnh mẫu, (b) ảnh cần nhận dạng, (c) histogram của ảnh mẫu và ảnh cần nhận dạng được vẽ chông lên nhau.

Đầu tiên ta vẽ mô hình Histogram dọc của văn bản mẫu và văn bản cần nhận dạng, sau đó ta chồng 2 Histogram của 2 văn bản lên cùng 1 trục tọa độ. Chúng ta nhận thấy nếu 2 histogram của 2 văn bản trùng nhau thì ảnh mẫu và ảnh cần nhận dạng không có sự sai lệch, nhưng ngược lại nếu ta thấy 2 Histogram của 2 văn bản mà lệch nhau thì văn bản mẫu và văn bản cần nhận dạng đã có sự dịch chuyển trong quá trình quét ảnh.

2.2.2 Phương pháp đánh giá độ dịch chuyển cấu trúc văn bản *theo* mẫu

2.2.2.1 Quan hệ Q_θ

+Định nghĩa : [Liên kết Q_θ]

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

(i) $U \equiv X_1$

(ii) $V \equiv X_n$

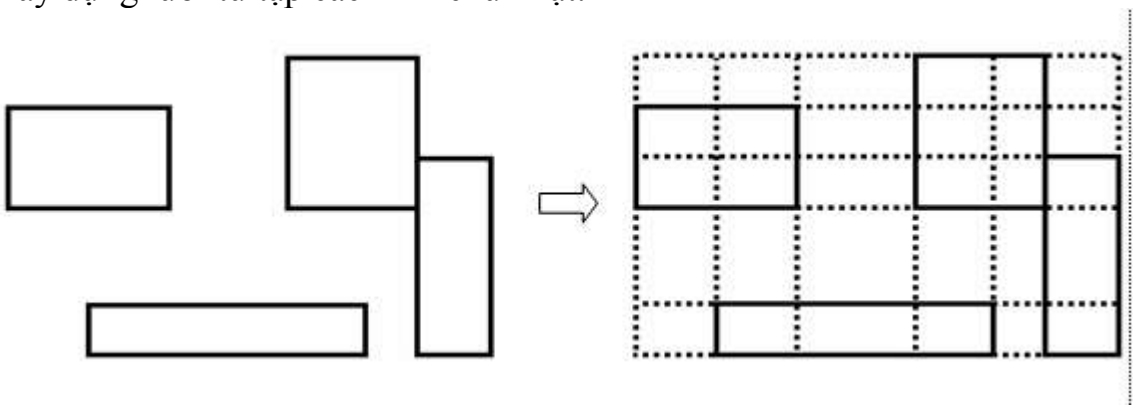
(iii) $h(X_i, X_{i+1}) < \theta \ \forall i, 1 \le i \le n-1$

+Quan hệ liên kết Q_θ là một quan hệ tương đương.

2.2.2.2 Đánh giá độ dịch chuyển của văn bản

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

Việc xây dựng lưới tựa các vùng hình chữ nhật tìm được trong văn bản thông qua việc chọn ngưỡng θ dựa vào biểu đồ tần xuất hay các vùng văn bản chữ nhật trong mẫu. Lưới là tập các toạ độ ngang dọc, hình 2.9 thể hiện ví dụ minh hoạ việc xây dựng lưới từ tập các hình chữ nhật.



Hình 2.9: Xây dựng lưới tựa các hình chữ nhật

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

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

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

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

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

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

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

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

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

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

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

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

Trong đó:

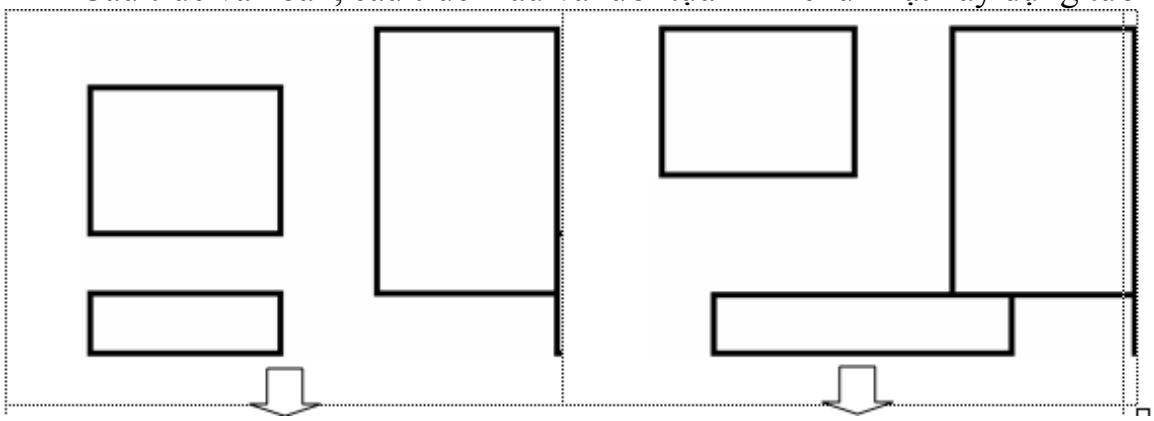
MG_{Grid}- là lưới kết hợp từ hai lưới được xây dựng từ các hình chữ nhật vùng của văn bản và mẫu

$n_c, n_c^{'}$ - là số vùng của văn bản và số vùng của mẫu

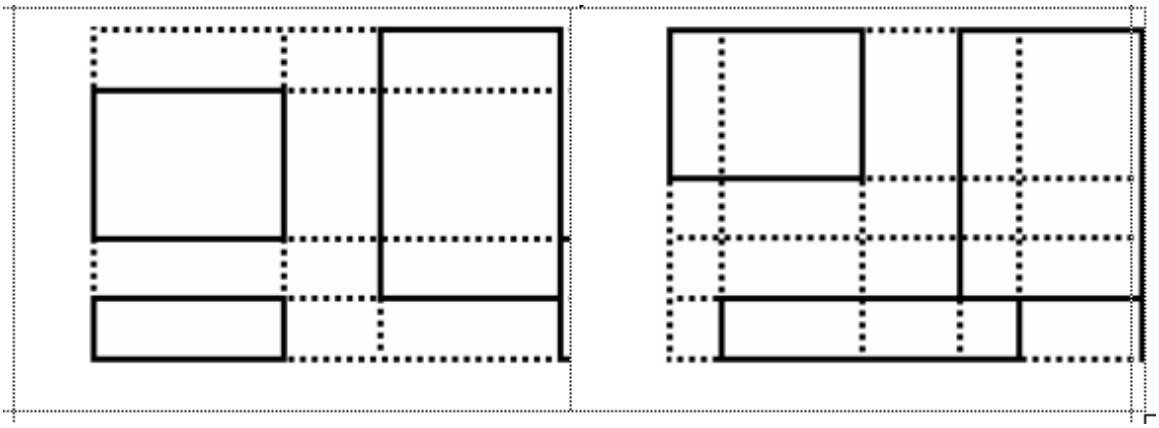
$N_{MG_{Grid}}(i,j), N_{MG_{Grid}}^{'}$ - là độ dịch chuyển của văn bản và mẫu so với ô lưới (i,j)

***Ví dụ minh họa đánh giá độ dịch chuyển văn bản so với mẫu.**

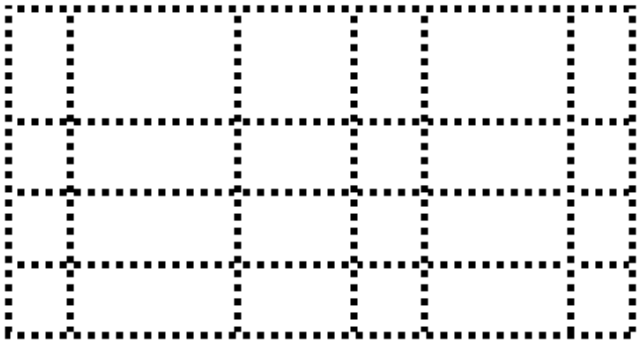
Cấu trúc văn bản, cấu trúc mẫu và lưới tựa hình chữ nhật xây dựng tương ứng



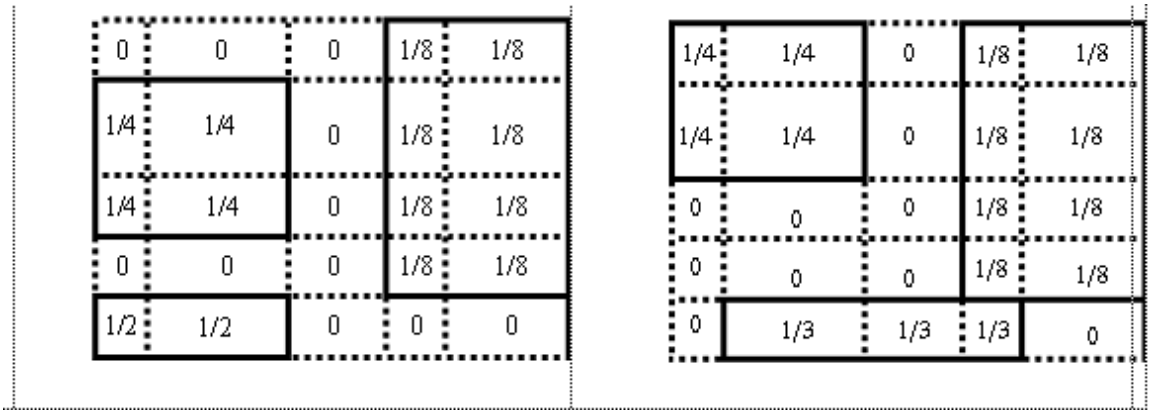
Lưới tựa hình chữ nhật tương ứng



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



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



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

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

2.2.3 Phát hiện độ dịch chuyển của ảnh mẫu so với ảnh cần nhận dạng dựa theo hướng tiếp cận trừ điểm ảnh..

Giả sử chúng ta có ảnh dạng RGB 24 bit, và được chuyển sang ảnh 256 cấp xám . Chúng ta gọi ảnh hiện thời là Ic, ảnh cần nhận dạng là Ip. Ảnh Igc, Igp cùng là ảnh xám 256 màu, được chuyển như sau:

//Ảnh ban đầu

$ColorI_c=(I_c(i,j).Red + I_c(i,j).Green + I_c(i,j).Blue)/3;$

$Ig_c(i,j).Blue= ColorI_c;$

$Ig_c(i,j).Green= ColorI_c;$

$Ig_c(i,j).Red= ColorI_c;$

//Ảnh cần nhận dạng

$ColorI_p=(I_c(i,j).Red + I_c(i,j).Green + I_c(i,j).Blue)/3;$

$Ig_p(i,j).Red= ColorI_p;$

$Ig_p(i,j).Green= ColorI_p;$

$Ig_p(i,j).Blue= ColorI_p;$

Tiếp theo, Igc, Igp được trừ theo từng điểm ảnh, và được so sánh với ngưỡng. Nếu giá trị tuyệt đối nhỏ hơn giá trị ngưỡng thì coi là điểm giống nhau, ngược lại coi là khác nhau. Tức là, tại vị trí i,j:

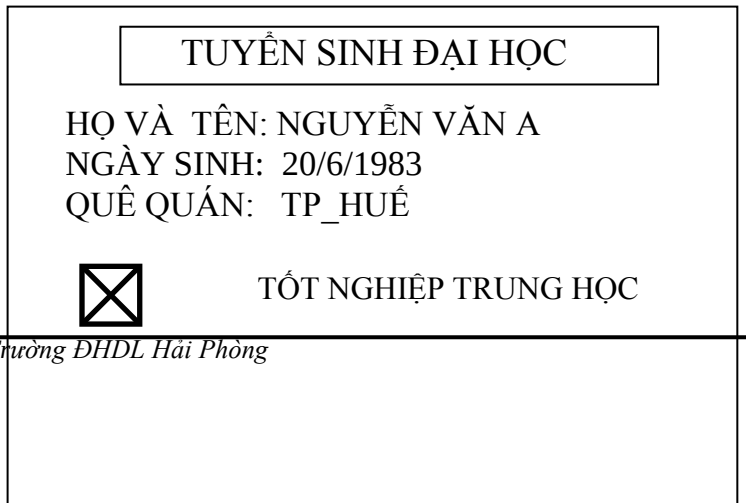
```
if(abs(ColorI_p- ColorI_c)<lThreshold)   //giống nhau
{
      Iwb(i,j).Red=0;
      Iwb(i,j).Green=0;
      Iwb(i,j).Blue=0;
} else   //khác nhau
{
      Iwb(i,j).Red=255;
      Iwb(i,j).Green=255;
      Iwb(i,j).Blue=255;
}
```

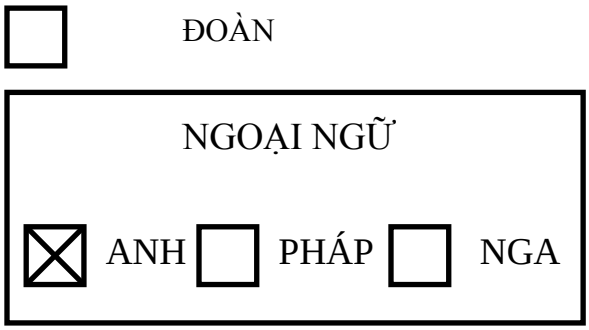
Iwb là ảnh đen trắng thể hiện vùng khác nhau giữa 2 ảnh, những điểm khác nhau sẽ có màu trắng, ngược lại có màu đen.

2.3 Phát biểu và phân tích bài toán ứng dụng, lựa chọn giải pháp sử lý

2.3.1 Phát biểu bài toán và phân tích bài toán

Bây giờ chúng ta tiến hành phân tích một bài toán cụ thể. Bài toán nhận dạng , hiệu chỉnh độ dịch chuyển các phiếu tuyển sinh đại học đơn giản sau:





Hình 2.10 Một phiếu tuyển sinh đơn giản.

Để nhận dạng phiếu điều tra mới trước tiên ta phải học mẫu. Đối với bài toán đơn giản này để nhận biết một học sinh nào đó có các thuộc tính như đã vào đoàn, hay ngoại ngữ thuộc Anh, Pháp...,Có thể tiến hành các bước sau:

+Đối với mỗi ô☐ ta lưu các toạ độ trái trên và phải dưới vào một file văn bản file1.DBF.

Đối văn bản trên thì file1.DBF sẽ có nội dung như sau:

rect1(top,left,down,bottom)

rect2(top,left,down,bottom)

rect3(top,left,down,bottom)

rect4(top,left,down,bottom)

rect5(top,left,down,bottom)

Sau đó tiến hành phân tích tệp file1.DBF, cứ mỗi ô recti(top,left,down,bottom) ta tính diện tích của ô đó. Diện tích ở đây là số pixel có trên đường biên và bên trong nó. Sau đó lưu chúng vào file2.DBF có nội dung như sau:

rect1(top,left,down,bottom,area)

rect2(top,left,down,bottom,area)

rect3(top,left,down,bottom,area)

rect4(top,left,down,bottom,area)

rect5(top,left,down,bottom,area)

Với mỗi phiếu mới bất kỳ (được lưu dưới dạng file ảnh), ta đọc và lấy nó ra có cấu trúc như file2.DBF và lưu nó vào trong file KQ.DBF.

Tiếp theo để xem một ô nào đó dược đánh dấu hay không, ta dùng phương pháp so sánh diện tích. Nếu có đánh dấu thì diện tích của ô trong bức ảnh cần nhận dạng sẽ lớn hơn ô tương ứng trong bức ảnh mẫu một ngưỡng θ (θ đủ lớn).

$|area_i(KQ)-area_i(file2)| <= \theta$ không đánh dấu, trong trường hợp ngược lại thì đánh dấu.

Tuy nhiên trong thực tế khi bức ảnh được quét vào do nhiều yếu tố khách quan nên bức ảnh cần nhận dạng bị lệch so với ảnh mẫu, khoảng cách từ lề trên và trái đến bức ảnh cũng bị lệch đi so với ảnh mẫu.

Điều này làm cho quá trình nhận dạng bị sai. Để khắc phục nhược điểm này phải điều chỉnh ảnh cần nhận dạng sao cho có cấu trúc giống với ảnh mẫu.Trong quá trình nhận dạng(Chẳng hạn đối với bài toán tuyển sinh) bước đầu tiên cần phải tách được toạ độ của các ô vuông nằm lẫn lộn trong văn bản. Đây là những khâu quan trọng trong nhập liệu tự động, Nó quyết định sự chính xác của bài toán nhập liệu tự động.

2.3.2 Phương pháp xử lý

2.3.2.1 Hiệu chỉnh độ dịch chuyển của văn bản so với văn bản gốc theo Histogram

Trong bài toán nhập liệu tự động việc hiệu chỉnh độ dịch chuyển của ảnh cần nhận dạng so với ảnh gốc là một bước quan trọng có ảnh hưởng đến kết quả quá trình nhận dạng. Để hiệu chỉnh độ dịch chuyển này thông thường dùng phương pháp Histogram.

Histogram theo chiều ngang hay dọc của một bức ảnh là tổng số các pixel đen trên một hàng ngang hay dọc của bức ảnh.Vậy ta có histogram ngang và histogram dọc của một bức ảnh. Đối với một dòng ảnh mà histogram ngang bằng 0 thì đó là dòng trắng (dòng gồm các điểm không thuộc ký tự).

$H(I)=\#\{(I,Y),I(I,Y)=0\}$.

Đây chính là histogram ngang của dòng i.

Giả sử chúng ta làm việc với ảnh đen trắng (ảnh 1 bits).Gọi Buf là mảng chứa bộ đệm ảnh

- Thủ tục Getpl() dùng để trả lại giá trị của một pixel tại vị trí x, trong hàng ảnh y.

```
int  Getpl(int x,int y,usnigend char * Buf[])
     {
       if (Buf[y][x] & (0x01<<(7-x%8)
           return 1;
       else
          return 0;
  }
```

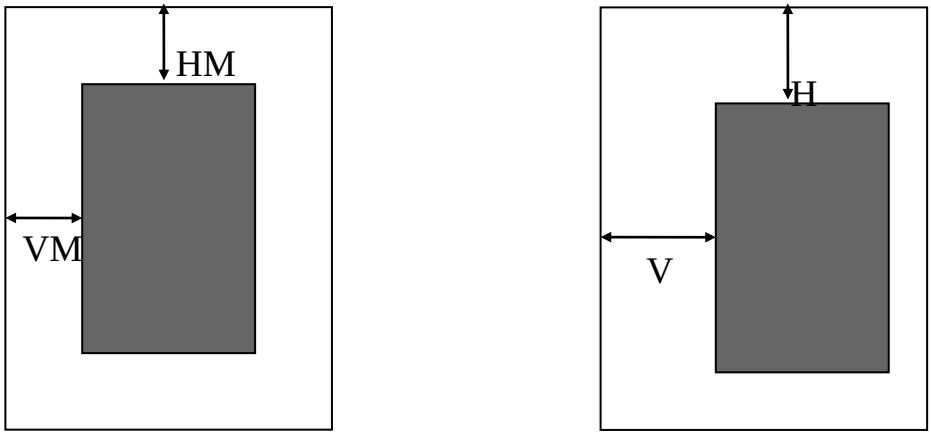
- Thủ tục His_H() dùng để tính lược đồ xám ngang (Histogram ngang) bắt đầu từ cột Sart_x đến cột End_x.

```
Int His_H(int y,int Sart_x,int End_x,unsigned Char * Buf[] )
 {
   int h,j;
   h=0;
   for(j=Sart_x;j<=End_x;j++)
       h+=Getpl(j,y,*Buf);
  return  h;
 }
```

Thủ tục His_V() dùng để tính lược đồ xám dọc tại cột x bắt đầu từ Sart_y đến End_y.

```
Int His_V( int x, int Sart_y,int End_y,Unsigned  char *Buf[ ] )
 {
   int v,i;
   for(i=Sart_y,i<= End_y;i++)
     v+=Getpl(x,i,*Buf);
   return v;
 }
```

Để hiệu chỉnh offset (lề trên và trái) của bức ảnh cần nhận dạng so với ảnh mẫu.



(a) (b)

Hình 2.11 (a) là ảnh mẫu (b) là ảnh cần nhận dạng.

Đồ án đưa ra hai phương pháp sau:

+ Phương pháp thứ nhất:

Trước tiên tìm khoảng cách hm,vm của ảnh mẫu (lề trên và lề trái). Để tìm được khoảng cách này ta lần lượt tính $H(i_0)$ và $V(j_0)$ từ trên xuống dưới và từ trái qua phải tại dòng i và cột j đầu tiên mà H(i) >θ ,V(j)> θ (θ đủ lớn) thì dừng lúc đó $i=i_0$ và $j=j_0$ chính là hm và vm. Bước tiếp theo cũng thực hiện tương tự đối với ảnh cần nhận dạng ta tìm được h và v tương ứng.

Sau đó ta so sánh sự chênh lệch giữa hai cặp hm và h,vm và v để tịnh tiến những dòng đen của ảnh lên hay xuống một khoảng cách (được tính theo đơn vị pixel) |hm-h|.Và tịnh tiến các cột đen của ảnh sang trái hay phải một khoảng |vm-v|.

Phương pháp này có ưu điểm là thời gian thực hiện khá nhanh tuy nhiên nó thực sự chính xác khi ảnh mẫu và ảnh cần nhận dạng phải rơi vào trường hợp khá lý

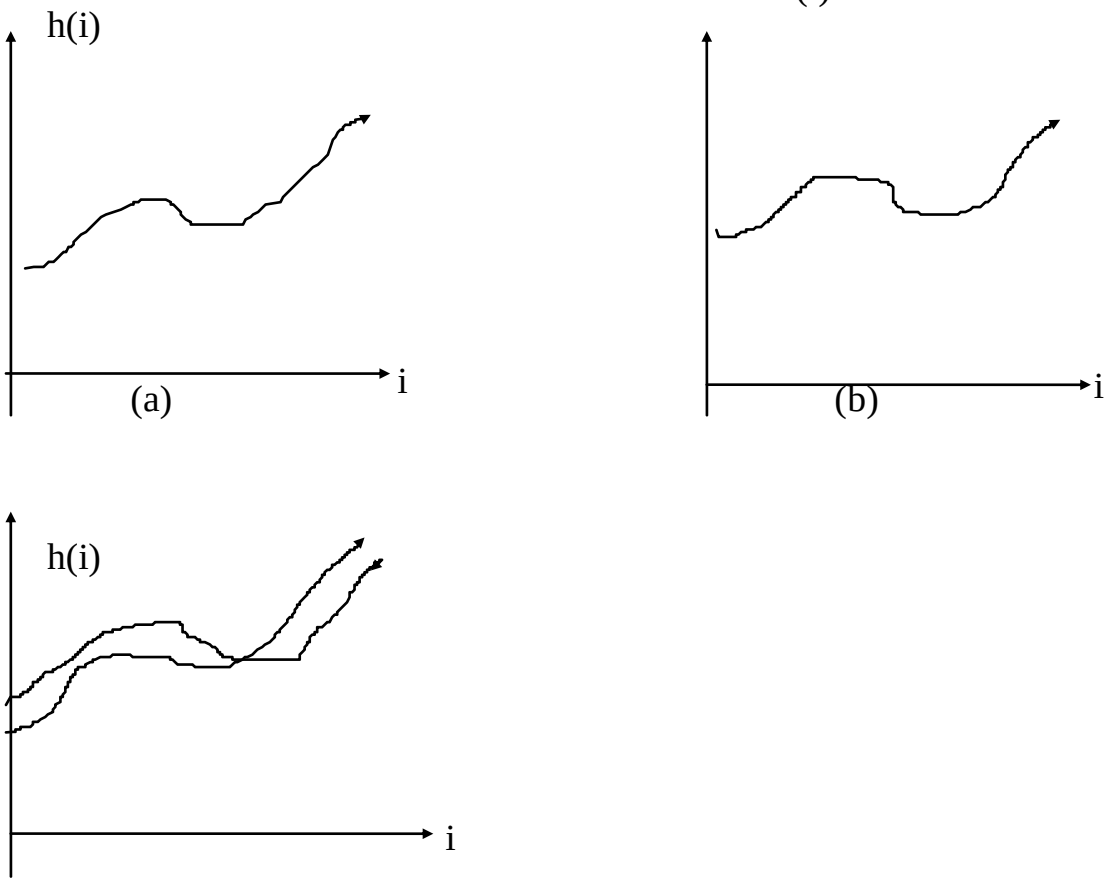
tưởng là ảnh mẫu và ảnh cần nhận dạng không có nhiễu. Trong trường hợp ngược lại kết quả thường không được như ý muốn.

Trong thực tế đôi khi ảnh mẫu và ảnh cần nhận dạng thường bị nhiễu khi quét vào, và có những trường hợp ảnh mẫu không bị nhiễu nhưng ảnh cần nhận dạng lại bị nhiễu hay trong trường hợp ngược lại.

Để khắc phục nhược điểm trên ta tiếp tục nghiêng cứu phương pháp thứ hai.

+ Phương pháp thứ hai:

Giả sử Histogram dọc của ảnh mẫu và ảnh cần nhận dạng như sau:



(c)

Hình 2.12 Mô hình Histogram của ảnh mẫu và ảnh cần nhận dạng.(a) ảnh mẫu ,(b) ảnh cần nhận dạng ,(c) Histogram của ảnh mẫu và ảnh cần nhận dạng được vẽ chồng lên nhau.

Ta tìm vị trí k ở ảnh mẫu và vị trí l ở ảnh cần nhận dạng sau cho:

$$\sum_{t=1}^{Hmax} (h1(k+t)-h2(l+t))^2 \rightarrow \min \quad (1)$$

Trong đó Hmax là một ước lượng đủ lớn. h1(i) là histogram dọc của ảnh mẫu h2(i) là histogram của ảnh cần nhận dạng. Thông thường ta cố định một đối số và tìm đối số còn lại. Chẳng hạn ta cố định k=0,và tìm vị trí theo công thức (1).Tại vị trí l chính là cột đầu tiên của bức ảnh sau khi điều chỉnh lề phía trái.

Tương tự để hiệu chỉnh lề trên của ảnh ta cũng tiến hành các bước như hiệu chỉnh lề trái nhưng thay vì sử dụng histogram ngang ta lại sử dụng histogram dọc.

2.4 Bước đầu cài đặt bài toán và nhận dạng phiếu điều tra.

Sau khi đã tiến hành nghiên cứu các thuật toán trợ giúp cho nhập liệu tự động, mặc dù để tiến hành một bài toán nhập liệu tự động cần phải có các bước tiền xử lý phức tạp khác như khữ nhiễu, làm trơn biên, tăng độ tương phản...Tuy nhiên việc tiến hành nghiên cứu các bước tiền xử lý đó đòi hỏi một lượng thời gian không nhỏ,Và để làm quen với bài toán nhập liệu tự động luận văn đã cài một bài toán nhận dạng đơn giản(nhận dạng các phiếu điều tra hay trắc nghiệm...). Bài toán chỉ dừng lại ở mức tham khảo tham khảo và có ý nghĩa động viên cho việc cài đặt một bài hệ nhập liệu tự động sau này, chứ độ chính xác thì chưa cao (do bỏ qua các khâu tiền xử lý). Để cài bài toán luận văn sử dụng thuật toán đối sánh diện tích. Bài toán này có hai bước. Bước thứ nhất là học form ảnh mẫu và bước thứ hai là nhận dạng.

2.4.1 Học form ảnh mẫu

Học form ở đây (đối với bài toán nhận dạng các phiếu điều tra) là tách các toạ độ trái trên và phải dưới của tất cả các ô vuông hoặc ô chữ nhật nằm xen kẻ trong văn bảnvà tính diện tích của các ô đó. Diện tích ở đây là tổng tất cả các pixel đen nằm trên biên của các ô (lưu ý rằng thông thường các ô mẫu là các ô mà không có các pixel đen nằm ở trong biên). Và sau đó lưu các toạ độ và diện tích của các ô trong ảnh mẫu vào một file văn bản (DBF).

2.4.2 Nhận dạng bài toán

Sau khi tiến hành bước thứ nhất. Để nhận dạng một bức ảnh ta tiến hành lấy toạ độ của các ô vuông đã lưu trong file văn bản ở bước 1. Và sau đó tính diện tích của ô vuông này trong bức ảnh cần nhận dạng (lưu ý rằng do bức ảnh cần nhận dạng và ảnh mẫu đã được detecting offset và detecting skew trước khi tiến hành nhận dạng nên toạ độ các ô vuông tương ứng là giống nhau). Tiếp theo là so sánh diện tích của hai ô tương ứng nằm ở ảnh mẫu và ảnh cần nhận dạng có hai khả năng xảy ra:

1) Khả năng thứ nhất diện tích của ô nằm ở ảnh cần nhận dạng lớn hơn ảnh mẫu một ngưỡng θ thì ô đó có đánh dấu.
2) Khả năng thứ hai là ngược lại của trường hợp thứ nhất thì ô đó không đánh dấu.

Kết quả một ô nào đó có đánh dấu hay không đều được lưu vào trong file văn bản theo cấu trúc như sau:

Record	htren	ctren	hduoi	cduoi	dien_tich	danh_dau
1	x11	y11	x12	y12	s1	co
2	x21	y21	x22	y22	s2	khong
3	x31	y31	x32	y32	s3	khong
. . .						
i	xi1	yi1	xi2	yi2	si	co

CHƯƠNG III

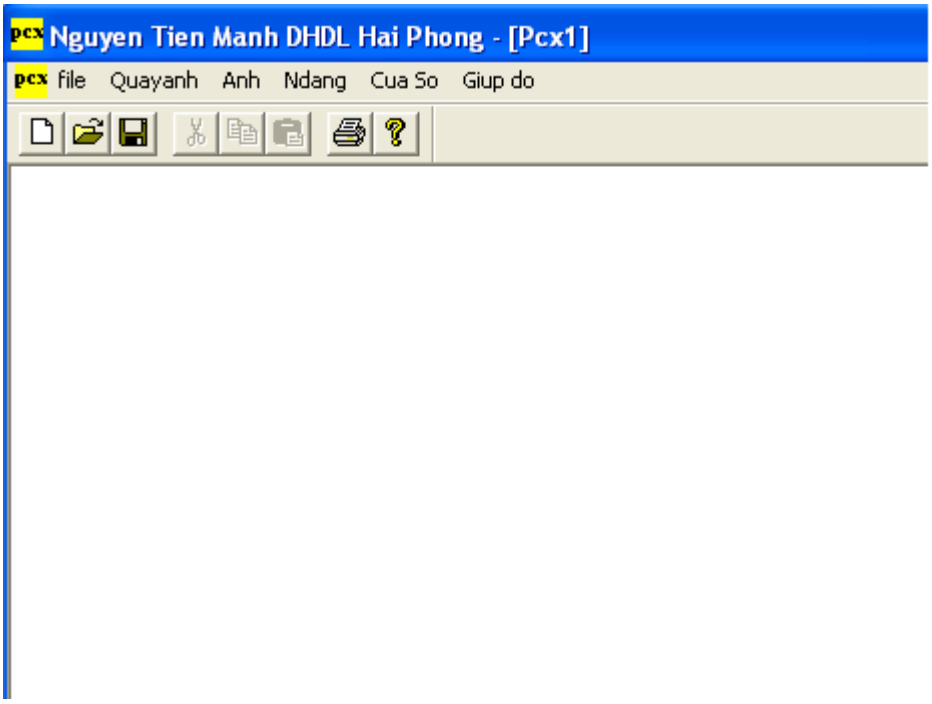
KẾT QUẢ CHƯƠNG TRÌNH VÀ HƯỚNG NÂNG CAO

3.1 CÀI ĐẶT CHƯƠNG TRÌNH

Chương trình được viết bằng ngôn ngữ Visual C^{++} trên môi trường Window. Chương trình bao gồm các chức năng:

+ Tự động hiệu chỉnh độ dịch chuyển của ảnh mẫu và ảnh cần nhận dạng.

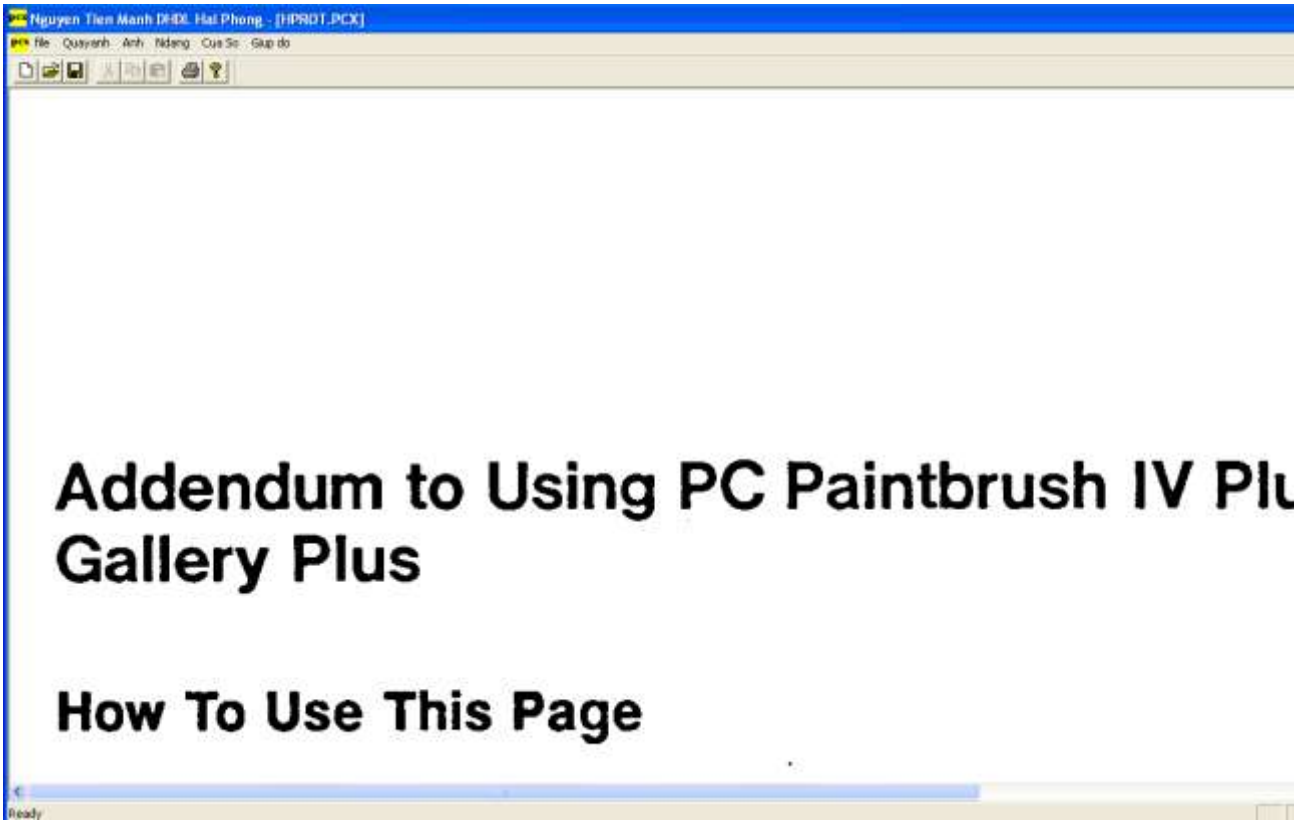
Giao diện chương trình:



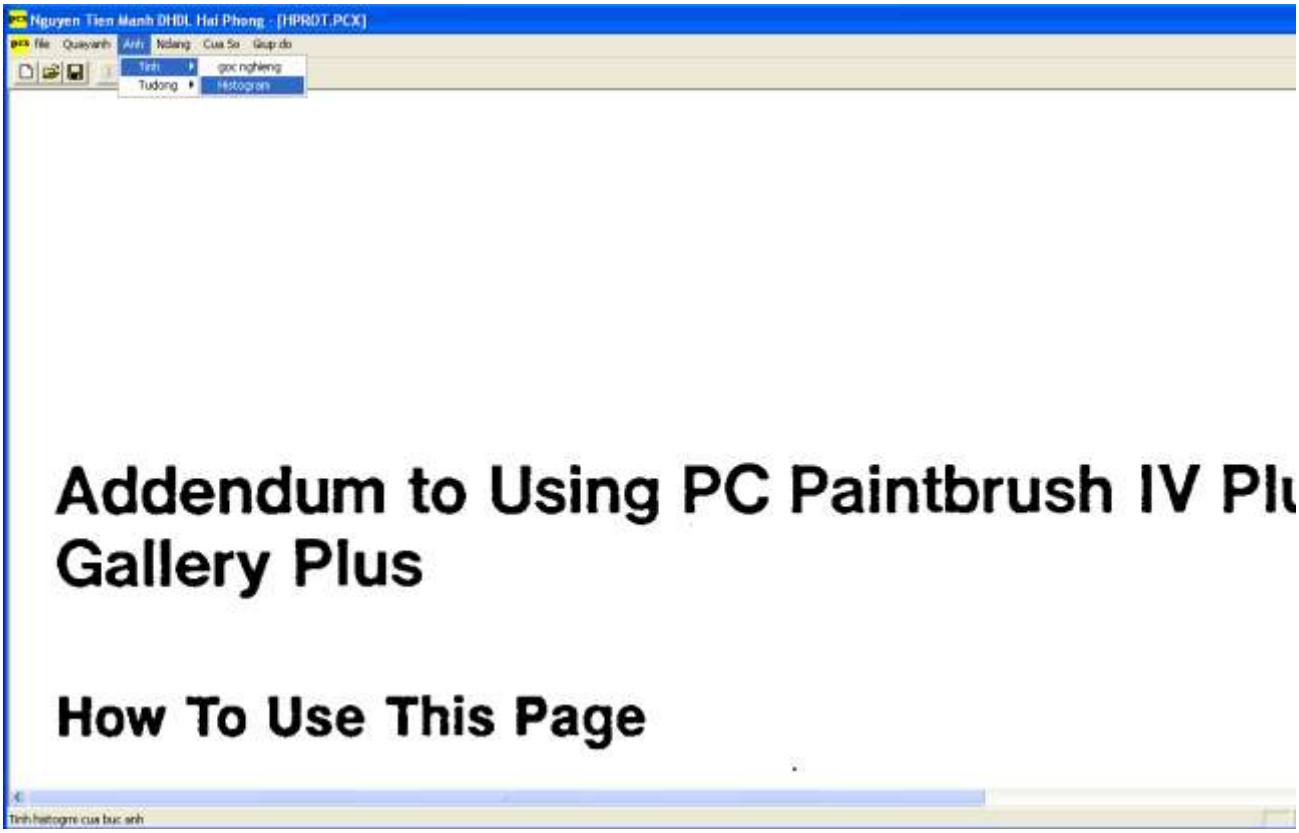
3.2 KẾT QUẢ

+ Phát hiện và điều chỉnh độ dịch chuyển văn bản: Với chức năng này chương trình đã đem lại kết quả khá chính xác đối với văn bản.

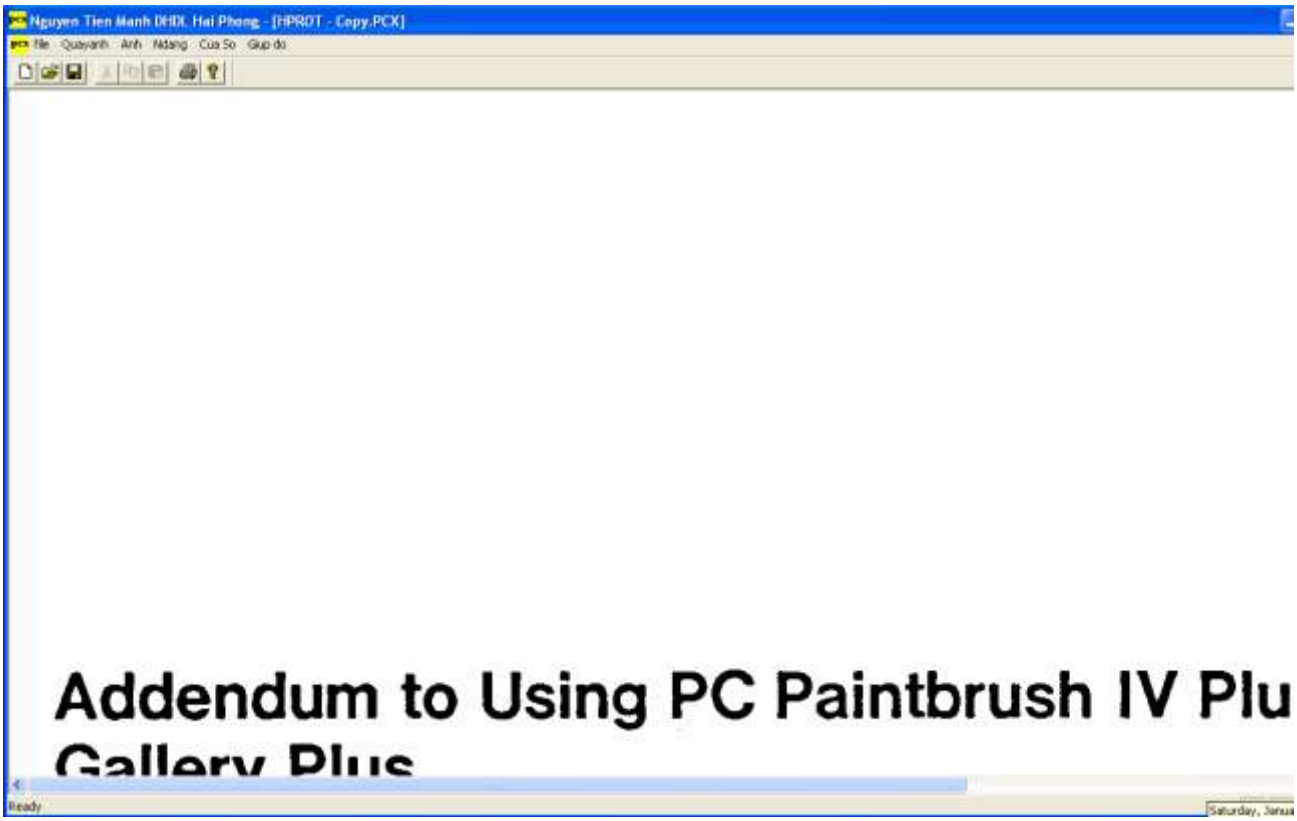
+Ảnh của văn bản gốc:



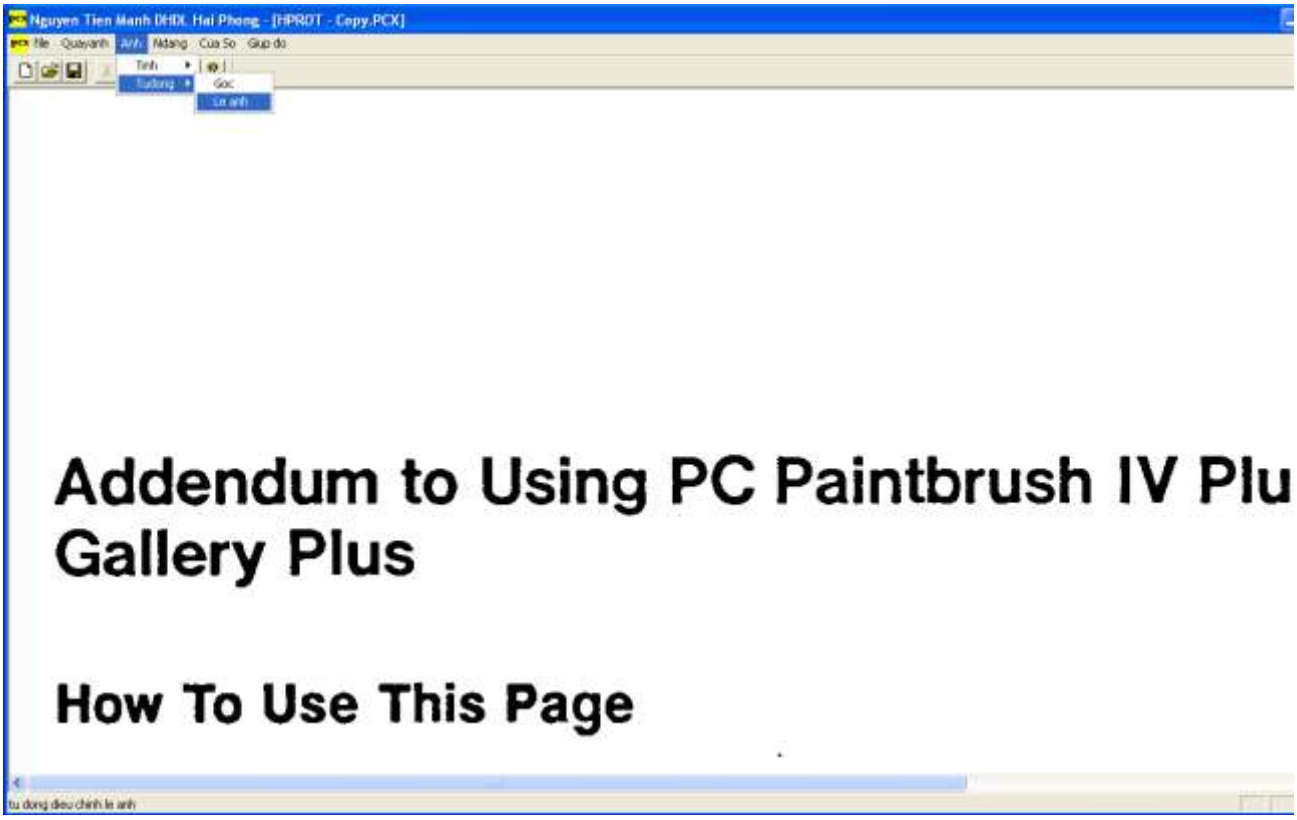
+ Tính Histogram của văn bản gốc:



+ Ảnh của văn bản cần nhận dạng (bị lệch so với ảnh gốc):



Và kết quả chương trình sau khi hiệu chỉnh tự động :



+ Ảnh sau khi được hiệu chỉnh độ dịch chuyển đã được đưa về đúng vị trí so với ảnh gốc.

+ Điều chỉnh offset của trang ảnh: Với chức năng này chương trình đã đem lại kết quả tương đối chính xác, Tuy nhiên với phương pháp đã đề ra trong luận văn đòi hỏi một khối lượng tính toán lớn vì vậy thời gian thực hiện chương trình khá lâu.

3.3 Ý NGHĨA ỨNG DỤNG:

Kết quả của việc nhập liệu tự động phụ thuộc rất nhiều vào quá trình tiền xử lý: như hiệu chỉnh độ dịch chuyển, hiệu chỉnh góc nghiêng, khử nhiễu,làm trơn biên làm đầy biên, xoá gai...Tuy nhiên với các chức năng ở trên đã phần nào trợ giúp cho nhập liệu tự động được chính xác hơn, ngoài ra việc phát hiện và tự động hiệu chỉnh độ dịch chuyển của trang văn bản còn là công cụ cho nhiều chức năng xử lý ảnh khác như nhận dạng chữ viết tay, chữ viết in...

3.4 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

Đồ án đã tiến hành nghiên cứu được các vấn đề sau:

+ Nghiên cứu các vấn đề, phương pháp phát hiện độ dịch chuyển trang văn bản so với văn bản gốc.

+ Điều chỉnh offset ảnh cần nhận dạng so với ảnh gốc.

Với những kỹ thuật trên đã phần nào trợ giúp cho hệ nhập liệu được nhanh chóng và chính xác hơn.

Tuy nhiên trong khuôn khổ đồ án do thời gian và kiến thức còn hạn chế nên việc nghiên cứu chỉ dừng lại ở mức cơ bản. Bài toán nhập liệu tự động là một bài toán lớn, nó bao gồm nhiều phần mà đồ án chỉ áp dụng và xử lý một phần nhỏ trong bài toán này. Vì vậy hướng phát triển của đề tài gồm các hướng như sau:

+ Phát hiện và hiệu chỉnh góc nghiêng của văn bản.

+ Tách các đối tượng nằm bất kỳ trong văn bản.

+ Khử nhiễu, làm trơn biên làm đầy biên, xóa gai....

Đây là bài toán phức tạp liên quan đến nhập liệu tự động, hiện nay loại bài toán kiểu này đã và đang được nghiên cứu bởi nhiều tác giả. Nó vẫn đang là bài toán mở.

PHỤ LỤC

Trong phần này luận văn sẽ đưa ra một số thủ tục đã sử dụng trong đồ án:

+ Đọc và hiển thị một ảnh PCX.

Còn các thủ tục khác như phát hiện độ độ dịch chuyển trang văn bản, điều chỉnh offset trang văn bản đã được đề cập khá chi tiết trong các chương trước của đồ án:

```
HDIB WINAPI ReadPCXFile(LPCSTR fName)
{
	#ifdef _WIN32
	HFILE hf = _lopen(fName, OF_READ);
	#else // 16 bit
	HFILE hf = _lopen(fName, READ);
	#endif// _WIN32
	if (!hf) return NULL;
	::SetCursor(::LoadCursor(NULL, IDC_WAIT));
	HDIB hDIB = ::ReadPCXFile(hf);
	_lclose(hf);
	::SetCursor(::LoadCursor(NULL, IDC_ARROW));
	return hDIB;
}
HDIB WINAPI ReadPCXFile(HFILE hf)
{
	PCXHEADER pcx;
	if (!::ReadPCXHeader(hf, &pcx)) return NULL;
	// make a new bitmap header
	BITMAPINFOHEADER bmi;
	::InitBitmapInfoHeader(&bmi, (DWORD)(pcx.window.xmax-
pcx.window.xmin+1),
                                        (DWORD)(pcx.window.ymax-
pcx.window.ymin+1), pcx.bitsperpixel);
	// Locate the memory
	HDIB hDIB = ::GlobalAlloc(GMEM_MOVEABLE,
(DWORD)sizeof(BITMAPINFOHEADER) +
```

```
(DWORD)::PaletteSize((LPSTR)&bmi) +

bmi.biSizeImage);
        if (!hDIB) return NULL; // Fail
        LPBITMAPINFOHEADER pDIB =
(LPBITMAPINFOHEADER)::GlobalLock(hDIB);
        *pDIB = bmi;        // Put the header
        // Calculate number of byte per line
        DWORD wBytes = (WORD)WIDTHBYTES(pDIB->biWidth*pDIB-
>biBitCount);
        // Get DIB line 0
        HBYTE pLine = ((HBYTE)::FindDIBBits((LPSTR)pDIB)) + wBytes*(pDIB-
>biHeight-1);
        WORD        sizeBuff = 10240,  // 10 KB
                    index = 10, cr = 0, tmp = 0;
        HGLOBAL hBuffers = ::GlobalAlloc(GMEM_MOVEABLE, sizeBuff+64);
        HBYTE       pBuffers = (HBYTE)::GlobalLock(hBuffers);
        // Decode
        for (int i = 0; i < (int)pDIB->biHeight; i++)
        {
                DWORD total = 0;
                while (total < pcx.bytesperline)
        {
                        if (index >= cr) // Buffers
                        {
                                if ((tmp > 0)&&(index == cr)) pBuffers[0]=pBuffers[index];
                                else tmp = 0;
                        index = 0;
                                #ifdef _WIN32
                                cr = _lread(hf, (LPVOID)(pBuffers+tmp), sizeBuff);
                                #else // 16 bit
                                cr = _lread(hf, (void _huge*)(pBuffers+tmp), sizeBuff);
                                #endif// _WIN32
                                if (!tmp) {tmp = 1; cr--;}
                        }
                        static BYTE b;
                        if ((b = pBuffers[index++]) >= 0xC0) // Get first byte
                        {
                                b &= 0x3F;
                                if (total < wBytes)
```

```
                                        #ifdef _WIN32
                                        memset((void*)(pLine+total), pBuffers[index++],
min((int)b, (int)(wBytes-total)));
                                        #else // 16 bit
                                        _fmemset((void __far*)(pLine+total),
pBuffers[index++], min((int)b, (int)(wBytes-total)));
                                        #endif// _WIN32
                                total += (WORD)b;
                        }
                        else if (total < wBytes) pLine[total++] = b;
                                 else total++;
                }
                pLine -= (LONG)wBytes;
        }
        LPRGBQUAD lpRGB = (LPRGBQUAD)(pDIB + 1);
        if (pDIB->biBitCount == 1) // Create the Look Up Table
        {
                lpRGB[0].rgbRed = lpRGB[0].rgbGreen = lpRGB[0].rgbBlue = 0; //
Black
                lpRGB[1].rgbRed = lpRGB[1].rgbGreen = lpRGB[1].rgbBlue = 255; //
White
                lpRGB[0].rgbReserved = lpRGB[1].rgbReserved = 0;
        } else // 8 bit image, read LUT from file
        {
                #ifdef _WIN32
                _llseek(hf, -768, FILE_END);
                _lread(hf, (LPVOID)pBuffers, 768); // Read
                #else // 16 bit
                _llseek(hf, -768, 2); //FILE_END
                _lread(hf, (void _huge*)pBuffers, 768); // Read
                #endif// _WIN32
                for (i = 0; i < 256; i++) // Convert to RGBQUAD
                {
                        lpRGB[i].rgbRed             = pBuffers[i*3];
                        lpRGB[i].rgbGreen           = pBuffers[i*3+1];
                        lpRGB[i].rgbBlue   = pBuffers[i*3+2];
                        lpRGB[i].rgbReserved = 0;
                }
        }
        ::GlobalUnlock(hDIB);
        ::GlobalUnlock(hBuffers);
```

```
        ::GlobalFree(hBuffers);
        return hDIB;
}
/*-------------------------------------------------------------------*/
BOOL WINAPI ReadPCXHeader(HFILE hf, LPPCXHEADER pcxh)
{
        // Read the file's header
        #ifdef _WIN32
        if (_lread(hf, (LPVOID)pcxh, 128) != 128) return FALSE;
        #else // 16 bit
        if (_lread(hf, (void _huge*)pcxh, 128) != 128) return FALSE;
        #endif// _WIN32
        if ( pcxh->manufacture != 0x0A ) // Check manufacture of the PCX file
                return FALSE;
        // Only work with B/W and 8 bit image
        if ((pcxh->bitsperpixel*pcxh->nplanes != 1) &&
                (pcxh->bitsperpixel*pcxh->nplanes != 8))
                return FALSE;
        if (pcxh->encoding != 1) // Unknow how to decode
                return FALSE;
        return TRUE;
}
/*-------------------------------------------------------------------*/
VOID WINAPI CreatePCXHeader(LPPCXHEADER pcxh,
LPBITMAPINFOHEADER lpDIB)
{
        pcxh->manufacture       = 0x0A;       // Signature
        pcxh->version           = (lpDIB->biBitCount == 1) ? 2 : 5;    // PCX version
        pcxh->encoding          = 0x01;       // Run length
        pcxh->bitsperpixel      = (char)lpDIB->biBitCount;
        pcxh->window.xmin       = 0;
        pcxh->window.ymin       = 0;
        pcxh->window.xmax       = (int)lpDIB->biWidth -1;
        pcxh->window.ymax       = (int)lpDIB->biHeight-1;
        pcxh->hres              = (WORD)lpDIB->biXPelsPerMeter;
        pcxh->vres              = (WORD)lpDIB->biYPelsPerMeter;
        pcxh->reserved          = 0x00;
        pcxh->nplanes           = 1;
        pcxh->bytesperline      = (WORD)WIDTHBYTES(lpDIB-
>biBitCount*lpDIB->biWidth);
        pcxh->palette_info= 1;
```

```
	for (int i = 0; i < 58; i++) pcxh->filler[i] = 0;
	if (lpDIB->biBitCount == 1)
	{	// create LUT
		pcxh->colormap[0] = pcxh->colormap[1] = pcxh->colormap[2] = 0;
		pcxh->colormap[3] = pcxh->colormap[4] = pcxh->colormap[5] = 0;
	}
}
/*-------------------------------------------------------------------*/
DWORD WINAPI CompressLine(HBYTE pDes, HBYTE pSource, DWORD Bytes)
{
	DWORD j = 0, iw = 0;
	while ( j < Bytes )
	{
		BYTE Count = 1;
		BYTE item  = pSource[j];
		while ((j < Bytes-1) && (item == pSource[j+1]) && (Count < 0xFF-
0xC0-1))
		{
			j++;
			Count++;
		}
		if ((Count > 1)||(item >= 0xC0))
		{
			pDes[iw++] = Count + 0xC0;
			pDes[iw++] = item;
		}
		else pDes[iw++] = item;
		j++;
	}
	return iw;
}
/*===================================================*/
```

TÀI LIỆU THAM KHẢO

1. Tài Liệu về Xử Lý Ảnh của PGS TS Đỗ Năng Toàn Và TS Phạm Việt Bình của trường Đại Học Thái Nguyên biên soạn.

2. Trịnh Thế Phong Trường đại học khoa học Huế , Nhập liệu tự động, Luận văn tốt nghiệp , năm 1999.

3. Lương Mạnh Bá, Nguyễn Thanh Thuỷ, Nhập Môn Xử lý ảnh số, Nhà xuất bản Thống kê, tháng năm 1998.

4. Trang tìm kiếm: http://www.google.com