

MỤC LỤC

LỜI NÓI ĐẦU	2
CHƯƠNG I KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ GHÉP ẢNH	4
1.1. Tổng quan về xử lý ảnh	4
1.1.1 Giới thiệu chung về xử lý ảnh.....	4
1.1.2 Ảnh và biểu diễn ảnh.....	7
1.1.3 Ảnh xám	10
1.1.4 Biến đổi ảnh	12
1.2. Ghép ảnh và ảnh Panorama.....	13
1.2.1. Kỹ thuật ghép ảnh	13
1.2.2 Ảnh panorama	19
Chương 2: GHÉP ẢNH DỰA TRÊN KỸ THUẬT NẮN CHỈNH HÌNH	
HỌC.....	26
2.1 Một số vấn đề của bài toán.....	26
2.1.1 Xác định các cặp điểm đặc trưng	26
2.1.2 Xác định ảnh cần nắn chỉnh	26
2.2 Nắn chỉnh hình dạng bức ảnh.....	27
2.2.1 Sơ đồ thuật toán ghép ảnh dựa trên nắn chỉnh hình học	27
2.2.2 Xây dựng thuật toán nắn chỉnh dựa vào các cặp điểm đặc trưng	27
2.2.3 Biến đổi hình dạng bức ảnh dựa trên hàm biến đổi hàm f.....	29
2.3. Ghép ảnh sau khi nắn chỉnh	30
Chương 3: CHƯƠNG TRÌNH THỬ NGHIỆM.....	32
3.1. Giới thiệu chương trình.....	32
3.2. Các chức năng của chương trình.....	32
KẾT LUẬN	38
TÀI LIỆU THAM KHẢO.....	40
PHỤ LỤC	41

LỜI NÓI ĐẦU

Ngày nay, Việt Nam đang bước vào hội nhập với nền kinh tế của thế giới, một điều đặt ra là: Làm sao để quảng bá hình ảnh Việt Nam nhiều hơn nữa với bạn bè quốc tế? Làm sao để những hình ảnh đất mẹ phải sống động, mới mẻ, quyến rũ chứ không lặp lại những cảm xúc đơn điệu? Từ lâu rồi các nhiếp ảnh gia đã dành trọn sự tìm tòi và đam mê của mình để vượt qua những câu hỏi đó với mong muốn được góp một phần vào việc "tiếp thị" hình ảnh Việt Nam. Tất cả họ đều mong muốn rằng thông qua những bức ảnh đó người xem có thể làm một chuyến “du lịch ảo” từ Bắc chí Nam qua những danh lam thắng cảnh nổi tiếng như Văn Miếu (Hà Nội), Sầm Sơn (Thanh Hóa), Cửa Lò (Nghệ An), Mỹ Sơn (Quảng Nam), lăng Khải Định (Huế), hồ Xuân Hương (Đà Lạt), TP.HCM... Điều thực sự mới mẻ và thú vị là những thắng cảnh này không phải được giới thiệu bằng những hình ảnh đơn chiều mà được giới thiệu bằng không gian “giả” đa chiều thật sống động khiến người xem có cảm giác không khác gì đứng trước cảnh thật.

Tuy nhiên để có những bức ảnh như thế thật không phải đơn giản. Khi chụp ảnh, độ rộng của ống kính không đủ để tạo nên những bức ảnh đó, bởi vậy ghép ảnh để tạo nên những bức ảnh đẹp là một phương pháp rất hay.



Hình 1 Ảnh được ghép từ 14 tấm khác nhau

Ghép ảnh còn có rất nhiều ứng dụng trong thực tế như trong ngành kiến trúc, xây dựng bản đồ địa lý...v.v..

Song việc ghép các thành phần của các đối tượng lại với nhau để thu được các ảnh tương ứng hoàn thiện hơn là một công việc khó khăn rất nhiều khi phải làm thủ công, mặt khác các ảnh khi thu nhận để ghép thường bị lệch hay biến dạng đi một khoảng nào đấy. Yêu cầu đặt ra cần xác định khoảng sai lệch về thông tin giữa các phần ảnh định ghép, sau đó hiệu chỉnh độ sai lệch và cuối cùng là ghép chúng lại. Nghiên cứu kỹ thuật ghép ảnh còn mở ra cho chúng ta một hướng phát triển mới trong tương lai đó là xây dựng kỹ thuật giả lập 3D.

Xuất phát từ vấn đề này, đề án của em là “Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh ”

Đề án của em gồm các phần sau :

Chương 1: KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ GHÉP ẢNH

Chương này trình bày tổng quan về bộ môn xử lý ảnh, phương pháp ghép ảnh và một số nội dung về ảnh panorama.

Chương 2: GHÉP ẢNH DỰA TRÊN KỸ THUẬT NẮN CHỈNH HÌNH HỌC

Chương này giới thiệu về thuật toán ghép ảnh dựa trên kỹ thuật nắn chỉnh hình học với các tập điểm đặc trưng đầu vào.

Chương 3: CHƯƠNG TRÌNH THỬ NGHIỆM

Chương 1: KHÁI QUÁT VỀ XỬ LÝ ẢNH VÀ GHÉP ẢNH

1.1. Khái quát về xử lý ảnh

1.1.1 Giới thiệu chung về xử lý ảnh

Xử lý ảnh là một môn khoa học tương đối mới mẻ so với nhiều ngành khoa học khác, nhất là trên quy mô công nghiệp, đó là một trong những mảng quan trọng nhất trong kỹ thuật thị giác máy tính, là tiền đề cho nhiều nghiên cứu thuộc lĩnh vực này. Hai nhiệm vụ cơ bản của quá trình xử lý ảnh là nâng cao chất lượng thông tin hình ảnh và xử lý số liệu cung cấp cho các quá trình khác trong đó có việc ứng dụng thị giác vào điều khiển.

Đã có rất nhiều công trình nghiên cứu tại nhiều quốc gia từ năm 1920 đến nay về xử lý ảnh đã góp phần thúc đẩy tiến bộ trong lĩnh vực này lớn mạnh không ngừng [GS. TS. Nguyễn Kim Sách].

Quá trình xử lý ảnh bắt đầu từ việc thu nhận ảnh nguồn (từ các thiết bị thu nhận ảnh dạng số hoặc tương tự) gửi đến máy tính. 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 hóa (loại CCD – Charge Coupled Device). Ảnh cũng có thể thu nhận từ vệ tinh qua các bộ cảm ứng (sensor), hay ảnh, tranh được quét trên scanner.

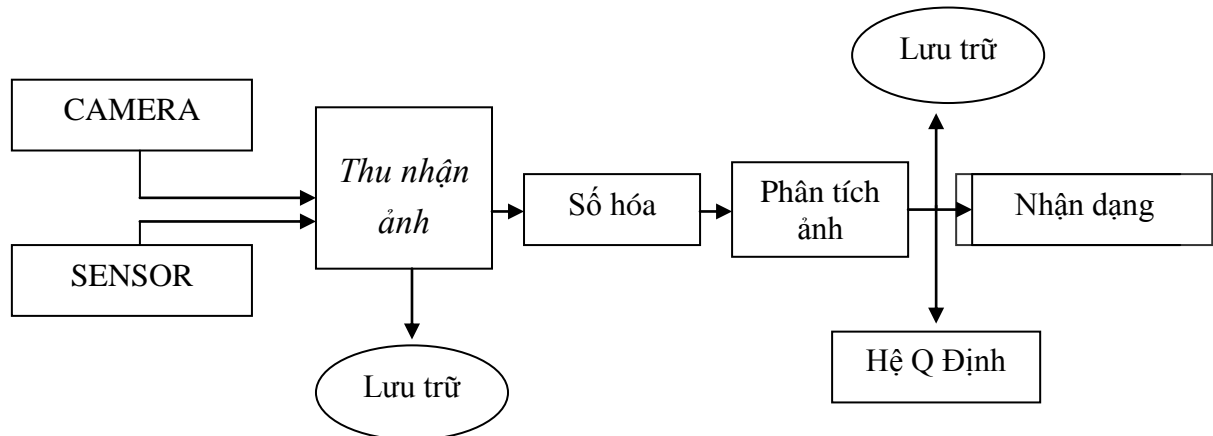
Tiếp theo là quá trình số hóa. Quá trình số hóa (Digitalizer) để biến đổi tín hiệu tương tự sang tín hiệu rời rạc (lấy mẫu) và số hóa bằng lượng hóa, trước khi chuyển sang giai đoạn xử lý, phân tích hay lưu trữ lại.

Quá trình phân tích ảnh thực chất bao gồm nhiều công đoạn nhỏ. Trước hết là công việc tăng cường ảnh 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 lại ảnh để làm nổi bật một số đặc tính chính của ảnh, hay làm cho ảnh gần giống

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

với trạng thái gốc – trạng thái trước khi ảnh bị biến dạng. Giai đoạn tiếp theo là phát hiện các đặc tính như biên, phân vùng ảnh, trích chọn các đặc tính, v.v....

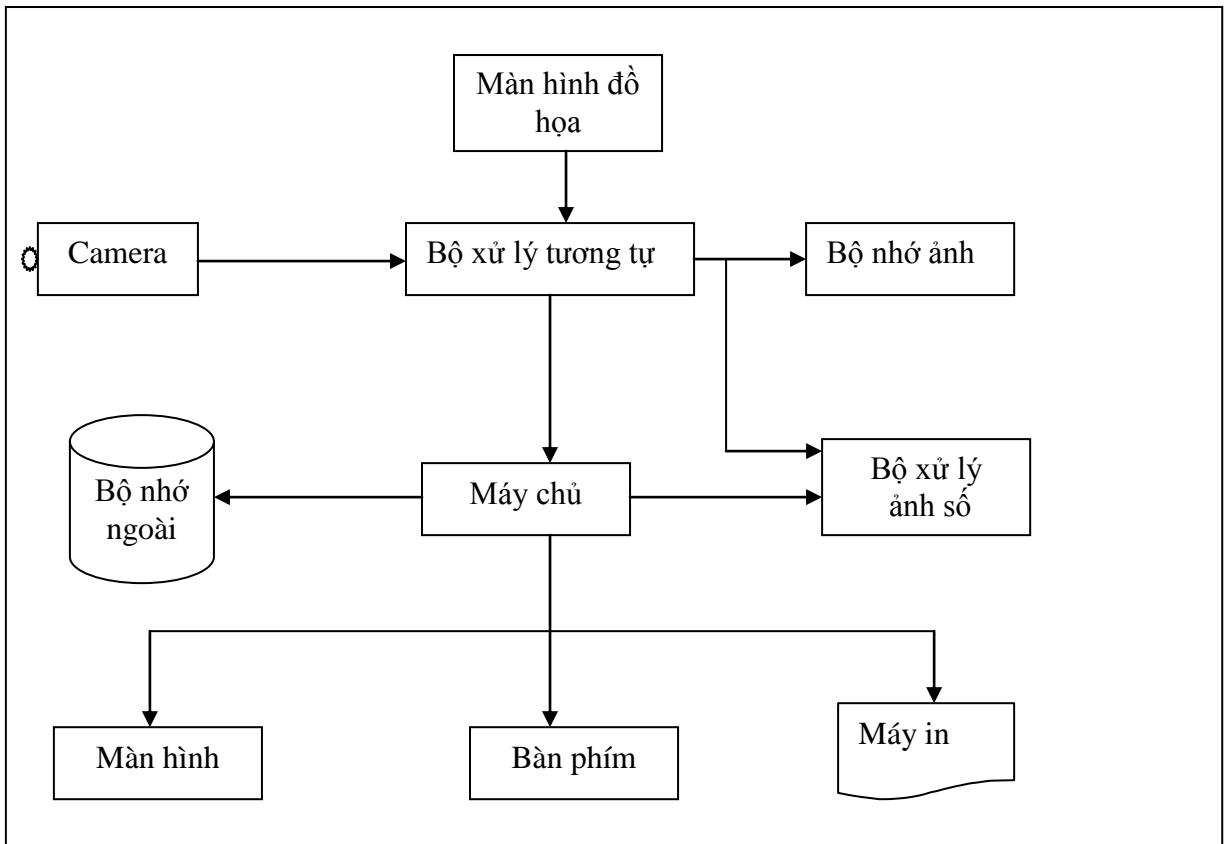
Cuối cùng, tùy theo mục đích của ứng dụng, sẽ là giai đoạn nhận dạng, phân lớp hay các quyết định khác. Các giai đoạn chính của quá trình xử lý ảnh có thể mô tả như hình 1.1



Hình 1.1 Các giai đoạn chính trong xử lý ảnh

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

Với các giai đoạn trên, một hệ thống xử lý ảnh (cấu trúc phần cứng theo chức năng) bao gồm các thành phần tối thiểu như trong hình 1.2



Hình 1.2 Cấu trúc phần cứng theo chức năng của hệ thống xử lý ảnh.

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 có 625 dòng. Loại CCD gồm các photo điốt và làm tương ứng một cường độ sáng tại một điểm ảnh ứng 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) 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 hóa (digitalizer). Thực hiện lấy mẫu và mã hóa.
- Tiền xử lý ảnh khi thu nhận: Dùng kỹ thuật bảng tra(Look Up Table-LUT).

Bộ xử lý ảnh số gồm nhiều bộ xử lý chuyên dụng như: 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 512×512 với 256 mức xám chiếm 256K bytes. Với một ảnh màu cùng kích thước dung lượng sẽ tăng gấp 3 lần.

1.1.2 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 xem 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.

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

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 phân 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:

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

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

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

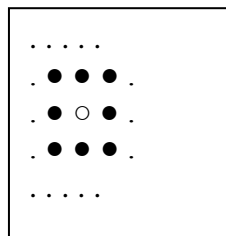
Ở đó $S_R(\lambda)$, $S_G(\lambda)$ và $S_B(\lambda)$ theo thứ tự là những đặc tính phổ của các cảm biến (bộ lọc) đỏ, lục và lam. R , G , B cũng không âm và hữu hạn.

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

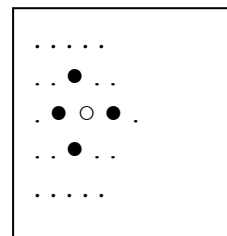
Mô hình Vector: Ngoài mục đích tiết kiệm không gian lưu trữ, dễ dàng hiển thị và in ấn, các ảnh biểu diễn theo mô hình vector còn có ưu điểm cho phép dễ dàng lựa chọn, sao chép, di chuyển, tìm kiếm... Theo những yêu cầu này thì kỹ thuật biểu diễn vector tỏ ra ưu việt hơn. Trong mô hình này, người ta 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 1.3 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° (Hình 1.3).

1.1.3 Ảnh xám

Ảnh xám là ảnh chỉ có các màu sắc độ xám. Thực chất màu xám là màu có các thành phần R,G,B trong hệ thống màu RGB có cùng cường độ. Tương ứng với mỗi điểm ảnh sẽ có một mức xám xác định. Ảnh có nhiều mức xám được gọi là ảnh đa cấp xám, ảnh chỉ có hai mức xám 0 và 1 được gọi là ảnh nhị phân.

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 tử hóa. Cách mã hóa kinh điển thường dùng 16, 32 hay 64 mức. Mã hóa 256 mức là phổ dụng nhất vì lý do kỹ thuật. Vì $2^8=256$ (0..255) nên với 256 mức mỗi pixel sẽ được mã hóa bởi 8 bit.

Lược đồ mức xám (Histogram) hay còn gọi là lược đồ xám của một ảnh là một hàm cung cấp tần suất xuất hiện của mỗi mức xám (Grey level).

Lược đồ xám của một ảnh số có các mức xám trong khoảng $[0, L-1]$ là một hàm rời rạc $p(r_k)=n_k/n$. Trong đó n_k là số pixel có mức xám thứ r_k , n là tổng số pixel của ảnh và $k = 1, 2, 3, \dots, L-1$. Do đó $p(r_k)$ cho một xấp xỉ xác suất xảy ra mức xám r_k . Vẽ hàm này với tất cả các giá trị của k sẽ biểu diễn khái quát sự xuất hiện các mức xám của một ảnh. Chúng ta cũng có thể thể hiện lược đồ mức xám của ảnh thông qua tần suất xuất hiện mỗi mức xám qua hệ tọa độ vuông góc Oxy. Trong đó, trục hoành biểu diễn số mức xám từ 0 đến N (số bit của ảnh xám). Trục tung biểu diễn số pixel của mỗi mức xám.

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

Bắt đầu

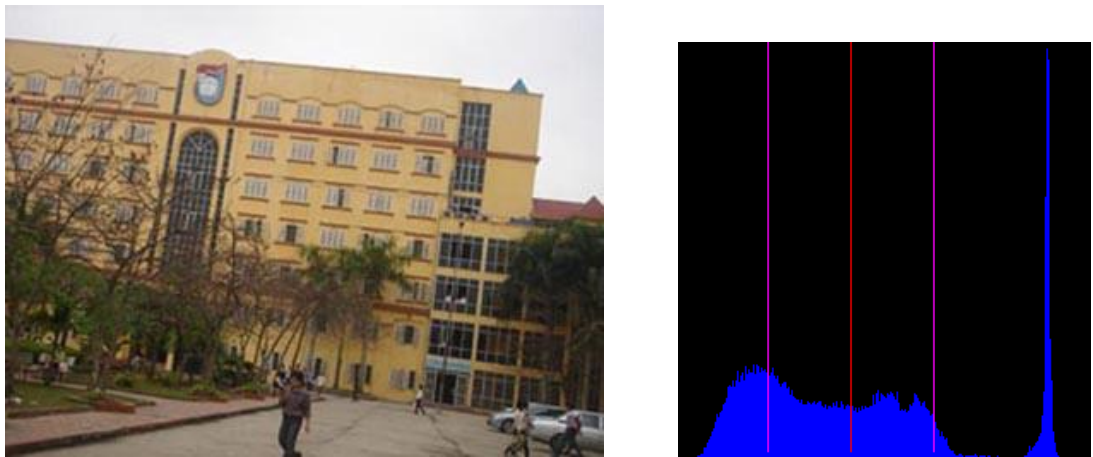
H là bảng chứa lược đồ xám (là véc tơ có N phần tử)

Khởi tạo bảng: Đặt tất cả các phần tử của bảng bằng 0

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

Tính giá trị max của bảng H. Sau đó hiển bảng trong khoảng từ 0 đến Max.

Kết thúc



Hình 1.4 Ví dụ về lược đồ xám

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 tính động của ảnh. Tính động cho phép phân tích trong khoảng nào đó phân bố phần lớn các mức xám của ảnh: ảnh rất xám hay rất đậm. Nếu ảnh sáng, lược đồ xám nằm bên phải (mức xám cao), còn ảnh đậm thì lược đồ xám nằm bên trái (mức xám thấp).

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.

Trong hầu hết quá trình xử lý ảnh, chúng ta chủ yếu chỉ quan tâm đến cấu trúc của ảnh và bỏ qua ảnh hưởng của yếu tố màu sắc. Do đó bước chuyển từ ảnh màu thành ảnh xám là một công đoạn phổ biến trong các quá trình xử lý ảnh vì nó làm tăng tốc độ xử lý là giảm mức độ phức tạp của các thuật toán trên ảnh.

Chúng ta có công thức chuyển các thông số giá trị màu của một pixel thành mức xám tương ứng như sau:

$$G = \alpha.C_R + \beta.C_G + \delta.C_B$$

Trong đó các giá trị C_R, C_G và C_B lần lượt là các mức độ màu đỏ, xanh lá và xanh biển của pixel màu. Các hệ số α, β , và δ là các giá trị thay đổi tùy thuộc hệ màu.

1.1.4 Biến đổi ảnh

Thuật ngữ biến đổi ảnh thường được 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 dưới 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 $AA^{*T} = I$. Các $A^*_{k,l}$ được định nghĩa ở trên với $k, l = 0, 1, 2, \dots, N-1$ là ảnh cơ sở. Có nhiều loại biến đổi được dùng như :

- Biến đổi Fourier, Sin, Cosin, Hadamard....
- Tích Kronecker.
- Biến đổi KL(Krhumen loeve).

Do phải xử lý nhiều thông tin, các phép toán nhân và cộng trong khai triển là quá lớn. Do vậy các phép biến đổi trên nhằm làm giảm thứ nguyên của ảnh để việc xử lý ảnh được hiệu quả hơn.

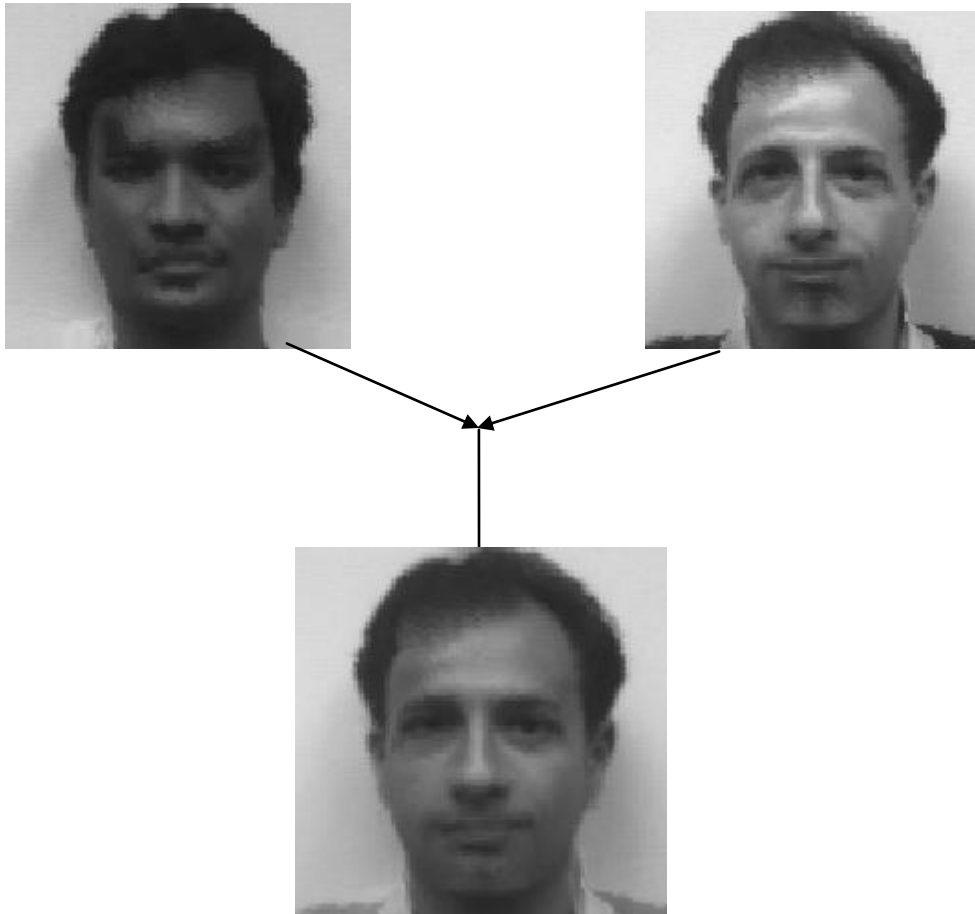
1.2 Ghép ảnh và ảnh Panorama

1.2.1. Kỹ thuật ghép ảnh

Thuật toán ghép ảnh dựa trên kỹ thuật morphing

Morphing là một kỹ thuật xử lý ảnh được sử dụng để thay đổi từ một ảnh này sang ảnh khác. Ý tưởng đó là đưa ra một chuỗi những ảnh trung gian, những ảnh trung gian này lần lượt theo thứ tự đều có những thay đổi nhỏ so với ảnh trước nó. Phương pháp đơn giản nhất của việc chuyển một ảnh này sang một ảnh khác là trộn hình ảnh chéo nhau giữa chúng. Điều này không làm ảnh hưởng nhiều đến việc đưa ra sự thay đổi thực tế. Với phương pháp này, sự chuyển đổi sẽ không cho kết quả tốt nếu giữa 2 ảnh không có cùng hình dạng xấp xỉ.

Ví dụ:



Hình 1.5 Ví dụ về sử dụng kỹ thuật morphing trong ghép ảnh

Sau đây là một thuật toán đơn giản để thực hiện việc trộn 2 ảnh với nhau: Giả sử ta sẽ thực hiện việc biến đổi ảnh từ một hình được biểu diễn bởi tập điểm $\{A_i\}$. Hình được chuyển tới (hình đích) được thể hiện bởi tập điểm $\{B_i\}$. Trong quá trình biến đổi từ $A \rightarrow B$ sẽ đi qua hình trung gian được thể hiện bởi tập điểm $\{P_i\}$. Cách tính P_i :

$$P[i].x = (1 - t) * A[i].x + t * B[i].x;$$

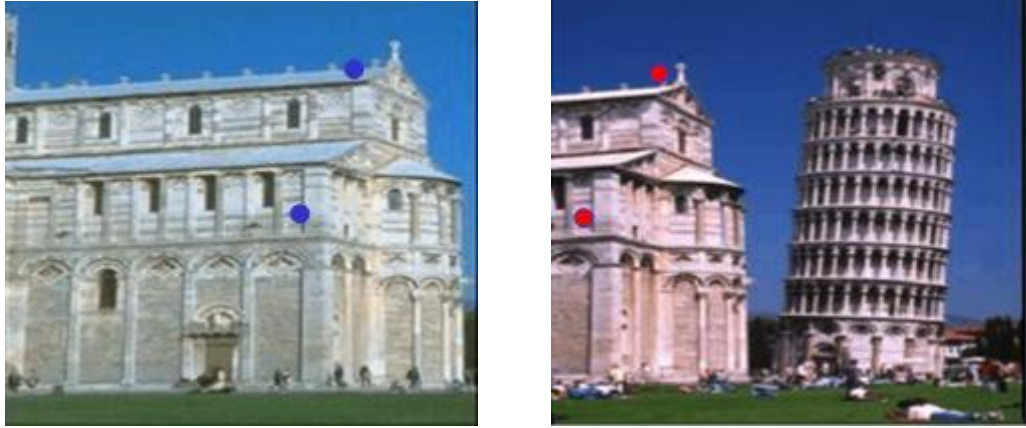
$$P[i].y = (1 - t) * A[i].y + t * B[i].y;$$

Trong đó, t là độ biến đổi giữa các hình ($0 < t < 1$); Để thực hiện việc biến đổi hình, ta chỉ việc thực hiện các phép tính tọa độ trên để tính r các hình trung gian P_i . Chú ý rằng $\{A_i\}$ và $\{B_i\}$ phải có cùng số điểm.

Như vậy để ghép 2 ảnh với nhau, trước hết ta chọn vùng cần ghép. Tiếp đến ta thực hiện trộn 2 vùng ảnh tương đồng với nhau theo kỹ thuật morphing.

Thuật toán ghép ảnh dựa trên kỹ thuật nắn chỉnh học

Trong thực tế khi thu nhận ảnh đối với các đối tượng có kích thước lớn, người ta thường phải tiến hành thu nhận từng phần. Việc thu nhận từng phần sẽ gây ra sự biến dạng hình học của đối tượng. Hơn nữa, góc độ ánh sáng khi nhận ảnh ở các vị trí khác nhau sẽ cho ta hiệu ứng ánh sáng thu nhận trên ảnh là khác nhau. Thực tế ta vẫn thấy bìa các quyển sách có thể hiện 2 hình ảnh ghép chụp ở các lần lần khác nhau sẽ có sắc màu khác nhau. Trong hình 1.6 là minh họa về sự biến dạng hình của các ảnh của cùng một đối tượng với góc độ chụp khác nhau.



Hình 1.6 Sự biến dạng hình học giữa 2 ảnh

Việc khắc phục sai lệch về hình dạng, thường do nguyên nhân bởi các thiết bị điện tử và quang học. Để khắc phục người ta thường sử dụng các kỹ thuật nắn chỉnh, thông qua các phép chiếu bởi các điểm điều khiển.

Để giải quyết vấn đề này ta dựa trên 2 tập điểm điều khiển $P(P_i)$ và $P'(P'_i)$. Từ 2 tập điểm điều khiển này xây dựng một hàm ánh xạ từ tập P sang tập P' :

$$f : \mathcal{P}_i \rightarrow f(P_i)$$

Với điều kiện khoảng cách từ điểm P_i tới điểm $f(P_i)$ đạt đến min. Tức là:

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

Với điều kiện này ta xây dựng được hàm f . Dựa vào hàm f ta tiến hành nắn chỉnh một trong 2 bức ảnh sao cho phù hợp với ảnh còn lại. Sau đó tiến hành ghép 2 ảnh sau khi được nắn chỉnh.

Thuật toán ghép ảnh dựa vào kỹ thuật nắn chỉnh hình học bước cơ bản sau:

Bước 1: Xây dựng các cặp điểm đặc trưng.

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

Gieo n điểm đặc trưng P lên ảnh Second để xác định vùng ảnh ghép, sau đó gieo n điểm đặc trưng P' lên ảnh Primary. Các điểm đặc trưng nằm trên các vị trí cần ghép của 2 đối tượng ảnh. Mỗi điểm đặc trưng P_i trong ảnh Primary image tương ứng với điểm đặc trưng p'_i ;

$$P = \{p_1, p_2, \dots, p_n\}$$

$$P' = \{p'_1, p'_2, \dots, p'_n\}$$

Bước 2: Xây dựng thuật toán nắn chỉnh dựa vào các cặp điểm đặc trưng

Nắn chỉnh ảnh second Image dựa trên các cặp điểm đặc trưng, mỗi cặp điểm đặc trưng gồm có: một điểm thuộc ảnh Primary image và một điểm thuộc ảnh Second image. Các điểm không phải điểm đặc trưng sẽ được ánh xạ dựa vào các điểm đặc trưng.

Từ các cặp điểm đặc trưng (p_i, p'_i) với $0 < i < n$, xây dựng được hàm ánh xạ f thỏa mãn:

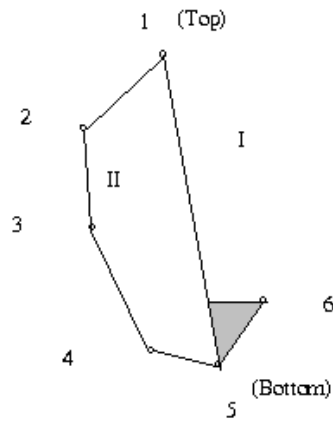
$$\sum_{i=1}^n \|P'_i - f(P_i)\|^2 \rightarrow \min_{\langle i \rangle}$$

Giả sử f có biến đổi tuyến tính $f(x, y) = (a_1x + b_1y + c_1, a_2x + b_2y + c_2)$, từ biểu thức $\langle i \rangle$ ta có thể tính được các hệ số $a_1, b_1, c_1, a_2, b_2, c_2$.

Sau đó lần lượt thực hiện ánh xạ một điểm ảnh $M(x, y)$ thuộc ảnh Primary thành điểm $f(M)$ bằng cách gán giá trị màu của M cho $f(M)$. Như vậy từ một ảnh second qua phép ánh xạ hàm f sẽ thu được ảnh mới tương thích với ảnh second.

Bước 3: Thuật toán ghép ảnh

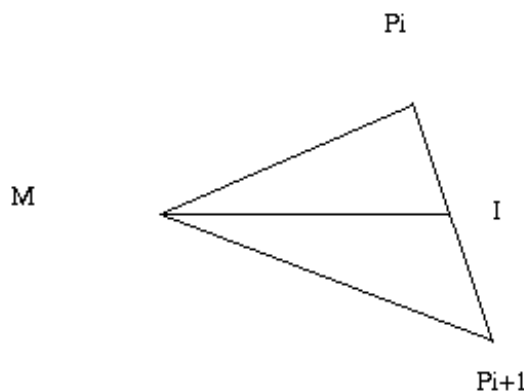
Thuật toán ghép dựa vào các điểm đặc trưng



Hình 1.7 Các điểm điều khiển

Ban đầu chia tập các điểm đặc trưng ra thành hai vùng, phân cách nhau bởi hai điểm Top và Bottom.

Với một điểm $M(x,y)$ thuộc vùng ảnh $\langle I \rangle$ xác định được 2 điểm đặc trưng (p_i, p_{i+1}) sao cho $y_i < y < y_{i+1}$, xác định điểm được một điểm $I(x_i, y_i)$ là giao điểm của đoạn thẳng đi qua M và song song với trục Ox với đoạn thẳng $p_i p_{i+1}$.



Hình 1.8 Xác định 2 điểm đặc trưng

Khi đó so sánh giá trị hoành độ của M và I , nếu $x < x_i$ thì M thuộc vùng ảnh 1, ngược lại M thuộc vùng ảnh 2.

Trường hợp có cặp điểm đặc trưng thuộc vùng ảnh 2 thì duyệt ngược lại.

Quét các điểm M thuộc ảnh đích, nếu M thuộc ảnh 1 thì gán giá trị màu của ảnh 1 cho M , ngược lại gán giá trị màu của ảnh 2 cho M . Quá trình lặp lại khi duyệt hết các điểm ảnh thuộc ảnh đích ta sẽ thu được ảnh đích.

Kỹ thuật này sẽ được phân tích rõ hơn trong chương sau.

Một kỹ thuật nắn chỉnh hình học đơn giản cho 2 ảnh bị xoay tịnh tiến và kích thước ảnh bị co hay nở ra:

Trích chọn 2 tập điểm đặc trưng, mỗi tập điểm đặc trưng chỉ gồm 2 điểm. 2 điểm trên mỗi ảnh sẽ tạo thành 1 vectơ trong không gian 2 chiều. Từ đó ta sẽ xác định được góc quay của vectơ(\vec{n}_1) so với vectơ(\vec{n}_2), cũng như là xác định được góc xoay của ảnh (α). Từ đó ta tiến hành xoay ảnh theo góc α . Như vậy ta đã giải quyết được trường hợp ảnh bị xoay tịnh tiến:

Ta có cách tính điểm ảnh $I'(x', y')$ mới của điểm ảnh $I(x, y)$ qua phép quay góc α :

$$x' = x \times \cos(\alpha) - y \times \sin(\alpha)$$

$$y' = y \times \cos(\alpha) + x \times \sin(\alpha)$$

Để xoay một bức ảnh ta lần lượt xoay từng điểm ảnh, sau đó gán giá trị màu theo từng điểm ảnh tương ứng.

Tuy nhiên trong quá trình quay ảnh, tọa độ chúng ta lấy có giá trị nguyên vì vậy không tránh khỏi việc ảnh bị mất thông tin, tức là có một số điểm ảnh bị mất đi trở thành những điểm trắng (hiện tượng lỗ). Để giải quyết vấn đề này chúng ta sử dụng một phương pháp như sau : Từ một điểm của ảnh kết quả, suy ngược lại từ ảnh gốc để lấy giá trị màu cần thiết. Với phương pháp này tất cả các điểm trên ảnh kết quả đều được gán giá trị màu của một điểm tương ứng (hay ít nhất cũng là điểm lân cận của điểm đó) ở ảnh gốc vì vậy sẽ không có hiện tượng 'lỗ' như trên.

$$x = x' \times \cos(\alpha) + y' \times \sin(\alpha)$$

$$y = y' \times \cos(\alpha) - x' \times \sin(\alpha)$$

Sau khi xoay ảnh, dựa vào 2 vectơ chúng ta tính được độ co giãn của ảnh này so với ảnh khác: $d = \frac{|\vec{n}_1|}{|\vec{n}_2|}$.

Nếu $d < 0$: tiến hành giãn ảnh.

Nếu $d > 0$: tiến hành co ảnh.

Ta có công thức đơn giản cho việc co (giãn) ảnh như sau:

Gọi w là tỷ lệ co (giãn) theo chiều rộng của ảnh 2 so với ảnh 1.

h là tỷ lệ co (giãn) theo chiều cao của ảnh 2 so với ảnh 1.

Tọa độ mới của điểm $I(i,j)$ của ảnh 1 được tính theo công thức sau:

$$i' = (\text{int})(i \div w);$$

$$j' = (\text{int})(j \div h);$$

1.2.2 Ảnh panorama

Khái niệm

Thuật ngữ panorama xuất hiện trước khi có chúng ta có máy ảnh panorama. Nguyên gốc của từ panorama được xác định là do họa sĩ người Ireland - Robert Baker dùng để mô tả những bức tranh diện rộng ở Edinburgh (Đức). Những bức tranh panorama này từng được triển lãm tại London vào năm 1792. Bề mặt của những bức tranh dạng panorama được cuốn trong một ống hình trụ và người ta kéo ra từ từ để bức tranh được dần hiển thị.

Năm 1881, họa sĩ người Hà Lan Hendrik Willem Mesdag đã tạo nên trường phái Panorama Mesdag với những ống hình trụ cuộn các bức tranh toàn cảnh với kích cỡ khổng lồ, cao 14m và dài có thể từ 40 -120 m. Thế kỷ XIX, có hai bức tranh panorama được coi là lớn nhất nhì thời kỳ này, đó là tranh mô tả trận chiến tại Atlanta với chiều cao gần 13m, dài 110m. Bức tranh được xác định lớn nhất là ở Wroclaw (Ba Lan) với kích cỡ 15m x 120m.

Thế rồi, với sự phát triển của khoa học công nghệ, người ta đã sáng tạo ra máy ảnh panorama.

Nếu máy ảnh thường chỉ có thể chụp ảnh với một góc 90 độ thì máy panorama có thể chụp với một góc 175 độ, 180 độ hoặc 360 độ. Trước một

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

không gian rộng lớn, máy ảnh thường bất lực trong việc ghi lại hình ảnh ở 1 góc rộng, nhưng máy panorama lại phát huy được tác dụng của nó.

Máy panorama thường được chụp bằng film phim dương bản(còn gọi là film Slide). Chụp xong bạn có thể xem phim là biết được ảnh sẽ được in ra như thế nào.

Chính vì góc ảnh của panorama rộng nên máy không có ống kính dài như máy thường. Ống kính của máy panorama hình vòng cung. Khi chụp ống kính sẽ quét từ trái sang phải , nên chúng ta phải sử dụng chân máy khi sử dụng.

Ảnh chụp panorama sẽ cho chúng ta những tấm ảnh với góc rộng lý tưởng. Hiệu quả nhất là ở thể loại ảnh Phong cảnh và Kiến trúc.

Ảnh panorama là ảnh được ghép từ những ảnh số chụp từng phần của một phong cảnh (những ảnh này có phần gối nhau) lại thành một ảnh toàn cảnh hoàn chỉnh.

Ta có thể hiểu một cách đơn giản Panorama là chế độ chụp ảnh khổ rộng bằng cách chụp nhiều tấm ảnh 4:3 liên tiếp, với thông tin của tấm ảnh trước được thể hiện một phần trong tấm ảnh sau, nhằm hỗ trợ người dùng. Sau đó với sự trợ giúp của phần mềm xử lý ảnh thì ta sẽ có được 1 tấm ảnh khổ rộng.

Hay nói một cách tổng quát panorama là một cách nhìn rộng của một vật chất trong không gian. Nó cho phép biểu thị một góc nhìn rộng của các bức tranh, bản vẽ đồ họa, nghệ thuật nhiếp ảnh, phim hoặc video, hay mô hình 3 chiều.

Ta hãy xem ví dụ sau:



Hình 1.9 Panorama

Đây là một bức ảnh được ghép từ 14 bức ảnh khác nhau. Mỗi bức ảnh được chụp ở mỗi góc độ khác nhau.

Có 4 kiểu ảnh panorama

1 - ảnh phẳng flat).

Nhược điểm là không giống thật nếu làm ảnh góc rộng. Cái ảnh chụp 360 độ mà trải thẳng ra thì không tương tự như chụp ở vị trí nào, cảnh ra sao. Ví dụ: con đường thẳng tắp, nhìn vào như đường conic

2- ảnh trụ (cylinder):

Chụp một vòng 360 độ để tạo ra panorama (tạm gọi là pano360), dán nó vào mặt hình trụ. Giả sử con mắt ở trục hình trụ, nhìn vào ảnh dán lên mặt hình trụ giống ta đứng và xoay một vòng ngắm cảnh.

3- ảnh cầu (sphere):

Ảnh được phủ lên khắp quả cầu tròn, tuyệt vời luôn, khi xem giống bạn đứng ở tâm điểm nhìn khắp xung quanh, lên trời, và cả xuống chân nữa chứ. Ảnh cầu có thể không phủ kín mặt cầu. Bạn tạo một pano360, sau đấy dán theo giao của hình cầu và mặt cắt qua tâm (như dán lên đường xích đạo ý). Nhìn qua tưởng giống kiểu 2, nhưng thực ra khác, vì đường sinh mặt trụ thẳng, còn mặt cầu cong. Đặc biệt nếu ảnh cao, hay chụp multi-row, sphere xem sẽ thật hơn cylinder.

4- ảnh hộp (cube):

Ảnh được trải lên một hình hộp. Nhìn có vẻ méo mó so với ảnh cầu. Nhưng thích hợp ví dụ cho: diễn tả lại đứng giữa phòng, ngắm một căn phòng hình hộp.

Mặt phẳng chiếu khi ghép ảnh: Hồi xưa ta nói các ảnh biến thành các mảnh ốp lên mặt cầu cho dễ hình dung. Thực tế thì các ảnh biến thành các mảnh ốp lên mặt phẳng chiếu. Ảnh này biến lên mặt phẳng chiếu sẽ nằm đối xứng quanh góc O. Vì các ảnh chồng lên nhau như một mạng lưới nên Yaw/Pitch/Roll của các ảnh khác được xác định.





Hình 1.10 Mặt phẳng chiếu khi ghép ảnh (minh họa) và Mặt phẳng chiếu khi thể hiện ảnh.

Mặt phẳng chiếu khi thể hiện ảnh: Xác định vị trí xong rồi, ghép các ảnh lại có ảnh panorama. Tóm lại, tùy phép chiếu và kích thước ảnh panorama, phần mềm khôn khéo chọn một mặt phẳng chiếu nào đó để thể hiện ảnh. Mặt phẳng chiếu khi chiếu ảnh: Là mặt phẳng chiếu cho ra sản phẩm cuối cùng. Có ảnh trên mặt phẳng chiếu khi thể hiện ảnh rồi, bạn chọn phép chiếu tùy thích, chọn mặt phẳng chiếu tùy thích để ra sản phẩm. Cho ngắn gọn, gọi mặt phẳng chiếu khi chiếu ảnh (thể hiện ảnh) là MẶT PHẪNG 2 (MẶT PHẪNG 1) và hệ lat/lon ứng với nó là HỆ 2 (HỆ 1).

Các bước trong ghép ảnh panorama

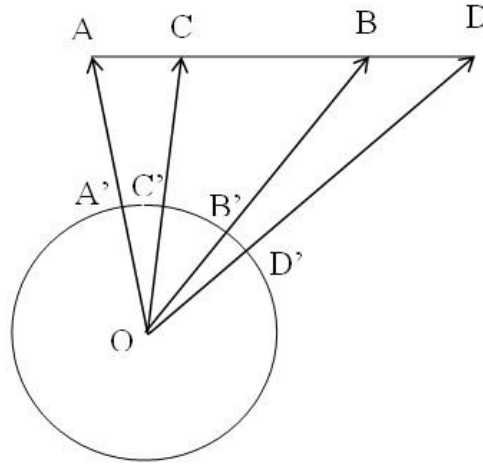
Cơ sở lý thuyết

Như chúng ta đã biết hình ảnh mà chúng ta nhìn thấy trong thế giới thực là hình ảnh 3 chiều. Hiện nay các thiết bị thu nhận ảnh từ thế giới thực đều cho hình ảnh dưới dạng 2 chiều.

Một số khái niệm:

Nodal point: Mỗi bức ảnh tương ứng với một phần của mặt cầu có tâm tại nodal point. Xét hai bức ảnh có cùng nodal point. Hai bức ảnh ứng với hai phần

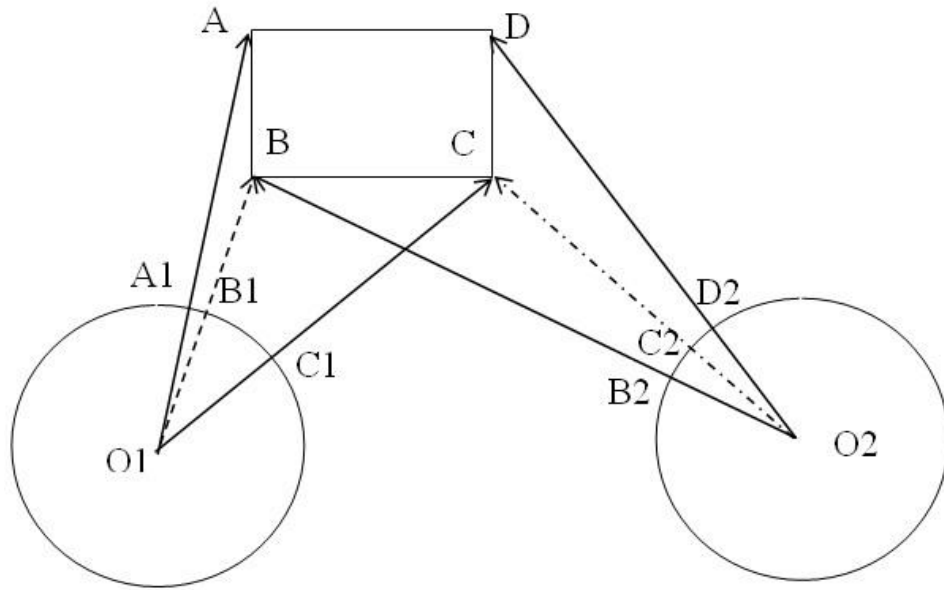
khác nhau của cùng một mặt cầu. Nếu một điểm xuất hiện trong cả hai bức ảnh thì nó tương ứng với cùng một điểm trên mặt cầu. Khi đó nếu biến 2 bức ảnh thành 2 miếng ốp lên mặt cầu thì nó sẽ chồng lên nhau tại những điểm xuất hiện trên cả 2 ảnh, vậy là vừa khít.



Hình 1.11 Ví dụ về nodal point

VD: máy ảnh với nodal point cố định tại O . Ảnh 1 chụp AB , ảnh 2 chụp CD . AB ứng $A'B'$ và CD ứng $C'D'$. CB là đoạn chung của 2 vật thì $C'D'$ cũng là đoạn chung của $A'B'$ và $C'D'$.(hình 1.11)

Parallax: Nodal point nằm ở 2 vị trí khác nhau thì có hai hình cầu khác nhau. Sẽ có những điểm của thế giới 3D nằm trên mặt cầu này mà không nằm trên mặt cầu kia và ngược lại. Có những đối tượng nằm trên cả 2 mặt cầu nhưng dạng hình học của chúng trên 2 mặt cầu lại khác nhau. Đây là hiện tượng *parallax*.



Hình 1.12 Mô tả hiện tượng *parallax*

VD: AB không xuất hiện trên mặt cầu tâm O_2 , CD không xuất hiện trên mặt cầu tâm O_1 . BC cùng xuất hiện trên 2 mặt cầu nhưng dạng hình học của $B1C1$ và $B2C2$ khác nhau.(hình 1.12)

Từ đây ta thấy việc tạo ảnh panorama chung quy có bốn bước:

- Biến đổi hình học: Biến các bức ảnh thành các miếng tương ứng trên mặt cầu để ta có thể chồng khít chúng lên nhau.(Sử dụng các phép biến đổi khác nhau để biến đổi hình dạng các bức ảnh sao cho có thể chồng khít chúng lên nhau).

- Các thuật toán so mẫu tìm các điểm giống nhau để chồng khít các bức ảnh (tìm control point).

- Trộn (blending): Dán các mảnh lại sao cho sự chuyển đổi màu sắc và ánh sáng ở các mối ghép phù hợp với nhau (nếu màu sắc giữa các ảnh không phù hợp thì có thể sử dụng các phương pháp thay đổi mức xám sao cho phù hợp).

- Đưa ra sản phẩm: Chiếu bức ảnh từ mặt cầu sang mặt phẳng theo các cách khác nhau (như làm bản đồ) thì được các bức ảnh panorama khác nhau.

Chương 2: GHÉP ẢNH DỰA TRÊN KỸ THUẬT NẮN CHỈNH HÌNH HỌC

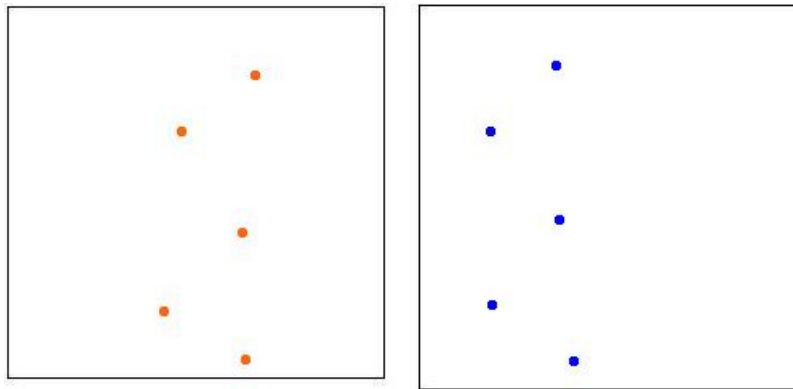
2.1 Một số vấn đề của bài toán

2.1.1 Xác định các cặp điểm đặc trưng

Từ hai ảnh đầu vào ta xác định tập các điểm điều khiển tương ứng trên mỗi vị trí cần ghép bằng cách đánh dấu chúng lại. Ta gọi các điểm này là các điểm đặc trưng, chúng chia ra làm hai tập điểm đặc trưng

$P = \{P_1, P_2, \dots, P_n\}$ được xác định trên ảnh thứ nhất.

$P' = \{P'_1, P'_2, \dots, P'_n\}$ được xác định trên ảnh thứ hai.



ảnh 1

ảnh 2

Hình 2.1 Ví dụ về hai tập điểm đặc trưng

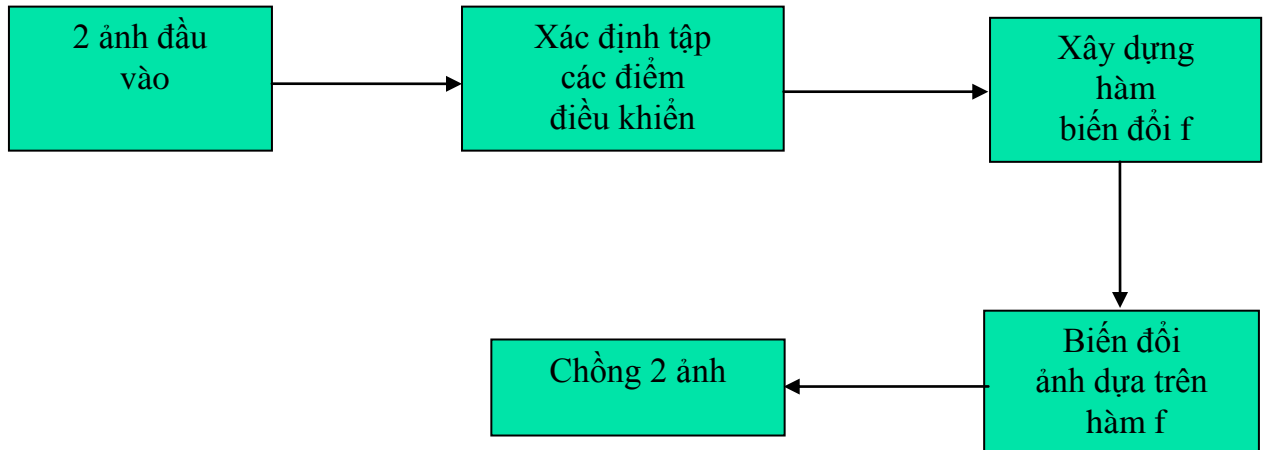
2.1.2 Xác định ảnh cần nắn chỉnh

Dựa trên các điểm đặc trưng ta xác định được độ biến dạng của một trong hai ảnh so với ảnh còn lại, ảnh này bị biến dạng theo nhiều chiều hướng khác nhau: Có thể bị tịnh tiến, co, giãn, méo mó... theo một tỷ lệ nào đó. Công việc cần làm là cần biến đổi và hiệu chỉnh sao cho khắc phục được độ biến dạng này đến mức tối thiểu. Do sự biến dạng này xảy ra ở một trong hai ảnh nên có thể chọn ảnh thứ nhất làm ảnh bị biến dạng và thực hiện nắn chỉnh qua ảnh này.

2.2 Nắn chỉnh hình dạng bức ảnh

Mục đích của quá trình nắn chỉnh là chuyển đổi các ảnh quét đang ở tọa độ hàng cột của các pixel về tọa độ trắc địa (tọa độ thực - hệ tọa độ địa lý hoặc tọa độ phẳng).

2.2.1 Sơ đồ thuật toán ghép ảnh dựa trên nắn chỉnh hình học



Hình 2.2 Sơ đồ thuật toán ghép ảnh dựa trên nắn chỉnh hình học

Như ta đã đề cập đến ở chương 1, để nắn chỉnh hình dạng của ảnh, chúng ta sử dụng các điểm điều khiển để xây dựng hàm f .

2.2.2 Xây dựng thuật toán nắn chỉnh dựa vào các cặp điểm đặc trưng

Nắn chỉnh ảnh thứ hai dựa trên các cặp điểm đặc trưng, mỗi cặp điểm đặc trưng gồm có: một điểm thuộc ảnh thứ nhất và một điểm thuộc ảnh thứ hai. Các điểm không phải điểm đặc trưng sẽ được ánh xạ dựa vào các điểm đặc trưng.

Từ hai tập điểm điều khiển này xây dựng một hàm f ánh xạ một điểm P_i thành điểm P_i' qua hàm f . Hàm f có dạng:

$$f: \mathbb{E}_i \mapsto f(P_i) \quad (2.1)$$

Sao cho thỏa mãn khoảng cách từ điểm P_i' đến điểm $f(P_i)$ là nhỏ nhất. Hay

$$\sum_{i=1}^n \|P_i' - f(P_i)\|^2 \rightarrow \min \quad (2.2)$$

Để giải quyết bài toán này giả sử ảnh thu nhận bị xoay tịnh tiến ta có phép biến đổi tuyến tính bậc nhất:

$$f(x, y) = (a_1x + b_1y + c_1, a_2x + b_2y + c_2) \quad (2.3)$$

Đặt

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

Để $\Phi \rightarrow \min$

$$\begin{cases} \frac{\partial \Phi}{\partial a_1} = 0 \\ \frac{\partial \Phi}{\partial b_1} = 0 \\ \frac{\partial \Phi}{\partial c_1} = 0 \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^n (a_1x_i + b_1y_i + c_1 - x_i')x_i = 0 \\ \sum_{i=1}^n (a_1x_i + b_1y_i + c_1 - x_i')y_i = 0 \\ \sum_{i=1}^n (a_1x_i + b_1y_i + c_1 - x_i') = 0 \end{cases}$$

$$\Leftrightarrow \begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i y_i & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i y_i & \sum_{i=1}^n y_i^2 & \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n y_i & n \end{pmatrix} \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i x_i' \\ \sum_{i=1}^n x_i y_i' \\ \sum_{i=1}^n x_i' \end{pmatrix} \quad (2.4)$$

Đặt

$$M_1 = n \sum_{i=1}^n (x_i y_i) - \sum_{i=1}^n x_i \sum_{i=1}^n y_i$$

$$M_2 = n \sum_{i=1}^n y_i^2 - \left(\sum_{i=1}^n y_i \right)^2$$

$$M_3 = n \sum_{i=1}^n (x_i' y_i) - \sum_{i=1}^n x_i' \sum_{i=1}^n y_i$$

$$N_1 = n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2$$

$$N_2 = n \sum_{i=1}^n (x_i y_i) - \sum_{i=1}^n x_i \sum_{i=1}^n y_i$$

$$N_3 = n \sum_{i=1}^n (x_i x'_i) - \sum_{i=1}^n x_i \sum_{i=1}^n x'_i$$

Giải hệ phương trình (2.4) với các biến số a_1, b_1, c_1 ta được :

$$a_1 = \frac{M_3 N_2 - M_2 N_3}{M_1 N_2 - M_2 N_1}$$

$$b_1 = \frac{N_3 M_1 - N_1 M_3}{M_1 N_2 - M_2 N_1}$$

$$c_1 = \frac{\sum_{i=1}^n x'_i - a_1 \sum_{i=1}^n x_i - b_1 \sum_{i=1}^n y_i}{n}$$

Tính toán tương tự ta tìm được a_2, b_2, c_2 .

Xác định được hàm kết quả f .

2.2.3. Biến đổi hình dạng bức ảnh dựa trên hàm biến đổi f

Sau khi xác định được hàm f ,thực hiện phép ánh xạ với mỗi điểm ảnh $I(i,j)$ xác được một điểm ảnh $I'(i',j')$ ở ảnh đích qua hàm f :

$$\begin{cases} j' = j \times a_1 + i \times b_1 + c_1 \\ i' = j \times a_2 + i \times b_2 + c_2 \end{cases} \quad (2.5)$$

Gán giá trị màu của điểm ảnh $I(i,j)$ cho $I'(i',j')$. Lần lượt thực hiện phép ánh xạ với các điểm ảnh còn lại thuộc ảnh thứ nhất ta sẽ thu được ảnh mới tương ứng .

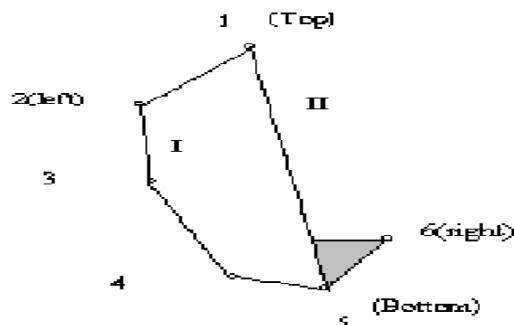
Ví dụ:



Hình 2.3 Ghép ảnh sau khi nắn chỉnh

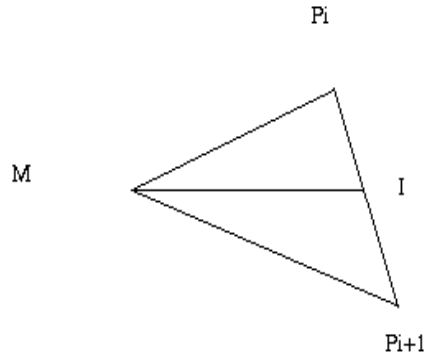
2.3 .Ghép ảnh sau khi nắn chỉnh

Phần chung (phần giao nhau) được xác định dựa trên một hình chữ nhật bao các điểm đặc trưng bởi các điểm left, top, right và bottom. Chia phần chung ra làm 2 phần I và II được xác định bởi hai điểm top và bottom.



Hình 2.4 Xác định phần chung của hai ảnh

Với một điểm $M(x,y)$ thuộc vùng ảnh (I) xác định được 2 điểm đặc trưng (p_i, p_{i+1}) sao cho $y_i < y < y_{i+1}$, xác định được một điểm $I(x_i, y_i)$ là giao điểm của đoạn thẳng đi qua M và song song với trục OX với đoạn thẳng $P_i P_{i+1}$.



Hình 2.5 Tìm điểm chung

Khi đó so sánh giá trị hoành độ của M và I , nếu $x < x_i$ thì M thuộc ảnh 1, ngược lại M thuộc ảnh 2.

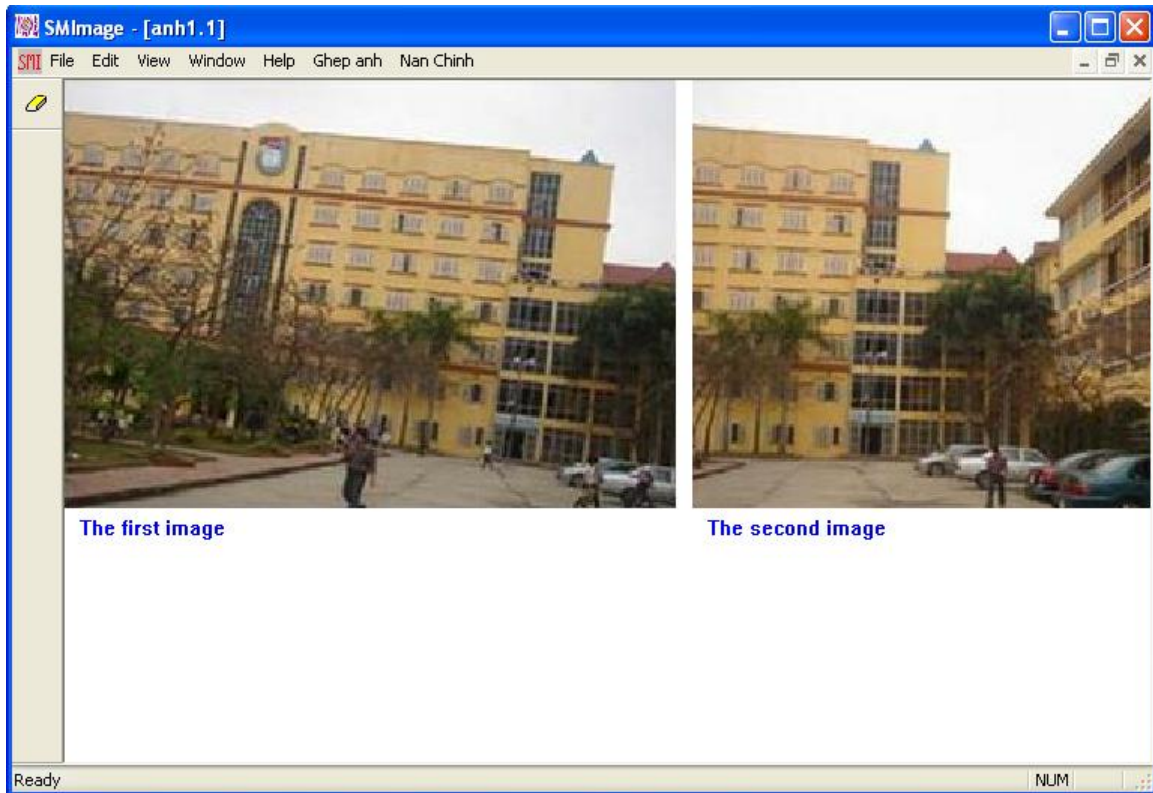
Trường hợp có cặp điểm đặc trưng thuộc vùng ảnh 2 thì duyệt ngược lại.

Duyệt các điểm M thuộc ảnh đích, nếu M thuộc ảnh 1 thì gán giá trị màu của ảnh 1 cho M , ngược lại gán giá trị màu của ảnh 2 cho M . Quá trình lặp lại khi duyệt hết các điểm ảnh thuộc phần chung nhau của 2 ảnh. Ta sẽ thu được ảnh ghép như mong muốn.

Chương 3: CHƯƠNG TRÌNH THỬ NGHIỆM

3.1. Giới thiệu chương trình

Chương trình ứng dụng kỹ thuật ghép ảnh minh họa cho các kết quả nghiên cứu của đề tài, nó được cài đặt bằng ngôn ngữ Visual C++ 6.0 và chạy trên môi trường Window. Đầu vào của chương trình là các file ảnh pex định dạng 256 màu.



Hình 3.1 Giao diện chương trình chính

3.2. Các chức năng của chương trình

Các nhóm chức năng chính:

-Nhóm chức năng File gồm các chức năng :

a.Mở file (Open ...):Nhóm chức năng này gồm hai chức năng :mở ảnh thứ nhất và ảnh thứ 2 để làm việc.

b.Đóng file :Đóng các file ảnh đang làm việc .

Một số nội dung về ảnh panorama và kỹ thuật ghép ảnh

c. Lưu file : Lưu file ảnh kết quả sau khi xử lý.

-Nhóm chức ghép ảnh gồm các chức năng :

a. Kích hoạt điểm điều khiển : tạo các điểm điều khiển trên các vị trí cần ghép trên 2 ảnh

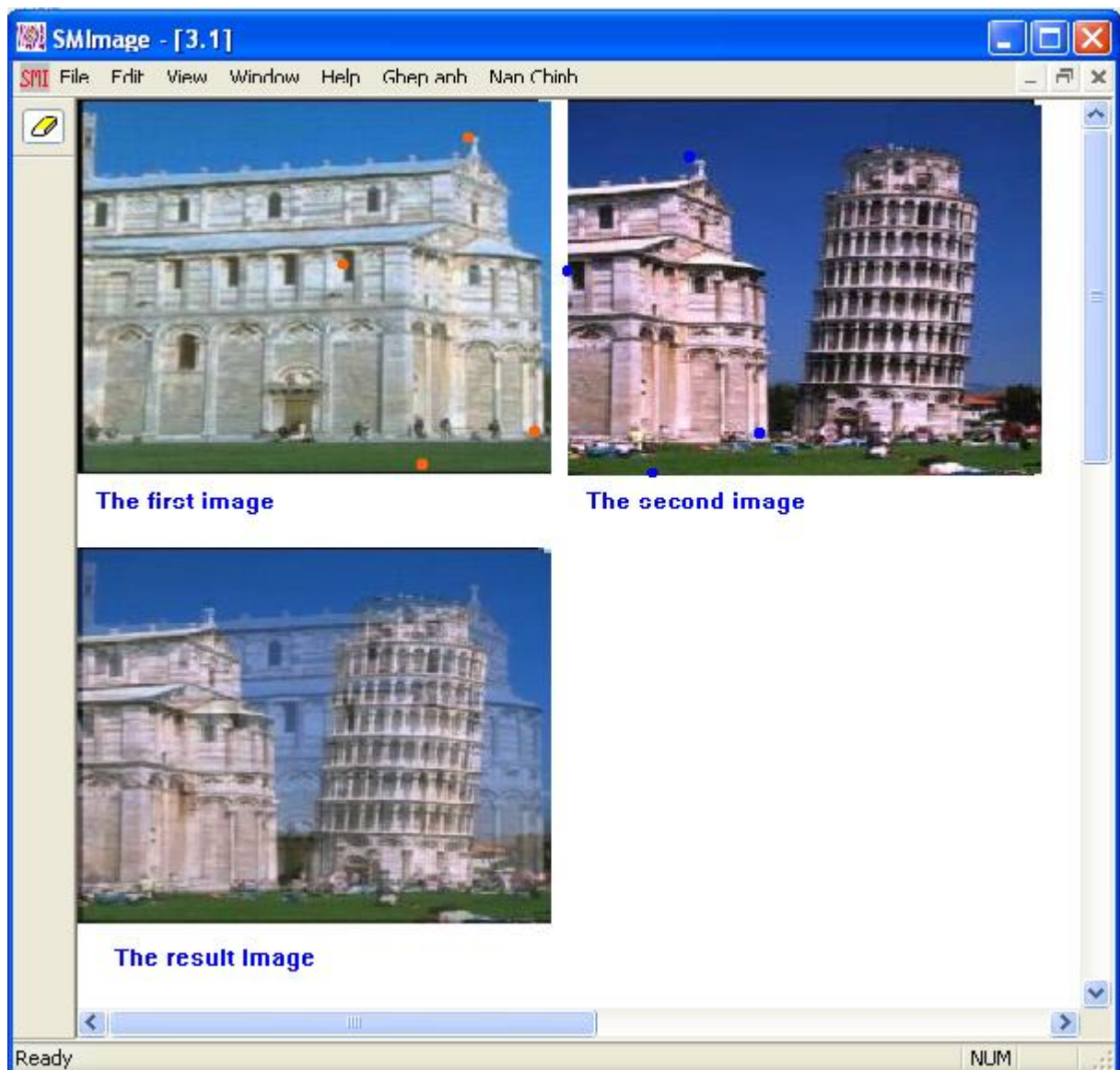
b. Ghép ảnh theo phương pháp đơn giản

c. Ghép ảnh theo phương pháp theo ngưỡng

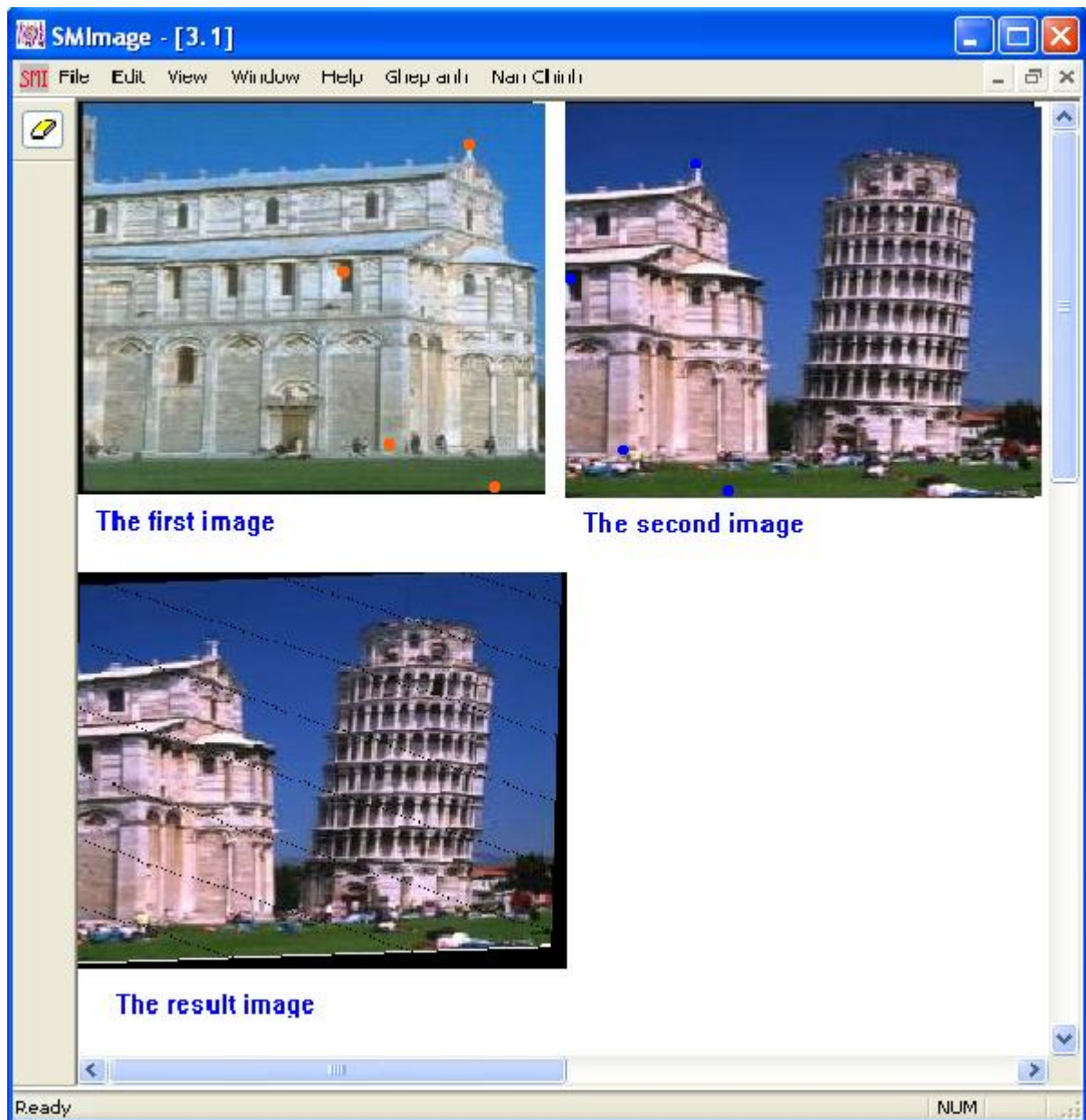
e. Chồng ảnh.

-Nhóm chức năng nắn chỉnh có : Nắn chỉnh hình học

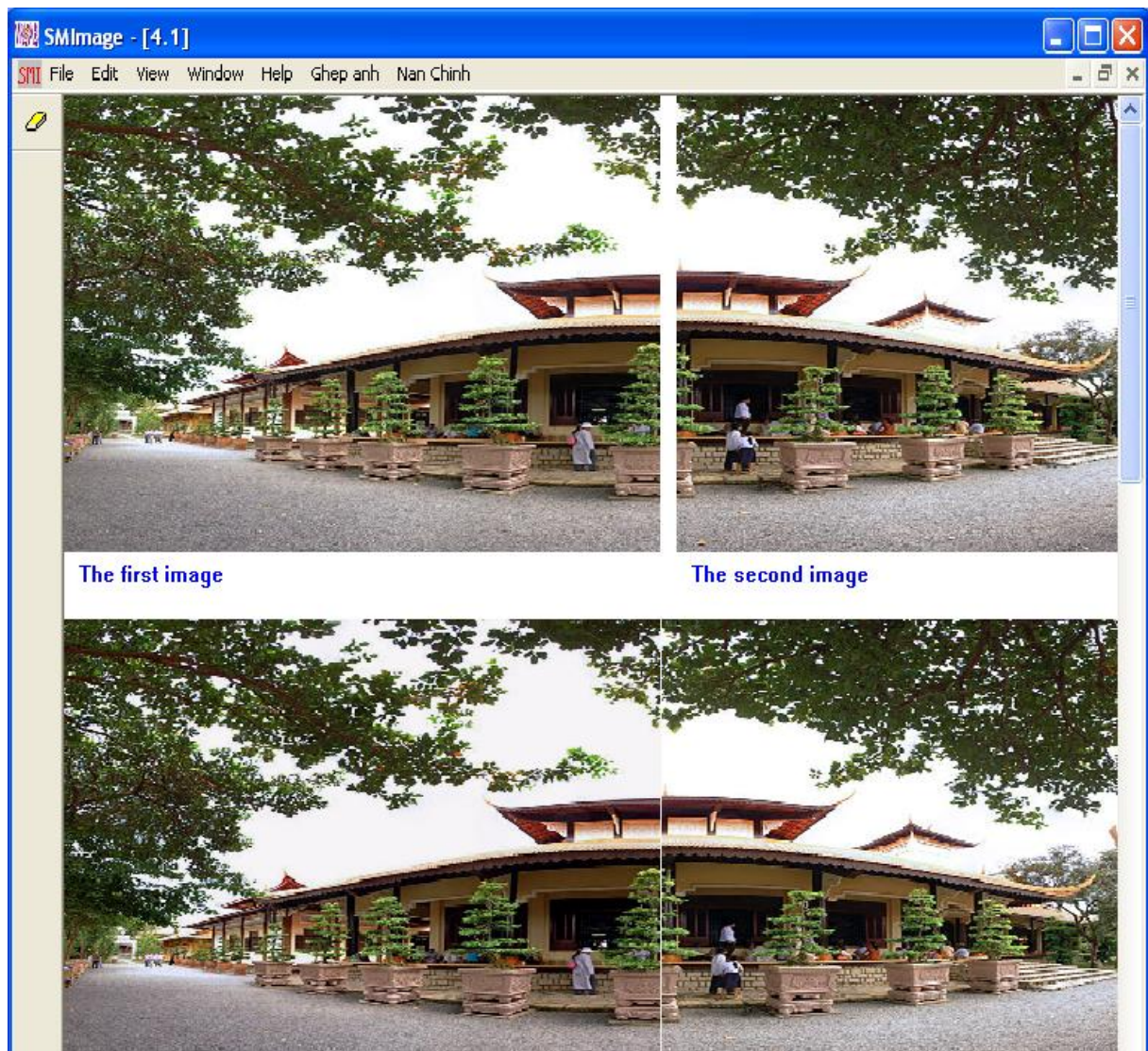
MỘT SỐ KẾT QUẢ MINH HỌA TRONG PHẦN MỀM SMImage



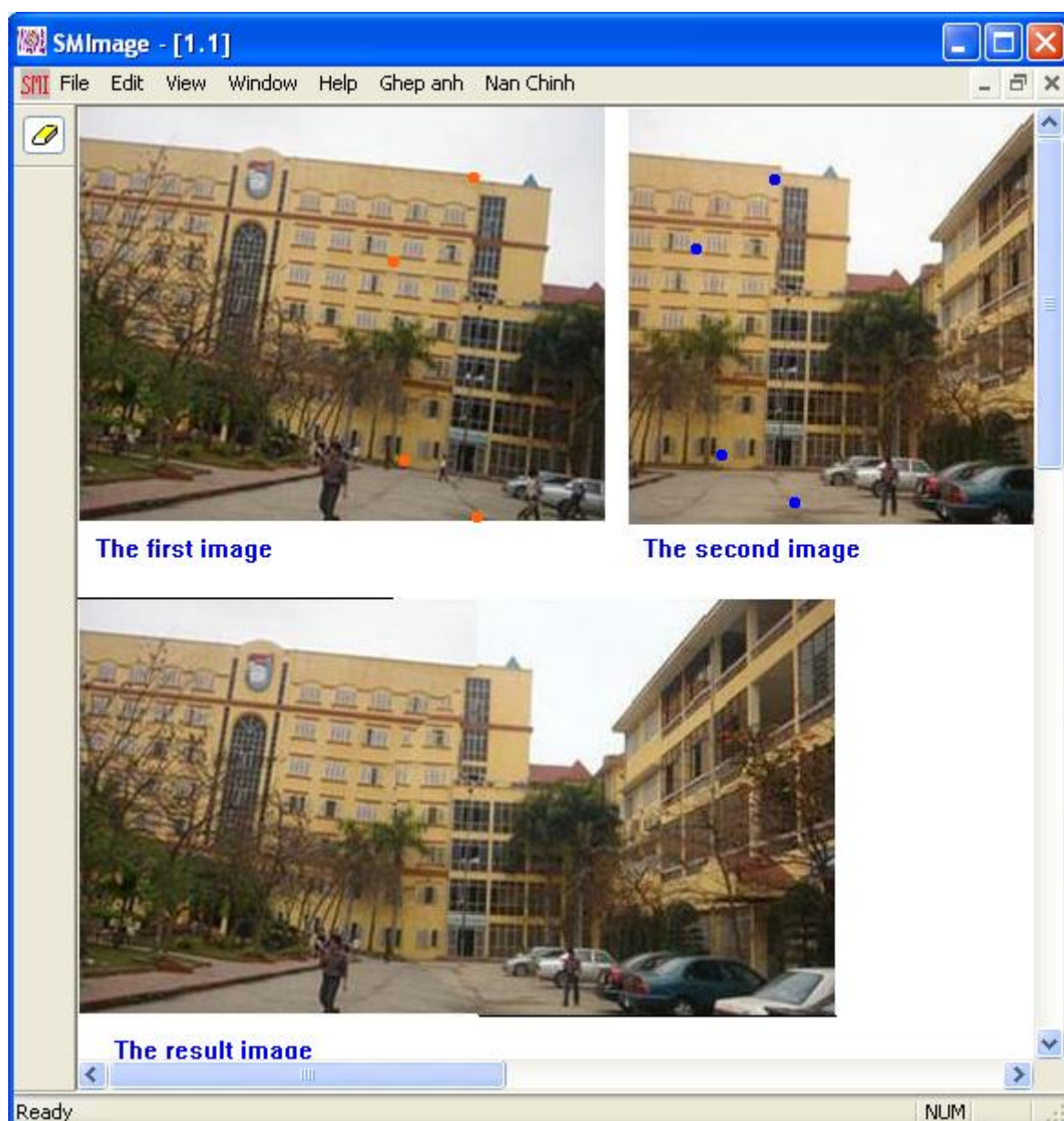
Hình 3.2 Chồng ảnh



Hình 3.3 Nắn chỉnh hình học



Hình3.4 Ghép ảnh đơn giản



Hình 3.5 Ghép theo ngưỡng

KẾT LUẬN

Ngày nay với tốc độ phát triển của ngành công nghiệp giải trí, đặc biệt là sự phát triển mạnh mẽ của nghề nhiếp ảnh, các nhiếp ảnh gia luôn mong muốn qua tấm ảnh của mình để quảng bá về đất nước hay phong cảnh của một địa điểm đẹp nào đấy. Với công nghệ ngày nay, đã có rất nhiều loại máy ảnh cho phép chụp ảnh toàn cảnh ra đời và cũng có rất nhiều phần mềm ghép ảnh chuyên nghiệp. Song nó đòi hỏi nhiếp ảnh gia phải có “tay nghề” tốt: Khi chụp ảnh trực của máy ảnh trên thân máy phải cố định, có nghĩa là người chụp sẽ phải tự xoay xung quanh máy ảnh để chụp các góc khác nhau; Các thông số kỹ thuật từ tiêu cự F, tốc độ chụp...phải giống nhau trong tất cả các kiểu ảnh chụp; Nếu dùng ống kính góc quá rộng sẽ gây ra lỗi méo mó hình ảnh; Ảnh chụp thường là trong cùng một thời gian để không bị sai lệch lớn về độ sáng của ảnh thì mới tạo ra được những bức ảnh ưng ý.

Nghiên cứu kỹ thuật ghép ảnh mở ra hướng phát triển mới trong công nghệ thu nhận ảnh, việc thu nhận ảnh của các đối tượng có kích thước lớn thường được thực hiện bởi nhiều phần, công việc tiếp theo sau khi thu nhận ảnh là cần ghép các phần ảnh này lại với nhau để thu được ảnh kết quả. Mặt khác các ảnh khi thu nhận thường bị biến dạng không giống nhau do rất nhiều nguyên nhân: Do góc độ thu nhận khác nhau, thời gian thu nhận khác nhau, thiết bị thu nhận không tốt. Báo cáo nghiên cứu một số kỹ thuật ghép ảnh: Kỹ thuật morphing, kỹ thuật nắn chỉnh hình học và biến đổi mức xám sẽ giải quyết được những sai lệch những thông tin biến dạng của ảnh. Đặc biệt báo cáo tập trung nghiên cứu sâu vào phương pháp ghép ảnh dựa trên kỹ thuật nắn chỉnh hình học.

Kỹ thuật nắn chỉnh hình bao gồm các phần chính sau:

- Nắn chỉnh hình học.

- Chồng ảnh.

- Ghép ảnh.

Hai kỹ thuật đều được cài đặt trong phần mềm merIMAGE.

Mặc dù đã có những kết quả chưa tốt đối với mục tiêu đề tài đặt ra. Nhưng do hạn chế về mặt thời gian thực hiện đề tài và đề tài tương đối rộng nên còn nhiều khía cạnh mà đề tài chưa thực hiện được. Vì thế hướng phát triển tiếp theo của đề tài còn rất mở, có thể nêu ra một số vấn đề lớn như sau:

Tìm hiểu một số phương pháp ghép ảnh trên những phần tương ứng của các đối tượng khác nhau dựa trên kỹ thuật nắn chỉnh hình học và biến đổi mức xám.

Tìm hiểu phương pháp ghép ảnh tự động dựa trên phương pháp morphing và phương pháp nắn chỉnh hình học.

Tài liệu tham khảo

Tiếng việt

[1] *Bài giảng Xử lý ảnh*, Đỗ Năng Toàn, Viện công nghệ thông tin Quốc Gia

[2] *Nhập môn Xử lý ảnh số*, Lương Mạnh Bá - Nguyễn Thanh Thủy, 1999

[3] *Xử lý ảnh* – (Trích dịch) *Two-dimensional SIGNAL and IMAGE PROCESSING*, jae s.lim , dịch giả Nguyễn Văn Ngọ, hiệu đính Nguyễn Viết Kính.

Tài liệu online

[8] <http://www.taybacgroup.com.vn/Default.aspx?g=posts&t=165&p=6>

PHỤ LỤC

Phụ lục A - PHẦN CODE CHÍNH CỦA CHƯƠNG TRÌNH

Ghép ảnh đơn giản

```
fipWinImage *normalMerge(fipWinImage *anh1,fipWinImage *anh2,int scale)
{
    int w1=anh1->getWidth();
    int h1=anh1->getHeight();
    int w2=anh2->getWidth();
    int h2=anh2->getHeight();

    int w=w1+w2;
    int h=(h1>=h2)?h1:h2;
    bool c=(h1>=h2)?true:false;
    long k;
    SCALE *color;
    color=Hesomau(anh1,anh2);
    SCALE *color1=new SCALE;
    SCALE *color2=new SCALE;
    color1=CalculatePer(scale,color,true);
    color2=CalculatePer(scale,color,false);
    anh1=HieuChinhMauOp(anh1,color2,true);
    anh2=HieuChinhMauOp(anh2,color1,false);
    if(anh1->getBitsPerPixel()!=8&&anh2->getBitsPerPixel()!=8)
    {
        fipWinImage *hDIB=new fipWinImage(FIT_BITMAP,w,h,anh1-
        >getBitsPerPixel());
        BYTE *pLine1=anh1->accessPixels();
        BYTE *pLine2=anh2->accessPixels();
```

```
BYTE *pLine=hDIB->accessPixels();
int i,j;
if(!c)
{
    for(i=0;i<w1;i++)
        for(j=0;j<h1;j++)

setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=w1;i<w;i++)
        for(j=0;j<h2;j++)
        {
            k=i-w1;

setColor(pLine,i,j,w,h,&getColor(pLine2,k,j,w2,h2));
        }
}
else
{
    for(i=0;i<w1;i++)
        for(j=0;j<h1;j++)

setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=w1;i<w;i++)
        for(j=0;j<h2;j++)
        {
            k=i-w1;

setColor(pLine,i,j,w,h,&getColor(pLine2,k,j,w2,h2));
        }
}
return hDIB;
}
return NULL;
```

}

Ghép ảnh theo ngưỡng

```
fipWinImage *mergeImg(fipWinImage *anh1,fipWinImage
*anh2,std::vector<POINT> p1,std::vector<POINT> p2,int scale)
{
    int i,j;
    POINT Left,Top,Right,Bottom;
    POINT Left2,Top2,Right2,Bottom2;
    int t,l,r,b;
    int i1;
    long L=1000,T=1000,R=0,B=0;
    for( i1=0;i1<(int)p1.size();i1++){
        if(p1.at(i1).x<L){                L=p1.at(i1).x;Left=p1.at(i1);l=i1;
        }
        if(p1.at(i1).y<T){ T=p1.at(i1).y;Top=p1.at(i1);t=i1;
        }
        if(p1.at(i1).x>R){ R=p1.at(i1).x;Right=p1.at(i1); r=i1;
        }
        if(p1.at(i1).y>B){
        B=p1.at(i1).y;Bottom=p1.at(i1);    b=i1;                }
    }
    L=10000;T=10000;R=0;B=0;
    for( i1=0;i1<(int)p2.size();i1++){
        if(p2.at(i1).x<L){L=p2.at(i1).x;Left2=p2.at(i1);}
        if(p2.at(i1).y<T){T=p2.at(i1).y;Top2=p2.at(i1);    }
        if(p2.at(i1).x>R){
        R=p2.at(i1).x;Right2=p2.at(i1);    }
        if(p2.at(i1).y>B){
        B=p2.at(i1).y;Bottom2=p2.at(i1);    }
    }
    long dl=p1.at(0).y-p2.at(0).y;
    int w1=anh1->getWidth();
```

```
int h1=anh1->getHeight();
int w2=anh2->getWidth();
int h2=anh2->getHeight();
int maxh=Max(h1,h2);

int w=w1+w2+10-Right2.x+Right.x;
int h=maxh+abs(dl);
if(anh1->getBitsPerPixel()!=anh2->getBitsPerPixel())
    return NULL;
SCALE *color;
color=Hesomau(anh1,anh2);
SCALE *color1=new SCALE;
SCALE *color2=new SCALE;
color1=CalculatePer(scale,color,true);
color2=CalculatePer(scale,color,false);
anh1=HieuChinhMauOp(anh1,color2,true);
anh2=HieuChinhMauOp(anh2,color1,false);

fipWinImage *hDIB=new fipWinImage(FIT_BITMAP,w,h,anh1-
>getBitsPerPixel());
BYTE *pLine=hDIB->accessPixels();
BYTE *pLine1=anh1->accessPixels();
BYTE *pLine2=anh2->accessPixels();
if(dl<0)
{
    for( i=0;i<=Left.x;i++)
        for(j=0;j<h1;j++)
            setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=Left.x;i<=Right.x;i++)
        for(j=h1-Top.y;j<=h1;j++)
            setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=Left.x;i<=Right.x;i++)
        for(j=0;j<=h1-Bottom.y;j++)
```

```
        setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    int dem=t;
    while(dem+1<b+1)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=h1-p1.at(dem+1).y;j<=h1-p1.at(dem).y;j++)
                if(timvung(p1.at(dem),p1.at(dem+1),i,h1-j))

        setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
        dem++;
    }
    for(i=(Right2.x-w1-10);i<w2;i++)
        for(j=0;j<=h2;j++)
            setColor(pLine,(i-Right2.x+w1+10+Right.x),(j-
dl),w,h,&getColor(pLine2,i,j,w2,h2));
    dem=t;
    while(dem<b)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=h2-p1.at(dem+1).y;j<=h2-p1.at(dem).y;j++)
                if(!timvung(p1.at(dem),p1.at(dem+1),i,h2-j))

        setColor(pLine,i,j,w,h,&getColor(pLine2,(i+Right2.x-w1-10-
Right.x),j+dl,w2,h2));
        dem++;
    }

    dem=0;
    while(dem<=t)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=h1-p1.at(dem).y;j<=h1-p1.at(dem+1).y;j++)
                if(!timvung(p1.at(dem),p1.at(dem+1),i,h1-j))
```

```
setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    dem++;
}
dem=b;
while(dem<(int)p1.size()-1)
{
    for(i=Left.x;i<=Right.x;i++)
        for(j=h1-p1.at(dem).y;j<=h1-p1.at(dem+1).y;j++)
            if(!timvung(p1.at(dem),p1.at(dem+1),i,h1-j))

setColor(pLine,i,j,w,h,&getColor(pLine1,i,j,w1,h1));
    dem++;
}
else
{
    for( i=0;i<=Left.x;i++)
        for(j=0;j<h1;j++)

setColor(pLine,i,j+dl,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=Left.x;i<=Right.x;i++)
        for(j=h1-Top.y;j<=h1;j++)

setColor(pLine,i,j+dl,w,h,&getColor(pLine1,i,j,w1,h1));
    for(i=Left.x;i<=Right.x;i++)
        for(j=0;j<=h1-Bottom.y;j++)

setColor(pLine,i,j+dl,w,h,&getColor(pLine1,i,j,w1,h1));
    int dem=t;
    while(dem+1<b+1)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=(h1-p1.at(dem+1).y);j<=(h1-p1.at(dem).y);j++)
```

```
        if(!timvung(p1.at(dem),p1.at(dem+1),i,h1-j))

setColor(pLine,i,j+dl,w,h,&getColor(pLine1,i,j,w1,h1));
        dem++;
    }

    for(i=(Right2.x-w1-10);i<=w2;i++)
        for(j=0;j<=h2;j++)
            setColor(pLine,(i-
Right2.x+w1+10+Right.x),j,w,h,&getColor(pLine2,i,j,w2,h2));
    dem=t;
    while(dem<b)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=h2-p1.at(dem+1).y;j<=h2-p1.at(dem).y;j++)
                if(!timvung(p1.at(dem),p1.at(dem+1),i,h2-j))
                    setColor(pLine,i,j+h-
h1,w,h,&getColor(pLine2,i+Right2.x-w1-10-Right.x,j+dl,w2,h2));
            dem++;
        }
    dem=0;
    while(dem<=t)
    {
        for(i=Left.x;i<=Right.x;i++)
            for(j=h1-p1.at(dem).y;j<=h1-p1.at(dem+1).y;j++)
                if(!timvung(p1.at(dem),p1.at(dem+1),i,h1-j))

setColor(pLine,i,j+dl,w,h,&getColor(pLine1,i,j,w1,h1));
            dem++;
        }
    dem=b;
    while(dem<(int)p1.size()-1)
    {
```

```
        for(i=Left.x;i<=Right.x;i++)
            for(j=h1-p1.at(dem).y;j<=h1-p1.at(dem+1).y;j++)
                if(!timvung(p1.at(dem),p1.at(dem+1),i,h1-j))

setColor(pLine,i,j+d1,w,h,&getColor(pLine1,i,j,w1,h1));
        dem++;
    }
}
return hDIB;
}
```

Nấn chỉnh hình học

```
fipWinImage *SampleResolved(fipWinImage *anh1,fipWinImage
*anh2,std::vector<POINT> p1,std::vector<POINT> p2)
{
    int w1=anh1->getWidth();
    int h1=anh1->getHeight();
    int w2=anh2->getWidth();
    int h2=anh2->getHeight();

    int u,v;
    float a1,b1,c1,a2,b2,c2;
    int i,j,x,y,xy,x1,y1,x2,y2,yy1,xx1,xy1,yx1;
    long X1,X2,X3,Y1,Y2,Y3,N1,N2,N3,M1,M2,M3;
    int count=(int)p1.size();

    int left1=10000,top1=10000,right1=0,bottom1=0;
    int left2=10000,top2=10000,right2=0,bottom2=0;
    std::vector<POINT>::iterator iter;
    for(iter=p1.begin();iter!=p1.end();iter++)
    {
```



```
    if((*iter).x<left1)left1=(*iter).x;
    if((*iter).y<top1) top1=(*iter).y;
    if((*iter).x>right1)right1=(*iter).x;
    if((*iter).y>bottom1)bottom1=(*iter).y;
}
for(iter=p2.begin();iter!=p2.end();iter++)
{
    if((*iter).x<left2)left2=(*iter).x;
    if((*iter).y<top2) top2=(*iter).y;
    if((*iter).x>right2)right2=(*iter).x;
    if((*iter).y>bottom2)bottom2=(*iter).y;
}
//chuan hoa khoi diem dieu khien thu 2
std::vector<POINT> np;
std::vector<POINT>::iterator iter1;
for(iter1=p2.begin();iter1!=p2.end();iter1++)
{
    POINT pt;
    pt.x=(*iter1).x-left2+left1;
    pt.y=(*iter1).y;
    np.push_back(pt);
}
i=0;
//x=tong (xi)
x=p1[0].x;
while(i<count-1){
    i++;
    x+=p1.at(i).x;
}
i=0;
//x2=tong(xi binh phuong)
x2=pow(p1.at(0).x,2);
while(i<count-1){
```

```
        i++;
        x2+=pow(p1.at(i).x,2);
    }
    i=0;
    //y=tong (yi)
    y=p1.at(0).y;
    while(i<count-1){
        i++;
        y+=p1.at(i).y;
    }
    i=0;
    //y2=tong (binhphuong(yi))
    y2=(long)pow(p1.at(0).y,2);
    while(i<count-1){
        i++;
        y2+=(long)pow(p1.at(i).y,2);
    }
    i=0;
    //xy=tong (xi*yi)
    xy=p1.at(0).x*p1.at(0).y;
    while(i<count-1){
        i++;
        xy+=p1.at(i).x*p1.at(i).y;
    }
    i=0;
    //x1=tong(x'i)
    x1=np.at(0).x;
    while(i<count-1){
        i++;
        x1+=np.at(i).x;
    }
    i=0;
    //y1=tong(y'i)
```

```
y1=np.at(0).y;
while(i<count-1){
    i++;
    y1+=np.at(i).y;
}

i=0;
//yy'=tong(y'i*yi)
yy1=np.at(0).y*p1.at(0).y;
while(i<count-1){
    i++;
    yy1+=np.at(i).y*p1.at(i).y;
}

i=0;
//xy1=tong (xi*y'i)
xy1=p1.at(0).x*np.at(0).y;
while(i<count-1){
    i++;
    xy1+=p1.at(i).x*np.at(i).y;
}

i=0;
//xx1=tong (xi*x'i)
xx1=p1.at(0).x*np.at(0).x;
while(i<count-1){
    i++;
    xx1+=p1.at(i).x*np.at(i).x;
}

i=0;
//yx1=tong(x'i*yi)
yx1=np.at(0).x*p1.at(0).y;
while(i<count-1){
    i++;
    yx1+=np.at(i).x*p1.at(i).y;
```

```
}
X1=(count*xy-x*y);
X2=(count*y2-pow(y,2));
X3=count*yy1-y*y1;
Y1=(count*x2-pow(x,2));
Y2=(count*xy-x*y);
Y3=(count*xy1-y1*x);
a2=((float)(X3*Y2-X2*Y3)/(float)(X1*Y2-X2*Y1));
b2=((float)(Y3*X1-Y1*X3)/(float)(X1*Y2-Y1*X2));
c2=((float)(y1-a2*x-b2*y)/(float)count);
/*-----*/
M1=(count*xy-x*y);
M2=(count*y2-pow(y,2));
M3=(count*yx1-y*x1);
N1=(count*x2-pow(x,2));
N2=(count*xy-x*y);
N3=(count*xx1-x1*x);
a1=((float)(M3*N2-M2*N3)/(float)(M1*N2-M2*N1));
b1=((float)(N3*M1-N1*M3)/(float)(M1*N2-N1*M2));
c1=((float)(x1-a1*x-b1*y)/(float)count);
//xay dung cac diem dau mut moi
int left=10000,top=10000,right=-10000,bottom=-10000;
POINT peak[4];
peak[0].x=c1;peak[0].y=c2;
peak[1].x=a1*w2+c1;peak[1].y=a2*w2+c2;
peak[2].x=b1*h2+c1;peak[2].y=b2*h2+c2;
peak[3].x=a1*w2+b1*h2+c1;peak[3].y=a2*w2+b2*h2+c2;
for(i=0;i<4;i++)
{
    if(peak[i].x<left)left    =peak[i].x;
    if(peak[i].y<top) top    =peak[i].y;
    if(peak[i].x>right)right=peak[i].x;
    if(peak[i].y>bottom)bottom=peak[i].y;
}
```

```
    }
    int w=right-left;
    int h=bottom-top;
    fipWinImage *hDIB=new fipWinImage(FIT_BITMAP,w,h,anh1->getBitsPerPixel());
    BYTE *pLine=hDIB->accessPixels();
    BYTE *pLine1=anh1->accessPixels();
    BYTE *pLine2=anh2->accessPixels();

    for(i=0;i<w2;i++)
        for(j=0;j<h2;j++)
        {
            u=i*a1+j*b1+c1;
            v=i*a2+j*b2+c2;
            setColor(pLine,u,v,w,h,&getColor(pLine2,i,j,w2,h2));
        }

    return hDIB;
}
```

Chồng ảnh

```
fipWinImage *ThresMerge(fipWinImage *anh1,fipWinImage *anh2,double
thres)
{
    int w=anh1->getWidth()>=anh2->getWidth()?anh1->getWidth():anh2->getWidth();
    int h=anh1->getHeight()>=anh2->getHeight()?anh1->getHeight():anh2->getHeight();
    fipWinImage *m_result=new fipWinImage(FIT_BITMAP,w,h,anh1->getBitsPerPixel());
    anh1->rescale(w,h,FILTER_BICUBIC);
    anh2->rescale(w,h,FILTER_BICUBIC);
```

```
BYTE *pLine=m_result->accessPixels();
BYTE *pLine1=anh1->accessPixels();
BYTE *pLine2=anh2->accessPixels();
int i,j;

for(i=0;i<w;i++)
    for(j=0;j<h;j++)
    {
        RGBQUAD rgb;

        rgb.rgbRed=(BYTE)(getColor(pLine1,i,j,w,h).rgbRed*thres+(1-
thres)*getColor(pLine2,i,j,w,h).rgbRed);

        rgb.rgbGreen=(BYTE)(getColor(pLine1,i,j,w,h).rgbGreen*thres+(1-
thres)*getColor(pLine2,i,j,w,h).rgbGreen);

        rgb.rgbBlue=(BYTE)(getColor(pLine1,i,j,w,h).rgbBlue*thres+(1-
thres)*getColor(pLine2,i,j,w,h).rgbBlue);
        setColor(pLine,i,j,w,h,&rgb);
    }
return m_result;
}
```

Phụ lục B: TÌM HIỂU VỀ CÁC DẠNG ẢNH

1. Định dạng ảnh IMG

Ảnh IMG là ảnh đen trắng, phần đầu của ảnh IMG có 16 byte chứa các thông tin sau:

6 byte đầu: dùng để đánh dấu định dạng ảnh. Giá trị của 6 byte này viết dưới dạng Hexa: 0x0001 0x0008 0x0001

2 byte tiếp theo: chứa độ dài mẫu tin. Đó là độ dài của dãy các byte kề liền nhau mà 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.

4 byte tiếp: mô tả kích cỡ pixel.

2 byte tiếp: số pixel trên một dòng ảnh.

2 byte cuối: số dòng ảnh trong ảnh.

Ảnh IMG được nén theo từng dòng, mỗi dòng bao gồm các gói (pack). Các dòng giống nhau cũng được nén thành một gói. Có 4 loại gói sau:

Loại 1: Gói các dòng giống nhau.

Quy cách gói tin này như sau: 0x00 0x00 0xFF Count. Ba byte đầu tiên cho biết số các dãy giống nhau, byte cuối cho biết số các dòng giống nhau.

Loại 2: Gói các dãy giống nhau.

Quy cách gói tin này như sau: 0x00 Count. Byte thứ hai cho biết số các dãy giống nhau được nén trong gói. Độ dài của dãy ghi ở đầu tệp.

Loại 3: Dãy các Pixel không giống nhau, không lặp lại và không nén được.

Quy cách gói tin này như sau: 0x80 Count. Byte thứ hai cho biết độ dài dãy các pixel không giống nhau không nén được.

Loại 4: Dãy các Pixel giống nhau.

Tuỳ theo các bit cao của byte đầu tiên được bật hay tắt. Nếu bit cao được bật (giá trị 1) thì đây là gói nén các byte 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 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 0. Tỷ lệ nén của kiểu định dạng này là khá cao.

2. Định dạng ảnh PCX

Định dạng ảnh PCX là một trong những định dạng ảnh cổ điển. Nó sử dụng phương pháp mã hoá loạt dài RLE (Run – Length – Encoded) để nén dữ liệu ảnh. Quá trình nén và giải nén được thực hiện trên từng dòng ảnh. Thực tế, phương pháp giải nén PCX kém hiệu quả hơn so với kiểu IMG. Tập PCX gồm 3 phần: đầu tệp (header), dữ liệu ảnh (Image data) và bảng màu mở rộng.

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

1 byte: chỉ ra kiểu định dạng. Nếu là PCX/PCC thì nó luôn có giá trị là 0Ah.

1 byte: chỉ ra version sử dụng để nén ảnh, có thể có các giá trị sau:

0: version 2.5.

2: version 2.8 với bảng màu.

3: version 2.8 hay 3.0 không có bảng màu.

5: version 3.0 có bảng màu.

1 byte: chỉ ra phương pháp mã hoá. Nếu là 0 thì mã hoá theo phương pháp BYTE PACKED, ngược lại là phương pháp RLE.

1 byte: Số bit cho một điểm ảnh plane.

1 word: tọa độ góc trái 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 bề rộng và bề cao của ảnh.

1 word: số điểm ảnh.

1 word: độ phân giải màn hình.

1 word.

48 byte: chia nó thành 16 nhóm, mỗi nhóm 3 byte. Mỗi nhóm này chứa thông tin về một thanh ghi màu. Như vậy ta có 16 thanh ghi màu.

1 byte: không dùng đến và luôn đặt là 0.

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

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 máy Macintosh. Hai loại này khác nhau rất lớn ở thứ tự các byte lưu trữ trong các số dài 2 hay 4 byte. Nếu trường này có giá trị là 4D4Dh thì đó là ảnh cho máy Macintosh, nếu là 4949h là của máy PC.

1 word: version. từ này luôn có giá trị là 42. 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 tới cấu trúc IFD 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.

Một IFD bao gồm:

2 byte: chứa các DE (Directory Entry).

12 byte là các DE xếp liên tiếp, mỗi DE chiếm 12 byte.

4 byte: chứa Offset trở tới IFD tiếp theo. Nếu đây là IFD cuối cùng thì trường này có giá trị 0.

Phần thứ 3: các DE: các DE có độ dài cố định gồm 12 byte và chia làm 4 phần:

2 byte: chỉ ra dấu hiệu mà tệp ảnh đã được xây dựng.

2 byte: kiểu dữ liệu của tham số ảnh. Có 5 kiểu tham số cơ bản:

1: BYTE (1 byte)

2: ASCII (1 byte)

3: SHORT (2 byte).

4: LONG (4 byte)

5: RATIONAL (8 byte)

4 byte: trường độ dài 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 liên quan tới dấu hiệu, tức là liên quan với DE không phải lưu trữ vật lý cùng với nó nằm ở một vị trí nào đó trong file.

Dữ liệu chứa trong tệp thường được tổ chức thành các nhóm dòng (cột) quét của dữ liệu ảnh. Cách tổ chức này làm giảm bộ nhớ cần thiết cho việc đọc tệp. Việc giải nén được thực hiện theo 4 kiểu khác nhau được lưu trữ trong byte dấu hiệu nén.