Chương 8
Khâu hình ảnh

trừu tượng Trong chương này, trước tiên chúng tôi giới thiệu nền ứng dụng và quy trình cơ bản của việc ghép ảnh, sau đó mô tả một số phương pháp ghép ảnh dựa trên vùng, các phương pháp ghép ảnh dựa trên điểm đặc trưng và kỹ thuật ghép ảnh video toàn cảnh.

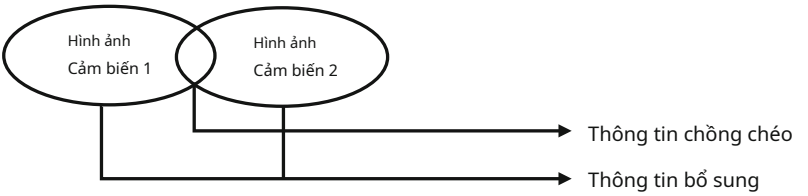
8.1 Giới thiệu

Trong thực tế, nó thường cần hình ảnh toàn cảnh có góc nhìn rộng và độ phân giải cao, nhưng kích thước của hình ảnh phụ thuộc vào hiệu suất của máy ảnh do giới hạn của thiết bị chụp ảnh. Do đó, một cách tiếp cận sử dụng phần mềm máy tính để ghép ảnh sau đó đã được đưa ra để tạo ảnh toàn cảnh. Ghép hình ảnh đề cập đến việc ghép một số hình ảnh có các phần chồng lên nhau thành một hình ảnh lớn, liền mạch và có độ phân giải cao. Nhân vật 8.1 hiển thị bản đồ phác thảo của khâu hình ảnh.

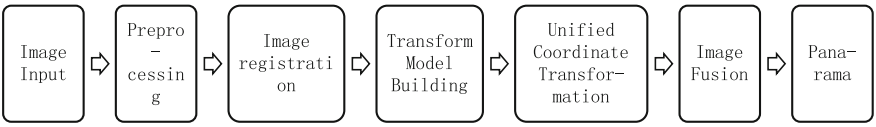
Nói chung, ghép ảnh chủ yếu bao gồm các bước sau:

(1) Xử lý trước hình ảnh. Nó chứa các hoạt động cơ bản của xử lý hình ảnh kỹ thuật số (chẳng hạn như khử nhiễu, trích xuất cạnh và xử lý biểu đồ), thiết lập các mẫu đối sánh hình ảnh, biến đổi hình ảnh (FT, WT, v.v.) và các hoạt động khác.
(2) Đăng ký hình ảnh. Nó áp dụng một số loại thuật toán đối sánh để xác định vị trí tương ứng của các mẫu hoặc điểm đặc trưng trong các hình ảnh ghép để xác định mối quan hệ chuyển đổi giữa hai hình ảnh.
(3) Xây dựng mô hình biến đổi. Một mô hình biến đổi toán học có thể được xây dựng giữa hai hình ảnh bằng cách tính toán các thông số của mô hình dựa trên sự tương ứng của các mẫu hình ảnh hoặc các đặc trưng.
(4) Phép biến đổi tọa độ Uni fi ed. Theo mô hình biến đổi toán học được xây dựng ở bước 3, ảnh được ghép sẽ được chuyển vào hệ tọa độ của ảnh tham chiếu để thực hiện phép biến đổi tọa độ đơn nhất.

S. Gong và cộng sự, *Xử lý hình ảnh và video nâng cao bằng MATLAB,*
Mô hình hóa và tối ưu hóa trong Khoa học và Công nghệ 12,
https://doi.org/10.1007/978-3-319-77223-3_8



Hình 8.1 Bản đồ phác thảo của khâu hình ảnh



Hình 8.2 Lưu đồ ghép ảnh

(5) Hợp nhất và tái tạo hình ảnh. Hợp nhất các phần chồng chéo của
hình ảnh được ghép thành một bức tranh toàn cảnh được tái tạo liền mạch và mượt mà.

Nhân vật 8.2 hiển thị sơ đồ cơ bản của việc ghép ảnh.

Đăng ký hình ảnh là chìa khóa cho các thuật toán ghép hình ảnh. Theo các phương pháp đăng ký ảnh khác nhau, các thuật toán ghép ảnh có thể được phân loại thành hai loại: ghép ảnh dựa trên vùng và ghép ảnh dựa trên các điểm đặc trưng.

8.2 Ghép ảnh dựa trên khu vực

Việc ghép ảnh dựa trên vùng bắt đầu từ việc so sánh các giá trị thang độ xám của một vùng trong ảnh sẽ được ghép với vùng trong ảnh được tham chiếu có cùng kích thước bằng các phương pháp bình phương nhỏ nhất và các phương pháp toán học khác. Từ các phép so sánh, chúng ta có thể đo lường mức độ giống nhau giữa các vùng chồng chéo trong hình ảnh được ghép và lấy phạm vi và vị trí của vùng chồng chéo trong hình ảnh sẽ được ghép để hoàn thành nhiệm vụ ghép ảnh. Chúng tôi cũng có thể chuyển đổi hình ảnh từ miền không gian thành miền tần số với FFT và vận hành đăng ký hình ảnh sau đó. Đối với các ảnh có độ dịch chuyển lớn, chúng ta có thể điều chỉnh việc xoay ảnh và sau đó thiết lập ánh xạ giữa hai ảnh. Khi lấy sự khác biệt giữa các giá trị thang độ xám của pixel ở hai vùng làm tiêu chí, cách tiếp cận đơn giản nhất là cộng trực tiếp sự khác biệt theo từng pixel. Một cách khác là tính toán

hệ số tương quan giữa các giá trị thang độ xám pixel của hai khu vực. Hệ số tương quan càng lớn thì mức độ khớp của hai hình ảnh càng cao và cách này cho thấy hiệu suất tốt hơn cũng như tỷ lệ thành công cao hơn. Ngày nay, các thuật toán ghép ảnh dựa trên vùng thường được sử dụng bao gồm Khớp tỷ lệ, Khớp dựa trên khối, Khớp dòng và Khớp lưới.

8.2.1 Khâu ghép ảnh dựa trên Khớp tỷ lệ

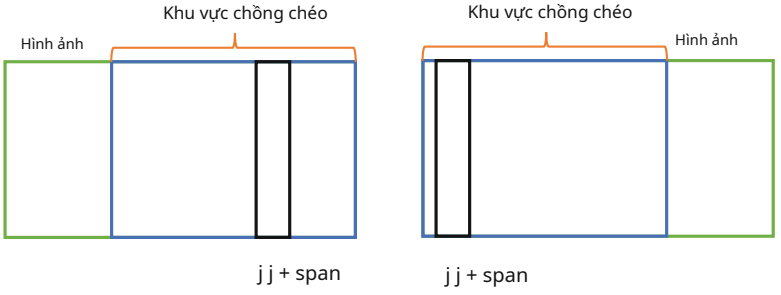
Việc ghép ảnh dựa trên đối sánh tỷ lệ, trước tiên chọn tỷ lệ hai cột pixel với khoảng cách nhất định giữa các phần được chồng lên nhau của hình ảnh làm mẫu [1]. Sau đó, tìm kiếm kết quả phù hợp nhất cho vùng chồng chéo trong hình ảnh thứ hai và tìm hai cột tương ứng với mẫu được lấy từ hình ảnh đầu tiên để hoàn thành việc ghép ảnh. Nhân vật 8,3 là một bản đồ phác thảo của thuật toán. Hình 1 là viết tắt của an ($W_1 \times H$) hình ảnh tính bằng pixel và Hình ảnh 2 là ($W_2 \times H$) một. W_1 và W_2 có thể bằng nhau hoặc không. Hình 1 nằm bên trái Hình 2. Một tình huống khác mà hình ảnh là chồng chéo theo chiều dọc sẽ không được thảo luận trong chương này vì chúng ta có thể xử lý nó theo cách tương tự.

Sau đây là các bước của thuật toán này:

(1) Chọn hai cột pixel với khoảng thời gian *nhịp* từ khu vực chồng chéo của Picture1, tính toán tỷ lệ pixel tương ứng làm mẫu *a.*

$$a(i, j) \quad \frac{P_1(tôi, j)}{P_1(i, j + span)}, \quad Tôi \in (1, H) \tag{8.1}$$

(2) Trong Hình 2, lần lượt chọn hai cột với khoảng cách khoảng từ cột đầu tiên, tỷ lệ các pixel tương ứng của nó được tính dưới dạng mẫu *b.*



Hình 8.3 Bản đồ phác thảo của việc chọn mẫu

$$b(i,j) \quad \frac{P_{21}(tôi,j)}{P_{22}(tôi,j)} \tag{8,2}$$

$P_{21}(tôi,j) \quad P_2(i,j), (i \in (1, H), j \in (1, W_2 - span)) P_2(i,j),$
$P_{22}(tôi,j) \quad (i \in (1, H), j \in (1, W_2 - nhịp))$

(3) Tính toán sự khác biệt giữa các mẫu *a* và *b* như mẫu *c.*

$$c(i,j) \quad (a(i,j) - b(i,j))_2, Tôi \in (1, H), j \in (1, W_2 - nhịp) \tag{8,3}$$

(4) *c* là một mảng hai chiều. Thêm từng vectơ cột vào một mảng khác gọi là tổng: *sum (j)* $\sum_{Tôi\,1}^{H} c(i,j).$ Giá trị của *sum (j)* tái tạo ra sự khác biệt của các cột đã chọn trong hai hình ảnh. Tọa độ cột của *sum (j)* tối thiểu giá trị *Tổng* min là trận đấu tốt nhất.

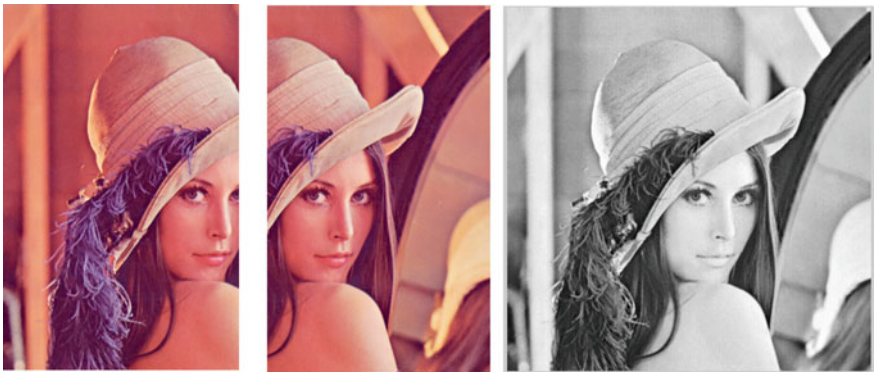
CHƯƠNG TRÌNH 8.1 là mã ghép ảnh dựa trên đối sánh tỷ lệ.

CHƯƠNG TRÌNH 8.1: Ghép ảnh dựa trên đối sánh tỷ lệ

```
thông thoáng;
clc;
A = imread ('lenna_left.jpg');% read Ảnh 1 A đại diện cho mảng pixel của Ảnh 1
B = imread ('lenna_right.jpg');% read Hình 2
[x1, y1] = size (A (:,:, 1));
% chuyển đổi thành hình ảnh thang độ xám , tính chiều dài và chiều cao của hình 1
[x2, y2] = size (B (:,:, 1));
A1 = double (A);
B1 = double (B);
sub_A = A1 (:, end-1) ./ A1 (:, end);% tính tỷ lệ của hai cột cuối cùng trong Hình 1 sub_D
= zeros (size (B, 2) -1,2);% xác định sub_D
cho y = 1: y2-1
    sub_B = B1 (:, y) ./ B1 (:, y + 1);% tính tỷ lệ của hai cột liền kề trong Hình 2 sub_C =
(sub_A-sub_B) '* (sub_A-sub_B);
% tính toán sự khác biệt giữa mẫu a và b, tính tổng của vectơ cột
    sub_D (y, 1) = y;
    sub_D (y, 2) = sub_C;% sub_D là mảng hai chiều
kết thúc
```

```
[ab] = sort (sub_D (:, 2));% sắp xếp tăng dần
row = b (1,:);% tọa độ của phần tử đầu tiên là khớp tốt nhất x3 =
x1;
y3 = y1-1 + y2-row;% chiều dài và chiều cao của hình ảnh được ghép
C = zeros (x3, y3);
cho i = 1: x3
    cho j = 1: y3
        nếu j <y1
            C (i, j) = A (i, j);
        khác
            C (i, j) = B (i, row + j-y1);
        kết thúc
    kết thúc
kết thúc% ghép hình ảnh
imwrite (C, 'picture3.bmp');
imshow (mat2gray (C));
```

Để xác định hiệu quả của việc ghép ảnh dựa trên đối sánh tỷ lệ, các thí nghiệm mô phỏng được thực hiện cho hai ảnh có vùng chồng lấn. Nhân vật 8,4 hiển thị kết quả.



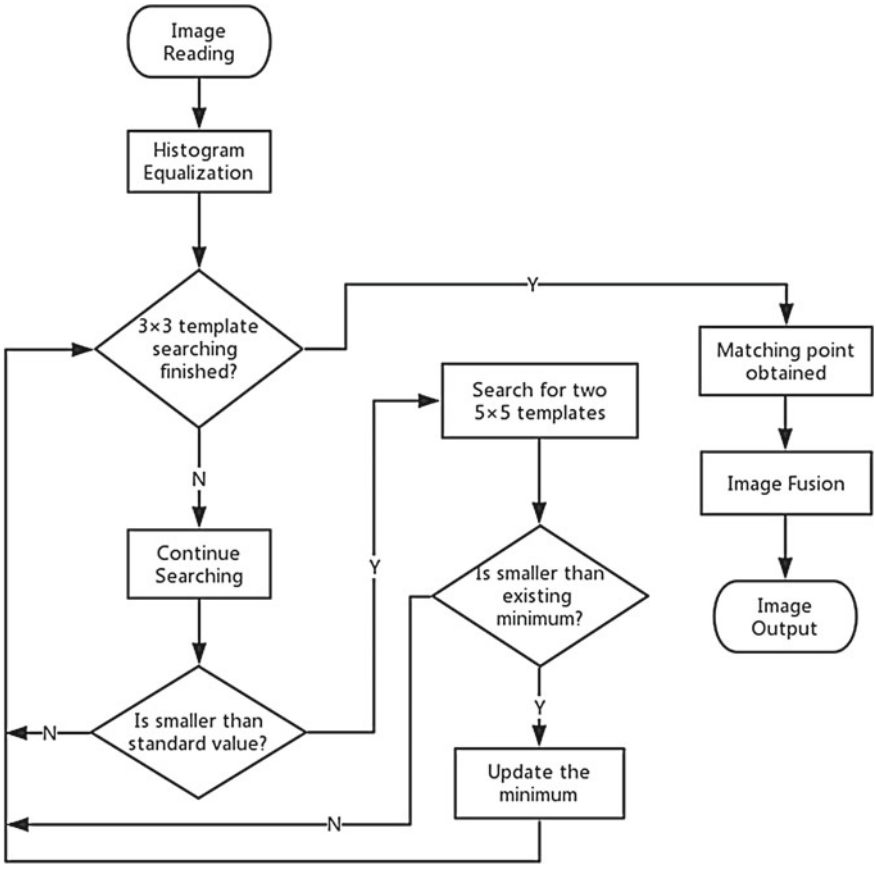
(a) Lenn a_ trái **(b) Lenn a_ ri g ht** **(c) O u t pu t**

Hình 8.4 Đầu vào và đầu ra của thử nghiệm

8.2.2 Khâu ghép ảnh dựa trên đặc điểm đường thẳng và mặt phẳng

Tính năng của đường và mặt phẳng ghép hình ảnh chủ yếu bao gồm: xử lý trước hình ảnh, tìm kiếm khối tính năng, ghép hình ảnh và kết hợp hình ảnh. Nhân vật 8.5 hiển thị biểu đồ fl owchart.

(1) Xử lý trước hình ảnh. Do độ chiếu sáng khác nhau nên rất dễ làm
lỗi khâu nếu ảnh thô thu được trực tiếp từ máy ảnh được khâu. Quá trình cân bằng biểu đồ là một cách hiệu quả để làm giảm bớt các tác động của ánh sáng. Sau khi áp dụng cân bằng biểu đồ cho hai hình ảnh được ghép, biểu đồ thang độ xám của hai hình ảnh được trải rộng thành tất cả các phạm vi mức xám và sự khác biệt về độ chiếu sáng trong các hình ảnh liền kề được giảm đi một cách đáng kể, điều này sẽ giúp việc ghép hình ảnh dễ dàng nhận ra.



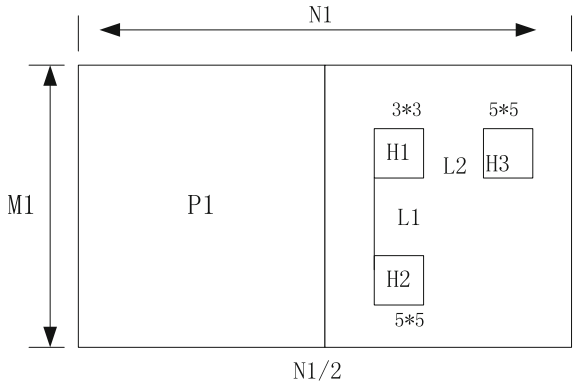
Hình 8.5 Sơ đồ ghép ảnh sử dụng tính năng đường kẻ và bề mặt

(2) Tìm kiếm khu vực tính năng. Chúng ta lấy P_1 như hình ảnh được tham chiếu và P_2 như hình ảnh được ghép. Kích thước của P_1 Là ($M_1 \times N_1$) và Hình 2 là ($M_2 \times N_2$). Thuật toán này sẽ chọn 3 mẫu tính năng nhỏ trong P_1 để phù hợp. Trước hết, giới hạn vùng dùng để chọn mẫu là từ dòng 1 đến dòng M_1 và cột $N_{1/2}$ đến cột N_1 trong P_1. Trước tiên, chúng tôi sẽ chọn một 3 × 3 mẫu nhỏ có tên H_1 trong khu vực này. Sau đó, theo các tính năng của hình ảnh, chúng tôi chọn hai 5 còn lại × 5 mẫu có tên H_2 và H_3 tương ứng trong khu vực. Như thể hiện trong Hình. 8.6 , một nhóm mẫu tính năng bao gồm 3 mẫu nhỏ được tạo thành. (Lưu ý: Chúng tôi giả sử rằng H_2 và H_1 ở cùng một mức theo hướng ngang và như vậy H_3 và H_1 theo phương thẳng đứng và khoảng cách từ Hl tương ứng là Ll và L2.) Chúng tôi áp dụng phương pháp tính toán các giá trị phương sai của pixel trong các mẫu khi chọn mẫu tính năng. Chúng tôi chọn mẫu có tổng phương sai tối đa làm mẫu tiêu chuẩn vì các đặc điểm chi tiết trong hình ảnh được xác định bởi các đặc điểm cạnh hoặc trong các điểm chỉnh sửa của giá trị thang độ xám. Trong đó tổng phương sai lớn nhất bằng với vị trí của các cạnh hoặc tại fl điểm tạo ra fl hiển thị nhiều nhất trong các đường cong của mức xám. Chúng tôi có thể đo lường có bao nhiêu tính năng chi tiết trong một mẫu bằng các giá trị phương sai pixel sumof. Các chi tiết hơn và thông tin kết cấu P_1 chứa, càng dễ dàng tìm thấy các khu vực tương tự trong P_2. Dưới đây là các phương trình để chọn mẫu tính năng.

$$S(x, y) \quad \sum_{Tôi\,1}^{M} \sum_{j\,1}^{M} Tôi(tôi, j) - \omega \,/_2 \tag{8,4}$$

$$S_{kết\ quả}(x, y) \quad MAX(S(x, y)) \tag{8,5}$$

Trong Eq. (8,4), $Tôi(tôi, j)$ đại diện cho giá trị pixel và ω là viết tắt của giá trị màu xám trung bình của mẫu. Khi đặt M là 3, chúng ta có thể thu được 3 tốt nhất × 3 tính năng



Hình 8.6 Trích xuất nhóm mẫu tính năng

mẫu thông qua Eqs. 8,4 và 8.5 . Lặp lại phép tính hai lần để được hai số còn lại là 5 × 5 mẫu. Bằng cách này, 3 mẫu tính năng có chi tiết tốt nhất sẽ được trích xuất. Ghi lại thông tin khoảng cách xung quanh chúng để tạo nhóm mẫu tính năng.

Sau khi trích xuất các nhóm mẫu tính năng thích hợp, tìm kiếm từ trên xuống dưới cùng và từ trái sang phải với 3 × 3 mẫu Fl trong $P_{2,}$ và tính toán pixel sự khác biệt giữa F_1 và H_1 từng cái một. Hàm MSE được sử dụng để xác định hàm khác biệt. Đây là định nghĩa:

$$S_{3\times3}(x, y) \quad \frac{1}{9}\sum_{Tôi,j}^{3} [F_1(tôi, j) - H_1(tôi, j)]^2 \qquad (8,6)$$

Ở đâu $F_1(tôi, j)$ là viết tắt của giá trị màu xám của các pixel tương ứng trong F_1 và như vậy $H_1(tôi, j)$ cho H_1.

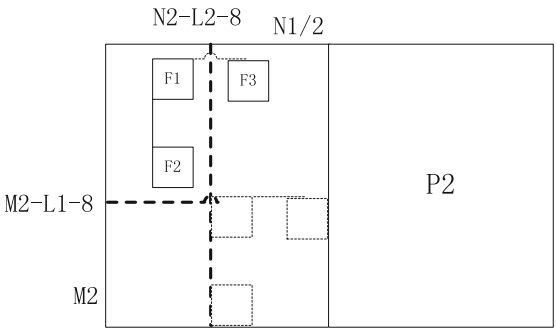
Kết quả thí nghiệm chỉ ra rằng khi sự chênh lệch của cường độ ánh sáng trong hình ảnh có kích thước nhỏ, việc đặt giá trị tiêu chuẩn của hàm khác biệt thành 30 là một giá trị thích hợp nếu mẫu tính năng đã chọn là 3 × 3. Khi giá trị tiêu chuẩn đã chọn lớn hơn 30, số lượng mẫu thiết lập các điều kiện đang tăng lên nhanh chóng, điều này sẽ dẫn đến thời gian tính toán lâu hơn. Nếu giá trị tiêu chuẩn đã chọn nhỏ hơn 30, thì sẽ rất khó để tìm các mẫu đáp ứng tiêu chí khi có sự giao thoa cao giữa hai hình ảnh và gây ra lỗi thuật toán.

Khi sự khác biệt được tính toán bởi Eq. 8.6 lớn hơn giá trị tiêu chuẩn 30, sự khác biệt giữa hai tiêu bản được coi là rất nhỏ và hầu như không thể là một khu vực phù hợp. Do đó, chúng ta nên loại bỏ mẫu và tiếp tục tính toán tiếp theo. Khi sự khác biệt nhỏ hơn 30, được coi là mẫu rất có thể là một mẫu phù hợp. Tuy nhiên, 3 × 3 mẫu quá nhỏ để định vị chính xác. Vì vậy, theo thông tin khoảng cách được ghi lại giữa các mẫu 3 × 3 và 5 × 5 trong một nhóm mẫu, chúng ta có thể tìm hai 5 tương ứng × 5 mẫu ở cùng một vị trí

thông tin khoảng cách xung quanh 3 × Mẫu 3 sở thích trong $P_{2.}$ Sau đó, tính toán sự khác biệt giữa 5 × 5 mẫu tương ứng với mẫu tính năng nhóm. Hàm tổng của các chênh lệch được định nghĩa như bên dưới.

$$S_{5\times5}(x, y) \quad \frac{1}{25}\sum_{Tôi,j}^{5} [F_2(tôi, j) - H_2(tôi, j)]^2 + \frac{1}{25}\sum_{Tôi,j}^{5} [F_3(tôi, j) - H_3(tôi, j)]^2 \qquad (8,7)$$

Tính toán tất cả F_1 có sự khác biệt nhỏ hơn 30 đến Eq. 8.7 và lưu mọi kết quả. Bên cạnh đó, lưu các giá trị ngang và sắp xếp của phần trên bên trái nhất các điểm pixel góc của mẫu Fl tại cùng một thời điểm. Cuối cùng, sử dụng Eq. 8.8



Hình 8.7 Đối sánh ngang của mẫu đối tượng địa lý

để có được tổng chênh lệch nhỏ nhất, các giá trị ngang và sắp xếp của góc trên bên trái của mẫu Fl tương ứng với tổng chênh lệch nhỏ nhất là tọa độ của các điểm phù hợp thu được.

$$S_{find}(x, y) \quad MIN(S_{5\times5}(x, y)) \tag{8,8}$$

Thuật toán này thu được một cách chính xác mẫu phù hợp nhất bằng cách chọn lọc các mẫu hai lần. Đầu tiên đo mức độ liên quan của 3 nhỏ × 3 mẫu và lưu từng mẫu với độ tương quan cao. Sau đó tính giá trị liên quan của hai 5 × 5 mẫu tương ứng với các mẫu đã lưu và thu thập 5 mẫu × 5 mẫu có mức độ liên quan cao nhất. Điều này có nghĩa là phải lọc lại các mẫu thu được ở bước đầu tiên để có mẫu phù hợp nhất (Hình. 8.7).

(3) Ghép hình ảnh và kết hợp hình ảnh. Sau khi kết thúc điểm khớp, sự chồng chéo đơn giản sẽ gây ra các đường viền rõ ràng trong ảnh là điều không mong muốn. Cần phải có một quá trình chuyển đổi mượt mà cho việc ghép ảnh để loại bỏ những điểm không đáng có như vậy. Thuật toán trong và ngoài được phân cấp có thể thu được hình ảnh liền mạch, nhưng trong thời kỳ hợp nhất hình ảnh, các vùng chồng chéo của hai hình ảnh được chồng lên nhau bằng trọng số tuyến tính và điều này chắc chắn làm cho các vùng chồng chéo bị mờ hơn so với hình ảnh gốc. Do đó, chúng tôi sử dụng phản ứng tổng hợp Gauss để thay thế. Bằng cách thực hiện sự thay đổi của hệ số gradient từ 0 đến 1 tuân theo đặc tính phân phối của đường cong Gauss một cách xấp xỉ, và đạt được sự chuyển đổi nhanh chóng giữa hai hình ảnh. Khu vực chồng chéo của kết quả đường khâu rõ ràng hơn so với cách tiếp cận không kéo ra phân cấp.

Chương trình Matlab của thuật toán nêu trên được trình bày như sau:

CHƯƠNG TRÌNH 8.2: Khâu hình ảnh sử dụng tính năng đường và bề mặt

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A = imread ('1.bmp');
subplot (1,2,1), imshow (A)
title ('Hình ảnh nguồn A')
B = imread ('2.bmp');
subplot (1,2,2), imshow (B)
title ('Hình ảnh nguồn B')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[cao, rộng] = size (A);
A1 = double (A);
B1 = double (B);
sub_A = A1 (cao / 2-39: cao / 2,3 * rộng / 4: 3 * rộng / 4 + 39);
% sub_A = A1 (high / 2-39: high / 2, end-39: end);
sub_B1 = B1 (11: 50,11: 50);
mod1 = sub_A-sub_B1;
mat1 = sum (sum (mod1. * mod1));
mat_best = mat1;
cho x1 = 1: 40: wid-40
    cho y1 = 1: 40: cao-40
        sub_B = B1 (y1: y1 + 39, x1: x1 + 39);
        mod = sub_A-sub_B;
        mat = sum (sum (mod. * mod));
        nếu mat <= mat_best
            mat_best = mat;
            xx = x1;
            yy = y1;
        kết thúc
    kết thúc
kết thúc
x = xx;          % cài đặt tùy chỉnh
y = yy;
cho x2 = xx-40: xx
    cho y2 = yy-20: yy + 80
        sub_B2 = B1 (y2: y2 + 39, x2: x2 + 39);
        mod2 = sub_A-sub_B2;
        mat2 = sum (sum (mod2. * mod2));
        nếu mat2 <= mat_best
            mat_best = mat2;
            x = x2;
            y = y2;
        kết thúc
```

```
        kết thúc
    kết thúc
% nhân vật
% bản đồ màu (xám);
% subplot (2,1,1); imagesc (sub_A)
% subplot (2,1,2); imagesc (sub_B2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x = 55; y = 1;
% ************************
  nếu y == 0
      AA = A (1: cao-1,1: 3 * rộng / 4);
      BB = B (1: cao-1,1: wid-1);
khác nếu y> = cao / 2
            AA = A (1: cao-y, 1: 3 * rộng / 4);
            BB = B (y: high-1, x: wid-1);
            khác
            AA = A (y: high-1,1: 3 * wid / 4);
            BB = B (1: cao-y, x: wid-20);
            kết thúc
kết thúc
C = [AABB];
% imwrite (C, 'Hình ảnh ghép trực tiếp 34.bmp');
figure, imshow (C)
title ('Hình ảnh được ghép trực tiếp') nếu y
== 0
      A2 = A (1: cao-1, :);
      B2 = B (1: cao, :);
khác nếu y> = 50
            A2 = A (1: cao-y, :);
            B2 = B (y: cao-1, :);
      khác
            A2 = A (y: cao-1, :);
            B2 = B (1: cao-y, :);
      kết thúc
kết thúc
x = 140;% ************************
[high2, wid2] = size (A2);
a1 = A2 (1: high2, wid-x + 1: wid);
b1 = B2 (1: cao2,1: x);
a = double (a1);
```

```
b = double (b1);
d1O = linspace (1,0, x);
d = 1: cao2;
d1 = d1O ';
[X1, y1] = meshgrid (d1, d);
im1 = a. * X1;% *************************
d20 = linspace (0,1, x);
d2 = d20 ';
[X2, y2] = meshgrid (d2, d);
im2 = b. * X2;
im11 = uint8 (im1);
im22 = uint8 (im2);
im3 = imadd (im11, im22);
figure, imshow (im3)
title ('Vùng kết hợp có phân loại')
 a_b = imadd (im11, im22);
 aa = A2 (1: high2,1: wid-x);
 bb = B2 (1: high2, x: wid);
 % D2 = [aa a_b bb];
 D2 = [aa (:, 1: wid2-x) a_b bb];% *************************
 imwrite (D2, '6.bmp');
 figure, imshow (D2)
 title ('Hình ảnh đã được phân loại');
```

Nhân vật 8.8 hiển thị kết quả.

(a) Vì thế u rce tôi ag e 1 **(b) Vì vậy u rce tôi ag e 2** **(c) F trong a lo u t pu t**

Hình 8.8 Kết quả của khâu

8.2.3 Khâu ghép ảnh dựa trên FFT

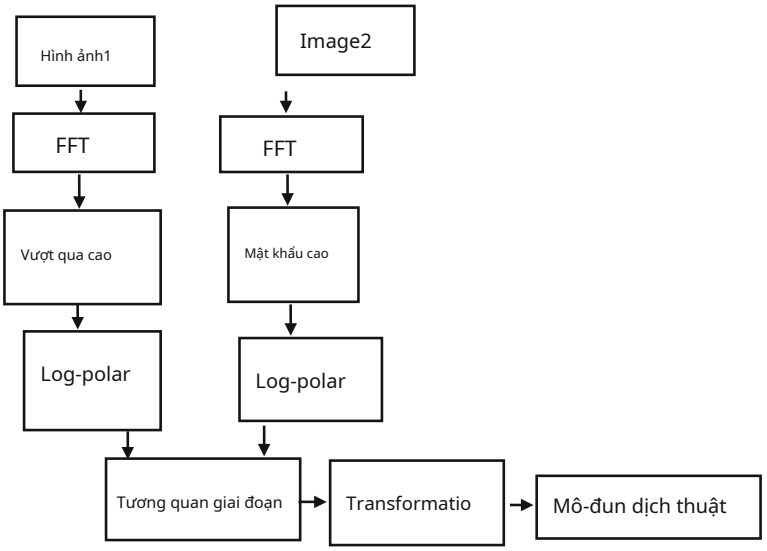
Toàn cảnh đề cập đến sự hình thành của chế độ xem đầy đủ, hình ảnh 360 ° có độ phân giải cao thông qua xử lý hình ảnh. Nó là một sự tái tạo tích hợp đối với các ống ngắm nhìn xung quanh và nó có thể hiển thị thông tin tích hợp tốt hơn về môi trường xung quanh.

Khâu hình ảnh dựa trên FFT lần đầu tiên chuyển đổi hình ảnh sang miền tần số và tính toán lượng quay và độ lệch theo phổ công suất chéo pha của nó. Sau đó đặt lại tọa độ của hình ảnh và áp dụng chuyển động. Cuối cùng, các hình ảnh được ghép lại với nhau. Khi ghép ảnh toàn cảnh 360 °, cần chuyển đổi độ dài tiêu cự và hình chiếu trước khi tính toán với các giai đoạn. Nhân vật 8.9
trình bày sơ đồ ghép ảnh dựa trên FFT.

Phương pháp áp dụng để ghép ảnh toàn cảnh hình trụ có thể được chia thành 3 phần:

(1) Xây dựng một hàm với tương quan pha của miền tần số. Func này-
tion sẽ thực hiện phép biến đổi 2-DFourier trên hai hình ảnh đầu vào và trả về các giá trị offset giữa hai hình ảnh liền kề.
(2) Tính toán các giá trị tiêu cự của một bộ ảnh thực tế 360 ° và áp dụng phép chiếu hình trụ cho chuỗi ảnh.
(3) Gọi lần lượt hàm trong phần 1 để ghép các hình ảnh sau khi chiếu và xử lý ánh sáng để tạo ra hình ảnh toàn cảnh hình trụ. được tạo ra.

Tiêu cự f là một tham số đáng kể khi sử dụng công thức chiếu hình trụ để biến đổi phép chiếu. Chúng tôi đặt các bản dịch giữa hai



Hình 8.9 Lưu đồ ghép ảnh dựa trên FFT

hình ảnh liền kề trong chuỗi hình ảnh trước khi chiếu như $C_1, C_2, \ldots, C_n$ tương ứng, ở đâu C_k đại diện cho phép dịch ngang giữa hình ảnh k và
hình ảnh $k+1$. Giá trị ban đầu của tiêu cự có tên là f_0 có thể được tính toán thông qua Công thức 8.9 :

$$f_0 \quad \sum_{k\,1}^{n} C_k \quad \overline{2\pi} \tag{8,9}$$

Mã nguồn của ghép ảnh dựa trên FFT được hiển thị dưới dạng CHƯƠNG TRÌNH 8.3.

CHƯƠNG TRÌNH 8.3: Khâu ảnh dựa trên FFT

Hàm main ()

```
clc; rõ ràng;
image1 = (imread ('D1.bmp'))); [h, w, r] = size (image1);
image2 = (imread ('D2.bmp'))); T0 (:,:,:, 1) = image1; T0 (:,:,:, 2) = image2;
subplot (121), imshow (image1); subplot (122), imshow (image2); image11 =
multi_resolution (image1,2);
image22 = multi_resolution (image2,2);
[r1, c1, d1] = size (image11);
[r2, c2, d2] = size (image22);
% tính toán hiệu số tương quan pha
tic
fprintf ('tính toán hiệu số tương quan pha ...');
[i, j] = poc_2pow (image11, image22);
coor_shift (1,1) = i;
coor_shift (1,2) = j;
coor_shift (2,1) = 0; coor_shift (2,2) = 0;
coor_shift = coor_shift * 2 ^ 2; %%% chuyển đổi thành phần bù của hình ảnh gốc toc

% biến đổi thành hệ tọa độ trụ tic

f = sqrt (h ^ 2 + ŵ 2);
[T1, coor_shift02] = coortransf (T0, f, coor_shift);
toc
% hợp nhất các khu vực chồng chéo
tic
fprintf ('Hợp nhất các vùng chồng chéo và ghép ảnh ...');
panorama1 = mosaic (T1 (:,:,:, 1), T1 (:,:,:, 2), coor_shift02 (1,1), coor_shift02 (1,2)); toc
```

```
% tái tạo hình ảnh
tic
fprintf ('Đang lưu và hiển thị kết quả ...');
imwrite (panorama1, 'pic2.jpg', 'jpg');
imshow (panorama1, []);
Toc

hàm T = multi_resolution (Xb, n)
% phân hủy đa độ phân giải
[r1, c1, d1, N] = size (Xb);
cho i = 1: N
    Xb (:,:, 1, N) = filter2 (fspecial ('gaussian'), Xb (:,:, 1, N))); %% Tham số mặc định của bộ lọc Gaussian [3 3]
  sigma = 0,5
    Xb (:,:, 2, N) = filter2 (fspecial ('gaussian'), Xb (:,:, 2, N));
    Xb (:,:, 3, N) = filter2 (fspecial ('gaussian'), Xb (:,:, 3, N));
    kết thúc
    bước = 2 ^ n;
    cho i = 1: bước: r1
        cho j = 1: bước: c1
            T ((i + step-1) / step, (j + step-1) / step,:,:) = Xb (i, j,:, :);
        kết thúc
    kết thúc

    function [dis, dm] = poc_2pow (imageL, imageR);
    Thuật toán tương quan pha %%%
    imageL = image1;
    % imageR = image2;
    [H1, W1, d1] = size (imageL);
    [H2, W2, d2] = size (imageR);
    nếu d1 == 3 imageL = rgb2gray (imageL); end %%% thang độ xám
    nếu d2 == 3 imageR = rgb2gray (imageR); kết thúc
    % trích xuất các đường viền nhị phân của biểu đồ 2 mũ [imageL, t1] = edge (imageL,
    'canny', [], 1.2); %%%% sigma = 1.2 (Mặc định 1) [imageR, t2] = edge ( imageR,
    'canny', [], 1.2); %%%% tự động chọn giá trị ngưỡng Xb = imageL; Yb = imageR;

    Biểu đồ% 2 hàm mũ
    cho i = 5: 11
        index2 = 2 ^ i;
        nếu index2 <= H1 && index2 <= W1 h1 = index2; end
        nếu index2 <= H2 && index2 <= W2 h2 = index2; end
```

```
kết thúc
% minhw1 = min (h1, w1);
% minhw2 = min (h2, w2);
minhw1 = h1; minhw2 = h2;
offset1 = round ((H1-minhw1) / 2);
offset2 = round ((H2-minhw2) / 2);
imageL = imageL (offset1: offset1 + minhw1-1, W1-minhw1 + 1: W1); %%% chọn chính giữa bên trái
    hình ảnh
imageR = imageR (offset2: offset2 + minhw2-1,1: minhw2); %%%% chọn tâm trái trong thuật toán tương
quan pha% hình ảnh bên phải để đo hiệu số
A = fft2 (im2double (imageL));% FFT trong miền tần số B =
fft2 (im2double (imageR));
AB = obs (A). * (B); %%% tích chập liên hợp, bằng với phép biến đổi pha
modAB = abs (AB);
%giá trị cao        IJ lưu các tọa độ đỉnh là các hiệu số
COR = ifft2 (AB); %%% không chuẩn hóa, biến đổi ngược cho hệ số emin =
100000;
% cho i = 1: 10
    [maxC, sorti] = max (COR);
    [C, J] = max (maxC);
    I = sorti (J);
    nếu tôi <20 dis = I;
    elseif H2-I <20 dis = (I-H2);
    khác dis = 0;
    kết thúc
    dm = J;


function [T1, coor_shift02] = coortransf (T0, f, coor_shift)
Chuyển đổi %% từ tọa độ ảnh sang tọa độ trụ% biến đổi chuỗi ảnh
đầu vào T0 với tiêu cự f
coor_shift02 = coor_shift; %%%% thứ nguyên đầu tiên (các giá trị hàng) không đổi và thứ nguyên thứ hai
    thứ nguyên (giá trị cột) cập nhật sau khi ánh xạ [H,
W, r, N] = size (T0);
w2f = W / 2 / f;
h2 = H / 2;
hằng2 = f * atan (W / (2 * f));
hằng số1 = h2;
```

```
cho y = 1: W %%%%%% cột
    angle = atan (y / f-w2f); %%%% atan ((yW / 2) / f);
      y1 = uint16 (f * góc + hằng số2);
      nếu y1 == 0 y1 = 1; kết thúc
    cho x = 1: H %%%%%%%%%%%% hàng
      x1 = uint16 ((x-h2) * cos (góc) + hằng số1);
      nếu x1 == 0 x1 = 1; kết thúc
      nếu r == 3 %%%%%%%%%%%%%% hình ảnh màu
          cho n = 1: N %%%%%%%%%%
              if (y == coor_shift (n, 2)) coor_shift02 (n, 2) = y1; end %%% các hiệu số tương ứng T1
              (x1, y1,:, n) = T0 (x, y,:, n); %%%% ánh xạ các điểm
          kết thúc
      elseif r == 1
      kết thúc
    kết thúc
kết thúc
[h, w, a, N] = size (T1);
cho i = 1: 60
    cho j = 1: w
        if (T1 (i, j,:,:) == 0)
            T1 (i, j,:,:) = 255;
        kết thúc
        if (T1 (chào, j,:,:) == 0)
            T1 (i, j,:,:) = 255;
        kết thúc
    kết thúc
kết thúc

hàm D = mosaic (image1, image2, i, j)
[ra, ca, a] = size (image1);
[rb, cb, b] = size (image2);
Xa = image1; Ya = image2;
% dis = i; %%% hiệu số trên và dưới dis
= i;
EXa = số không (abs (dis), ca, 3) +255;
EXb = số không (abs (dis), cb, 3) +255;
nếu dis> 1
  Xa = [EXa; Xa];
  Ya = [Ya; EXb];
elseif dis <-1
    Xa = [Xa; EXa];
    Ya = [EXb; Ya];
```

```
kết thúc
dm = j; %%% chiều rộng vết nứt đường khâu, giới hạn không quá 50 viên
    A = Xa (:, 1: (ca-dm-1), :);
    B1 = Xa (:, (ca-dm): ca, :);
    B2 = Ya (:, 1: dm, :);
    B = imagefusion02 (B1, B2); %% chồng chéo một phần (hợp nhất)
    C = Ya (:, (dm + 1): cb,:); %% cắt phần còn lại của hình ảnh thứ hai D = [A,
    B, C]; %% hợp nhất và hoàn thành khâu
    %%%% loại bỏ lỗi tích lũy [r, c] = size
    (D);
    nếu dis> 1
        D = D (1: (r-dis),:, :);
    elseif dis <-1
        D = D ((abs (dis) +1): r,:, :);
    kết thúc
    function [dis, dm] = phase_correlation (image1, image2);
    thuật toán tương quan% pha
    %%% C = phase_correlation (hình ảnh, imageb)        nhập hai hình ảnh
    [H1, W1, d1] = size (image1);
    [H2, W2, d2] = size (image2);
    nếu d1 == 3 image1 = rgb2gray (image1); end %%% thang độ xám
    nếu d2 == 3 image2 = rgb2gray (image2); end
    Xb = image1; Yb = image2;
    [image1, t1] = edge (image1, 'canny');
      [image2, t2] = edge (image2, 'canny');
    A = fft2 (im2double (image1));% FFT trong miền tần số B =
    fft2 (im2double (image2));
    AB = suggest (A). * (B); %%% tích chập liên hợp, tương đương với biến đổi pha
    COR = ifft2 (AB); %%% không chuẩn hóa, biến đổi ngược cho hệ số [C, i] = max
    (COR); [ C, J] = max (C); I = i (J); %%% giá trị đỉnh                IJ lưu các tọa độ đỉnh là các hiệu số
    nếu tôi <15 dis = I;
    elseif H2-I <15 dis = I-H2;
    khác dis = 0;
    kết thúc
    dm = J;
    hàm C = imagefusion02 (A, B)
    %%% kết hợp hình ảnh
    [M, N, D] = size (A);
```

```
nếu D == 3
cho i = 1: (N-1)
C (:, i,:) = round ((double (A (:, i,:)) * (Ni) + double (B (:, i,:)) * i) / N); kết thúc

elseif D == 1
cho i = 1: (N-1)
C (:, i) = round ((double (A (:, i)) * (Ni) + double (B (:, i)) * i) / N); kết
thúc
kết thúc
% figure, imshow (C / max (max (max (C))));
```

Nhân vật 8.10 là kết quả của việc ghép ảnh.

Khâu ảnh dựa trên FFT yêu cầu ảnh phải có cùng kích thước và hơn 30% diện tích chồng lên nhau. Ngoài ra, nó chỉ áp dụng cho việc đăng ký ảnh với dịch, xoay và chia tỷ lệ. Không áp dụng được các biến dạng phi tuyến tính như phép biến đổi tiếp tuyến. Thuật toán này chỉ sử dụng thông tin pha trong phổ công suất chéo để đăng ký hình ảnh do đó nó không nhạy cảm với sự thay đổi độ sáng giữa các hình ảnh.

(a) Như vậy u rce tôi ag e A (b) Như vậy u rce tôi ag e B

(c) Cái o u t pu t pa cũng không a m a

Hình 8.10 Bản đồ phác thảo để ghép ảnh

8.3 Ghép ảnh dựa trên điểm đặc trưng

Thay vì sử dụng các giá trị pixel của hình ảnh, ghép hình ảnh [2] dựa trên điểm tính năng [3] tính toán các đặc điểm như kết cấu, cạnh, đối tượng, v.v., từ các pixel, sau đó sử dụng các đối tượng địa lý làm tiêu chuẩn và tìm kiếm đối sánh cho các vùng đặc trưng tương ứng của hình ảnh chồng chéo. Loại phương pháp tiếp cận này mạnh mẽ hơn. Có hai quy trình để ghép ảnh dựa trên các điểm đặc trưng: trích xuất tính năng và đăng ký tính năng. Đầu tiên, trích xuất các điểm, đường và vùng mà thang màu xám thay đổi rõ ràng thành tập hợp tính năng của forma. Thứ hai, cố gắng chọn các tính năng được ghép nối bằng cách sử dụng các thuật toán đối sánh tính năng giữa hai tập hợp tính năng. Một loạt các phương pháp tiếp cận phân đoạn hình ảnh đã được áp dụng để trích xuất đặc trưng và phát hiện cạnh, chẳng hạn như bộ mô tả canny, bộ mô tả Laplace-gauss và hạt giống vùng đang phát triển (RSD). Các đặc điểm không gian được trích xuất bao gồm các cạnh đóng, các cạnh mở, các đường chéo và các đặc trưng khác.

8.3.1 Phát hiện điểm tính năng SIFT

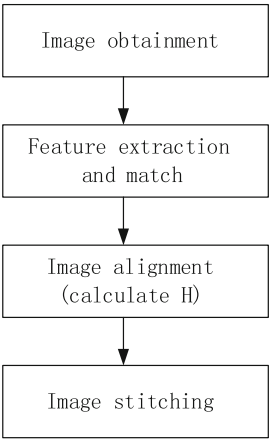
Quá trình ghép ảnh dựa trên các điểm đặc trưng của SIFT bao gồm: thu nhận ảnh, trích xuất và đối sánh đặc điểm, đăng ký ảnh (tính toán H) và ghép ảnh cuối cùng.

(1) Thu nhận hình ảnh. Thu nhận hình ảnh là điều kiện tiên quyết để ghép hình ảnh. Các bộ thu thập hình ảnh khác nhau có thể thu được các chuỗi hình ảnh đầu vào khác nhau và tạo ra các hiệu ứng ghép hình ảnh khác nhau. Hiện tại, có ba phương pháp khác nhau để thu được chuỗi hình ảnh: (1) x đặt máy ảnh vào chân máy và xoay nó để lấy dữ liệu hình ảnh; (2) fi x máy ảnh trên nền có thể di chuyển và dữ liệu hình ảnh thu được bằng cách di chuyển song song nó; (3) Cầm máy ảnh để chụp dữ liệu hình ảnh bằng điểm fi xed xoay hoặc di chuyển theo hướng vuông góc với trục quang học của máy ảnh. Quá trình này sử dụng các hình ảnh đã cho.

(2) Khai thác và đối sánh tính năng. Trích xuất các điểm đặc trưng của SIFT từ chuỗi hình ảnh đầu vào. Thuật toán tính toán và trích xuất các điểm đặc trưng đồng thời trong miền không gian và miền tỷ lệ. Do đó, các điểm đặc trưng thu được có tỷ lệ bất biến, có thể trích xuất chính xác các điểm đặc trưng tồn tại trong chuỗi ảnh với tỷ lệ lớn và thay đổi góc. Khoảng cách Euclide được sử dụng để tính toán khoảng cách giữa hai bộ mô tả điểm đặc trưng SIFT.

(3) Đăng ký hình ảnh với tính toán H. Đăng ký hình ảnh dựa trên các điểm đặc trưng có nghĩa là ma trận chuyển đổi giữa các chuỗi hình ảnh được xây dựng bằng các điểm phù hợp để hoàn thành việc ghép các hình ảnh toàn cảnh. Để cải thiện độ chính xác của đăng ký hình ảnh, thuật toán RANSAC [4] được sử dụng để tính toán và xác định lại ma trận biến đổi. Thuật toán H tự động tính toán ma trận biến đổi: tính điểm đặc trưng trong từng ảnh;

Hình 8.11 Quá trình ghép ảnh dựa trên SIFT



phù hợp với các điểm tính năng; tính giá trị ban đầu của ma trận; sử dụng phép lặp để tái lập ma trận biến đổi H; kết hợp khởi động; lặp lại các bước cho đến khi ổn định số điểm tương ứng.

(4) Hợp nhất hình ảnh. Theo ma trận biến đổi H của hai hình ảnh, các hình ảnh tương ứng có thể được chuyển đổi để xác định vùng chồng chéo của các hình ảnh và đăng ký các hình ảnh sẽ được hợp nhất vào một hình ảnh trống mới để tạo thành một sơ đồ khảm. Một thuật toán làm mịn có trọng số nhanh chóng và đơn giản được sử dụng để giải quyết vấn đề về đường may.

Quy trình của thuật toán ghép ảnh dựa trên các điểm đặc trưng SIFT được thể hiện trong Hình. 8,11 :

Chương trình nguồn TheMATLAB (mã chính) của thuật toán ghép ảnh dựa trên các điểm đặc trưng SIFT được trình bày trong CHƯƠNG TRÌNH 8.4. Chương trình phát hiện tính năng SIFT được hiển thị trong Chap. 4 CHƯƠNG TRÌNH 4.9.

CHƯƠNG TRÌNH 8.4: Ghép ảnh dựa trên các điểm đặc trưng của SIFT

```
function [imgout] = imMosaic (img1, img2, adjColor)%
sử dụng SIFT để tìm điểm tương ứng
[matchLoc1 matchLoc2] = siftMatch (img1, img2); %
sử dụng RANSAC để tìm ma trận đồng nhất
[H corrPtIdx] = findHomography (matchLoc2 ', matchLoc1'); H% #
được
tform = maketform ('projective', H ');
```

```
img21 = imtransform (img2, tform); % reproject img2%
fis (img1)
% cá (img21)
% điều chỉnh tuyến tính màu hoặc thang độ xám, sử dụng thông tin tương
ứng [M1 N1 dim] = size (img1);
[M2 N2 ~] = size (img2);
nếu tồn tại ('adjColor', 'var') && adjColor == 1
    bán kính = 2;
    x1ctrl = matchLoc1 (corrPtIdx, 1);
    y1ctrl = matchLoc1 (corrPtIdx, 2);
    x2ctrl = matchLoc2 (corrPtIdx, 1);
    y2ctrl = matchLoc2 (corrPtIdx, 2);
    ctrlLen = chiều dài (corrPtIdx);
    s1 = số không (1, ctrlLen);
    s2 = số không (1, ctrlLen);
    cho màu = 1: mờ
      cho p = 1: ctrlLen
            left = round (max (1, x1ctrl (p) -radius));
            right = round (min (N1, left + radius + 1));
            up = round (max (1, y1ctrl (p) -radius));
            down = round (min (M1, up + radius + 1));
            s1 (p) = sum (sum (img1 (up: down, left: right, color))); % tính toán sắc độ của các điểm xung quanh
      kết thúc
      cho p = 1: ctrlLen
            left = round (max (1, x2ctrl (p) -radius));
            right = round (min (N2, left + radius + 1));
            up = round (max (1, y2ctrl (p) -radius));
            down = round (min (M2, up + radius + 1));
            s2 (p) = sum (sum (img2 (up: down, left: right, color)));
      kết thúc
      sc = (bán kính * 2 + 1) ^ 2 * ctrlLen;
      adjcoef = polyfit (s1 / sc, s2 / sc, 1);
      img1 (:,:, color) = img1 (:,:, color) * adjcoef (1) + adjcoef (2); kết thúc

kết thúc
% làm khảm
pt = số không (3,4);
pt (:, 1) = H * [1; 1; 1];
pt (:, 2) = H * [N2; 1; 1];
pt (:, 3) = H * [N2; M2; 1];
```

```
pt (:, 4) = H * [1; M2; 1];
x2 = pt (1,:) ./ pt (3, :);
y2 = pt (2,:) ./ pt (3, :);
up = round (min (y2));
Yoffset = 0;
nếu lên <= 0
    Yoffset = -up + 1;
    lên = 1;
kết thúc
left = round (min (x2));
Xoffset = 0;
nếu trái <= 0
    Xoffset = -left + 1;
    trái = 1;
kết thúc
[M3 N3 ~] = size (img21);
imgout (lên: lên + M3-1, trái: trái + N3-1, :) = img21;
    % img1 ở trên img21
imgout (Yoffset + 1: Yoffset + M1, Xoffset + 1: Xoffset + N1, :) = img1; kết thúc

% [matchLoc1 matchLoc2] = siftMatch (img1, img2)
% Hàm này đọc hai hình ảnh, tìm các tính năng SIFT của chúng
và%   hiển thị các đường kết nối các điểm chính phù hợp. Amatch được chấp nhận%
      chỉ khi khoảng cách của nó nhỏ hơn distRatio nhân với khoảng cách
đến%  trận đấu gần nhất thứ hai.
% Nó trả về các điểm phù hợp của cả hai hình ảnh, matchLoc1 = [x1, y1; x2, y2;
...]% Ví dụ: match ('scene.pgm', 'book.pgm');
function [matchLoc1 matchLoc2] = siftMatch (img1, img2)%
load matchdata
% tải img1data
% tải img2data
% {,
% Tìm các điểm khóa SIFT cho mỗi hình ảnh
[des1, loc1] = sift (img1);
[des2, loc2] = sift (img2); %
save img1data des1 loc1%
save img2data des2 loc2
% Để đạt hiệu quả trong Matlab, sẽ rẻ hơn nếu tính các tích số chấm
giữa các vectơ đơn vị% hơn là khoảng cách Euclide. Lưu ý rằng tỷ lệ%
góc (tích số chấm của vectơ đơn vị) là một giá trị gần đúng
```

```
% so với tỷ lệ các khoảng cách Euclide đối với các góc nhỏ.
% distRatio: Chỉ giữ các kết quả phù hợp trong đó tỷ lệ các góc vectơ từ%
      gần nhất với hàng xóm gần nhất thứ hai nhỏ hơn distRatio.
distRatio = 0,6;
% Đối với mỗi bộ mô tả trong hình ảnh đầu tiên, hãy chọn đối sánh của nó với hình ảnh thứ hai. des2t
= des2 ';                                          % Chuyển vị ma trận tính trước
matchTable = zeros (1, size (des1,1));
for i = 1: size (des1,1)
    dotprods = des1 (i, :) * des2t;                % Tính vectơ của các sản phẩm chấm
    [vals, indx] = sort (acos (dotprods)); % Lấy cosin nghịch đảo và sắp xếp kết quả% Kiểm tra
xem người hàng xóm gần nhất có góc nhỏ hơn distRatio lần 2 hay không.
    if (vals (1) <distRatio * vals (2))
        matchTable (i) = indx (1);
    khác
        matchTable (i) = 0;
    kết thúc
kết thúc
% save matchdata
matchTable%}
% Tạo một hình ảnh mới hiển thị hai hình ảnh cạnh nhau.
img3 = appendimages (img1, img2);
Hiển thị một con số với các dòng tham gia các trận đấu được
chấp nhận. figure ('Vị trí', [100 100 size (img3,2) size (img3,1)]);
bản đồ màu ('xám');
imagesc (img3);
Giữ lấy;
cols1 = size (img1,2);
for i = 1: size (des1,1)
   nếu (matchTable (i)> 0)
       dòng ([loc1 (i, 2) loc2 (matchTable (i), 2) + cols1], ...
             [loc1 (i, 1) loc2 (matchTable (i), 1)], 'Màu', 'c');
   kết thúc
kết thúc
giữ lại;
num = sum (matchTable> 0);
fprintf ('Tìm thấy% d phù hợp. \ n', num);
idx1 = find (matchTable);
idx2 = matchTable (idx1);
x1 = loc1 (idx1,2);
x2 = loc2 (idx2,2);
```

```
y1 = loc1 (idx1,1);
y2 = loc2 (idx2,1);
matchLoc1 = [x1, y1];
matchLoc2 = [x2, y2];
kết thúc
% [descriptors, locs] = sift (img)
function [f inlierIdx] = ransac1 (x, y, ransacCoef, funcFindF, funcDist)%
[f inlierIdx] = ransac1 (x, y, ransacCoef, funcFindF, funcDist)% Sử dụng
RANdom SAmple Consensus để tìm sự phù hợp từ X đến Y. % X là ma
trận M * n gồm n điểm có dimM, Y là N * n; % Sự phù hợp, f và các chỉ
số của các nội số, được trả về.
%
% RANSACCOEF là một cấu trúc có các trường sau:%
minPtNum, iterNum, thDist, thInlrRatio
% MINPTNUM là số điểm tối thiểu mà chúng tôi% có thể tìm thấy điểm phù hợp.
Đối với điều chỉnh dòng, nó là 2. Đối với đồng nhất, nó là 4.
% ITERNUM là số lần lặp, THDIST là số trong
% khoảng cách ngưỡng và ROUND (THINLRRATIO * n) là ngưỡng số nội bộ. %

% FUNCFINDF là một xử lý func, f1 = funcFindF (x1, y1)
% x1 là M * n1 và y1 là N * n1, n1> = ransacCoef.minPtNum% f1 có
thể thuộc bất kỳ loại nào.
% FUNCDIST là một xử lý func, d = funcDist (f, x1, y1)
% Nó sử dụng f được trả về bởi FUNCFINDF và trả về khoảng
cách% giữa f và các điểm, d là 1 * n1.
% Để điều chỉnh dòng, nó sẽ tính toán khoảng cách giữa dòng và các
điểm% [x1; y1]; đối với phép đồng nhất, nó sẽ chiếu x1 đến y2 sau đó%
tính toán khoảng cách giữa y1 và y2.
minPtNum = ransacCoef.minPtNum;
iterNum = ransacCoef.iterNum;
thInlrRatio = ransacCoef.thInlrRatio;
thDist = ransacCoef.thDist;
ptNum = size (x, 2);
thInlr = round (thInlrRatio * ptNum);
inlrNum = zeros (1, iterNum);
fLib = ô (1, iterNum);
cho p = 1: iterNum
    % 1. phù hợp bằng cách sử dụng điểm ngẫu nhiên
    sampleIdx = randIndex (ptNum, minPtNum);
    f1 = funcFindF (x (:, sampleIdx), y (:, sampleIdx));
```

```
    % 2. đếm các nội số, nếu nhiều hơn thInlr, tái trang bị; else lặp
    dist = funcDist (f1, x, y);
    inlier1 = find (dist <thDist);
    inlrNum (p) = length (inlier1);
    nếu length (inlier1) <thInlr, tiếp tục; end fLib
    {p} = funcFindF (x (:, inlier1), y (:, inlier1));
kết thúc
% 3. chọn coef có nhiều nội số nhất [~, idx]
= max (inlrNum);
f = fLib {idx};
dist = funcDist (f, x, y);
inlierIdx = find (dist <thDist); kết
thúc
thông thoáng
đóng tất cả
f = 'a';
ext = 'jpg';
img1 = imread ([f '1.' ext]); img2 =
imread ([f '2.' ext]); % img3 =
imread ('b3.jpg'); img0 =
imMosaic (img2, img1,1);
% img0 = imMosaic (img1, img0,1);
cá (img0)
imwrite (img0, ['mosaic_' f '.' ext], ext)
```

Kết quả của việc ghép ảnh được thể hiện trong Hình. 8.12 :

(a) ori g trong a tôi là tôi ag e A **(b) ori g trong a tôi là tôi ag e B** **(c) stitchin g Tôi a m g e**

Hình 8.12 Kết quả ghép ảnh

8.3.2 Khâu ghép ảnh dựa trên các điểm đặc trưng của Harris

Quá trình ghép ảnh dựa trên các điểm đặc trưng của Harris như sau.

(1) Phát hiện các điểm đặc trưng Harris của hình ảnh;
(2) Kết nối các điểm đặc trưng giữa hai hình ảnh và khớp hình ảnh hoàn chỉnh; (3) Lọc tất cả các điểm phù hợp và các điểm phù hợp cần thiết để ghép ảnh; (4) Tính khoảng cách giữa các điểm đặc trưng của hai ảnh và làm mịn
các phần chồng chéo của hình ảnh.

Mã lõi được hiển thị trong CHƯƠNG TRÌNH 8.5.

CHƯƠNG TRÌNH 8.5: Ghép ảnh dựa trên các điểm đặc trưng của Harris

```
clc
Làm sạch tất cả
% đọc hình ảnh
pic1 = imread ('lena1.jpg');
pic2 = imread ('lena2.jpg');
% Harris tính năng điểm phát hiện
điểm1 = myHarris (pic1);
points2 = myHarris (pic2);
% vẽ các điểm đặc trưng của Harris
hình (1)
drawHarrisCorner (pic1, points1, pic2, points2);
% mô tả tính năng Harris
des1 = myHarrisCornerDescription (pic1, điểm1);
des2 = myHarrisCornerDescription (pic2, points2);
% so khớp thô
trận đấu = myMatch (des1, des2);
% có được vị trí của các điểm phù hợp đã
khớpPoints1 = points1 (matchs (:, 1), :);
matchPoints2 = points2 (matchs (:, 2), :);
% dòng điểm đối sánh thô
con số (2)
drawLinedCorner (pic1, matchPoints1, pic2, matchPoints2); %
Harris điểm tính năng phù hợp tốt
[newLoc1, newLoc2] = pointsSelect (matchPoints1, matchPoints2); %
dòng điểm đối sánh tốt
hình (3)
drawLinedCorner (pic1, newLoc1, pic2, newLoc2);
% khâu hình ảnh
```

```
im = picMished (pic1, newLoc1, pic2, newLoc2);
% hiển thị hình ảnh đường
may (4)
imshow (im);
set (gcf, 'Color', 'w');
function point = myHarris (pic)
% chức năng: Phát hiện điểm tính năng Harris% đầu
vào: Hình ảnh RGB hoặc hình ảnh thang màu xám
% đầu ra: ma trận hàng và cột N × 2 của điểm đặc trưng
if length (size (pic)) == 3
pic = rgb2gray (pic);
kết thúc
pic = đôi (pic);
hx = [- 1 0 1];
Ix = filter2 (hx, pic);
hy = [- 1; 0; 1];
Iy = filter2 (hy, pic);
Ix2 = Ix. * Ix;
Iy2 = Iy. * Iy;
Ixy = Ix. * Iy;
h = fspecial ('gaussian', [7 7], 2);
Ix2 = filter2 (h, Ix2);
Iy2 = filter2 (h, Iy2);
Ixy = filter2 (h, Ixy);
[heigth, width] = size (pic);
alpha = 0,06;
R = số không (chiều cao, chiều rộng);
for i = 1: heigth
cho j = 1: chiều rộng
M = [Ix2 (i, j) Ixy (i, j); Ixy (i, j) Iy2 (i, j)];
R (i, j) = det (M) -alpha * (dấu vết (M) ^ 2);
kết thúc
kết thúc
Rmax = max (cực đại (R));
pMap = zeros (heigth, width);
cho i = 2: heigth-1
cho j = 2: width-1
nếu R (i, j)> 0,01 * Rmax
tm = R (i-1: i + 1, j-1: j + 1);
tm (2,2) = 0;
```

```
nếu R (i, j)> tm
pMap (i, j) = 1;
kết thúc
kết thúc
kết thúc
kết thúc
[row, col] = find (pMap == 1);
điểm = [hàng, cột];
function drawHarrisCorner (pic1, points1, pic2, points2)
% function: draw Harris feature point 'match connection%
input:
% pic1 、 pic2: những hình ảnh cần khâu
% điểm1 、 điểm2: Vị trí điểm tính năng Harris
X1 = điểm1 (:, 2);
Y1 = điểm1 (:, 1);
X2 = điểm2 (:, 2);
Y2 = điểm2 (:, 1);
dif = size (pic1,2);
imshowpair (pic1, pic2, 'dựng phim');
Giữ lấy
plot (X1, Y1, 'b *');
plot (X2 + dif, Y2, 'b *');
set (gcf, 'Color', 'w');
function des = myHarrisCornerDescription (pic, điểm)
% Chức năng: mô tả các điểm tính năng của Harris%
Đầu vào:
% pic: hình ảnh nguồn
% điểm: điểm tính năng 'vị trí%
Sản lượng:
% des: 8 × N ma trận mô tả điểm đặc trưng của Harris nếu chiều dài
(kích thước (pic)) == 3
pic = rgb2gray (pic);
kết thúc
len = length (điểm);
des = zeros (8, len);
cho k = 1: len
p = điểm (k, :);
pc = pic (p (1), p (2));
des (1, k) = pic (p (1) -1, p (2) -1) -pc;
des (2, k) = pic (p (1), p (2) -1) -pc;
```

```
des (3, k) = pic (p (1) + 1, p (2) -1) -pc;
des (4, k) = pic (p (1) + 1, p (2)) - pc;
des (5, k) = pic (p (1) + 1, p (2) +1) -pc;
des (6, k) = pic (p (1), p (2) +1) -pc;
des (7, k) = pic (p (1) -1, p (2) +1) -pc;
des (8, k) = pic (p (1) -1, p (2)) - pc;
des (:, k) = des (:, k) / sum (des (:, k));
kết thúc
function matchs = myMatch (des1, des2)
% Chức năng: đối sánh hai chiều của điểm đặc trưng
% đầu vào:
% des1 、 des2: ma trận mô tả điểm tính năng%
Đầu ra:
% matchs: quan hệ tương ứng của các điểm so khớp
len1 = length (des1);
len2 = length (des2);
match1 = zeros (len1,2);
cor1 = số không (1, len2);
cho i = 1: len1
d1 = des1 (:, i);
cho j = 1: len2
d2 = des2 (:, j);
cor1 (j) = (d1 '* d2) / sqrt ((d1' * d1) * (d2 '* d2));
kết thúc
[~, indx] = max (cor1);
match1 (i,:) = [i, indx];
kết thúc
match2 = zeros (len2,2);
cor2 = số không (1, len1);
cho i = 1: len2
d2 = des2 (:, i);
cho j = 1: len1
d1 = des1 (:, j);
cor2 (j) = (d1 '* d2) / sqrt ((d1' * d1) * (d2 '* d2));
kết thúc
[~, indx] = max (cor2);
match2 (i,:) = [indx, i];
kết thúc
```

```
trận đấu = [];
for i = 1: length (match1)
for j = 1: length (match2)
nếu match1 (i,:) == match2 (j, :)
trận đấu = [trận đấu; trận đấu1 (i, :)];
kết thúc
kết thúc
kết thúc
function drawLinedCorner (pic1, loc1, pic2, loc2)
% Chức năng: vẽ kết nối giữa các điểm khớp% Đầu
vào:
% pic1 、 pic2: hình ảnh cần ghép% loc1 、 loc2:
vị trí của các điểm được ghép nối
X1 = loc1 (:, 2);
Y1 = loc1 (:, 1);
X2 = loc2 (:, 2);
Y2 = loc2 (:, 1);
dif = size (pic1,2);
imshowpair (pic1, pic2, 'dựng phim');
Giữ lấy
cho k = 1: chiều dài (X1)
plot (X1 (k), Y1 (k), 'b *');
plot (X2 (k) + dif, Y2 (k), 'b *');
line ([X1 (k), X2 (k) + dif], [Y1 (k), Y2 (k)], 'Màu', 'r');
kết thúc
set (gcf, 'Color', 'w');
function [newLoc1, newLoc2] = pointsSelect (loc1, loc2)
% Bộ lọc: lọc các điểm đối sánh được ghép nối và lấy điểm đối sánh tốt% Đầu
vào:
% loc1 、 loc2: vị trí của các điểm đối sánh thô
% Đầu ra:
% newLoc1 、 newLoc2: vị trí của các điểm đối sánh tốt độ
dốc = (loc2 (:, 1) -loc1 (:, 1)) ./ (loc2 (:, 2) -loc1 (:, 2));
cho k = 1: 3
dốc = độ dốc-trung bình (độ dốc);
len = chiều dài (độ dốc);
t = sort (abs (độ dốc));
thresh = t (round (0.5 * len));
ind = abs (độ dốc) <= thresh;
dốc = độ dốc (ind);
loc1 = loc1 (ind, :);
```

```
loc2 = loc2 (ind, :);
kết thúc
newLoc1 = loc1;
newLoc2 = loc2;
function im = picMatch (pic1, newLoc1, pic2, newLoc2)
% Chức năng: lấy hình ảnh đường may%
Đầu vào:
% pic1 、 pic2: hình ảnh cần ghép
% newLoc1 、 newLoc2: vị trí mới của điểm tính năng%
Sản lượng:
% im: hình ảnh đường may nếu
chiều dài (kích thước (pic1)) == 2
pic1 = cat (3, pic1, pic1, pic1);
kết thúc
if length (size (pic2)) == 2
pic2 = cat (3, pic2, pic2, pic2);
kết thúc
SZ = 2000;
X1 = newLoc1 (:, 2);
Y1 = newLoc1 (:, 1);
X2 = newLoc2 (:, 2);
Y2 = newLoc2 (:, 1);
sel = randperm (length (newLoc1), 3);
x = X2 (sel) ';
y = Y2 (sel) ';
X = X1 (sel) ';
Y = Y1 (sel) ';
Ư = [x; y; cái (1,3)];
V = [X; Y; cái (1,3)];
T = V / Ư;
cntrX = SZ / 2;
cntrY = SZ / 2;
im = số không (SZ, SZ, 3);
cho i = 1: kích thước (pic2,1)
cho j = 1: kích thước (pic2,2)
tmp = T * [j; i; 1];
nx = round (tmp (1)) + cntrX;
ny = round (tmp (2)) + cntrY;
if nx> = 1 && nx <= SZ && ny> = 1 && ny <= SZ im
(ny, nx,:) = pic2 (i, j, :);
```

```
kết thúc
kết thúc
kết thúc
im = imresize (im, 1, 'bicubic');
tpic1 = số không (SZ, SZ, 3);
tpic1 (1 + cntrY: size (pic1,1) + cntrY, 1 + cntrX: size (pic1,2) + cntrX,:) = pic1; re =
rgb2gray (uint8 (im)) - rgb2gray (uint8 (tpic1));
cho k = 1: 3
ta = im (:,:, k);
tb = tpic1 (:,:, k);
ta (re == 0) = tb (re == 0);
im (:,:, k) = ta;
kết thúc
rõ ràng ta tb lại tpic1
im = getPicture (im, SZ);
im = uint8 (im);
if length (size (pic1)) == 2
im = rgb2gray (im);
kết thúc
function im = getPicture (pic, SZ)
% Chức năng: lấy vùng hình ảnh hữu ích%
Đầu vào
% pic: hình ảnh đường may% SZ:
kích thước đã cho
% Đầu ra:
% im: vùng hình ảnh hữu ích nếu
chiều dài (kích thước (pic)) == 2
pic = cat (3, pic, pic, pic);
kết thúc
k = 1;
trong khi k <SZ
nếu có (bất kỳ (pic (k,:, :)))
phá vỡ
kết thúc
k = k + 1;
kết thúc
ceil = k; % Biên trên k =
SZ;
trong khi k> 0
nếu có (bất kỳ (pic (k,:, :)))
```

```
phá vỡ
kết thúc
k = k-1;
kết thúc
đáy = k; % Biên dưới k = 1;

trong khi k <SZ
nếu có (bất kỳ (pic (:, k, :)))
phá vỡ
kết thúc
k = k + 1;
kết thúc
trái = k; % ranh giới bên trái
k = SZ;
trong khi k> 0
nếu có (bất kỳ (pic (:, k, :)))
phá vỡ
kết thúc
k = k-1;
kết thúc
đúng = k; % bên phải ranh giới
%% lấy hình ảnh
im = pic (ceil: bottom, left: right, :);
```

8.3.3 Tự động sắp xếp cho chuỗi hình ảnh

Để ghép ảnh, chuỗi ảnh đầu vào [5] được đặt hàng theo nội dung cảnh thực tế, nghĩa là mỗi hai hình ảnh liền kề phải có các phần chồng lên nhau, để có thể ghép hình ảnh toàn cảnh chính xác. Tuy nhiên, trong quá trình chụp ảnh và lưu trữ hoặc nhập liệu, chuỗi ảnh có thể gây nhầm lẫn và không thể ghép trực tiếp. Để thực hiện tự động sắp xếp trình tự hình ảnh, có 3 vấn đề cần giải quyết trước hết:

(1) Xác định xem có vùng chồng chéo giữa hai hình ảnh hay không, nghĩa là, liệu hai hình ảnh có liên quan với nhau không;
(2) Xác định ảnh đầu và ảnh đuôi của dãy ảnh;
(3) Xác định mối quan hệ giữa vị trí bên trái và bên phải của hai trùng hình ảnh.

Trong phần này, chúng tôi sử dụng phương pháp tương quan pha để sắp xếp chuỗi ảnh. Nguyên tắc của phương pháp tương quan pha như sau:
Giả sử có một phần bù (*x, y)* giữa hình ảnh 1 $Tôi_1(x, y)$ và hình ảnh 2 $Tôi_2(x, y)$:

$$Tôi_1(x, y) \quad Tôi_2(x - x, y - y) \tag{8.10}$$

Theo đặc điểm dịch chuyển của phép biến đổi Fourier, đây là:

$$\text{Tôi}_1(u, v) \quad e^{-j2\pi(ux+vy)}\,\text{Tôi}_2(u, v) \tag{8,11}$$

Phổ công suất tương hỗ chuẩn hóa của nó được biểu diễn như sau:

$$\tilde{C}orr(u, v) \quad \frac{\hat{\text{Tôi}}_1(u, v)\,\hat{\text{tôi}}_2^{*}(u, v)}{|\hat{\text{Tôi}}_1(u, v)\,\hat{\text{tôi}}_2|}\, e^{-j2\pi(ux+vy)} \tag{8.12}$$

$\hat{\text{Tôi}}_1$ và $\hat{\text{Tôi}}_2$ Chúng tôi Tôi_1 và Tôi_2 biến đổi Fourier của riêng biệt, $\hat{\text{Tôi}}_2^{*}$ là khu phức hợp liên hợp của Tôi_2.

Pha của mật độ quang phổ công suất chéo bằng độ lệch pha của hai hình ảnh. Mật độ phổ công suất chéo chuẩn hóa được vận hành để có được hàm xung thông qua phép biến đổi Fourier nghịch đảo:

$$Corr(x, y) \quad F^{-1}[e^{-j2\pi(ux+vy)}] \quad \delta(x - x, y - y) \tag{8.13}$$

Hàm này nhận giá trị lớn nhất ở độ dịch chuyển tương đối (*x, y) (* điểm khớp) của hai hình ảnh, bất kỳ nơi nào khác gần 0. độ dịch chuyển tương đối (*x, y)* được xác định bằng cách loại bỏ vị trí của điểm nhìn lén trong công thức (4.4). Trong trường hợp chỉ dịch giữa các ảnh, độ lớn của đỉnh của hàm xung tạo ra mối tương quan giữa hai ảnh và nhận một giá trị trong một khoảng [0, 1]. Vùng chồng lấn giữa hai ảnh càng lớn thì giá trị càng lớn. Nếu hai hình ảnh có cùng nội dung, giá trị là 1 và giá trị là 0 khi nó hoàn toàn khác nhau. Nếu vẫn có phối cảnh, nhiễu hoặc mục tiêu chuyển động giữa hai hình ảnh, năng lượng của hàm xung sẽ được phân phối từ một đỉnh duy nhất đến các đỉnh nhỏ khác, nhưng vị trí đỉnh tối đa của nó có độ bền nhất định.

Theo nguyên tắc của phương pháp tương quan pha, thuật toán sắp xếp tự động như sau:

(1) Xác định hình ảnh đầu và đuôi (hình ảnh ngoài cùng bên trái và ngoài cùng bên phải) và liền kề hình ảnh.

Đối với một chuỗi hình ảnh nhất định có n hình ảnh, bất kỳ hình ảnh nào cũng có thể được tính toán bởi phần còn lại N - 1 hình ảnh để lấy N - 1 mức độ tương quan. Vì hình ảnh sẽ tiếp giáp với tối đa hai hình ảnh (một hình ảnh trung gian), nên ít nhất nó sẽ tiếp giáp với một trong các hình ảnh (hình ảnh đầu và đuôi). Do đó, nếu hai giá trị lớn nhất đầu tiên được chọn từ N - 1 tương quan tính từ ảnh thì ảnh sẽ chồng lên 2 hoặc 1 trong 2 tương quan. Hoạt động ở bên trái N - 1 hình ảnh, 2 N mức độ tương quan lớn nhất thu được. Đối với hình ảnh đầu và hình ảnh đuôi, hai mức độ tương quan tương ứng của chúng sẽ có một mức độ không đủ điều kiện. Rõ ràng, độ tương quan tương ứng của ảnh đầu và ảnh đuôi nhỏ hơn các độ khác. Khi xác định mức độ tương quan nhỏ nhất của 2 N độ, hình ảnh đầu và đuôi thu được tương ứng. Cuối cùng, hình ảnh đầu và đuôi khác với các hình ảnh bên cạnh.

(2) Xác định mối quan hệ giữa vị trí bên trái và bên phải của hai hình kề nhau.

Trong phương pháp thuật toán tương quan pha, kết quả đo cho thấy δ hàm xung với các đỉnh tương quan rất sắc nét khi hai hình ảnh thực sự có liên quan. Các tham số dịch ngang của hai ảnh có thể được tính bằng các điểm pixel tương ứng của đỉnh. Khi tham số dịch ngang x lớn hơn một nửa chiều rộng hình ảnh, bạn có thể trừ nó khỏi chiều rộng hình ảnh và sau đó lấy giá trị âm. Nếu dịch ngang giữa hình ảnh **A** và **B** là tiêu cực, Hình ảnh **A** ở bên trái của hình ảnh **B,** và ngược lại, hình ảnh **A** ở bên phải của hình ảnh **B.**

Do đó, việc sắp xếp tự động các chuỗi hình ảnh được hoàn thành.

Mã MATLAB được hiển thị trong CHƯƠNG TRÌNH 8.6, trong đó hàm poc_2pow đã được mô tả trong CHƯƠNG TRÌNH 8.3:

CHƯƠNG TRÌNH 8.6: Sắp xếp tự động chuỗi hình ảnh

```
rõ ràng; clc;
%%%%%%%%%%%%%%%%%% hình ảnh đầu vào, hình ảnh kiến   tạo kim tự tháp, lưu trữ trong một mảng tic


fprintf ('đầu vào hình ảnh, hình ảnh tòa nhà kim tự tháp
...'); cấp = 2;
T0 =   uint8 ([]); T = uint8 ([]);
%%%%%%%%%%%%%%%%%%%%%%%%%
i1 = imread ('X1.jpg'); T0 (:,:,:, 1) = i1;
i2 = imread ('X2.jpg'); T0 (:,:,:, 2) = i2;
i3 = imread ('X3.jpg'); T0 (:,:,:, 3) = i3;
[h, w, d] = size (T0 (:,:,:, 1)); %%% cùng kích thước hình ảnh được yêu
cầu% [T, Terr] = multi_resolution (T0, level);
T = multi_resolution (T0, cấp độ);
toc
% %%%%%%%%%%%%%%%%%%%%%%%% tính toán bù đắp thông qua tương quan pha tic


fprintf ('tính toán độ lệch thông qua tương quan pha
...'); M = 3; %% số hình ảnh
% L = M * w; %% tổng độ dài của hình ảnh
% suml = 0; %% tổng chiều dài của vùng chồng chéo cho
N = 1: M
if N <M [i, j] = poc_2pow (T (:,:,:, N), T (:,:,:, N + 1));
elseif N == M [i, j] = poc_2pow (T (:,:,:, N), T (:,:,:, 1));
kết thúc
```

```
coor_shift (N, 1) = i;
coor_shift (N, 2) = j;
%           suml = suml + j;
kết thúc
coor_shift = coor_shift * 2 ^ level; %%% chuyển đổi các hiệu số trong phân cấp kim tự tháp thành bù đắp
    nguyên
toc
%%%%%%%%%%%%%%%%%% Chuyển sang hệ tọa độ trụ tic

fprintf ('Biến đổi sang hệ tọa độ trụ ...'); f = sqrt (ĥ 2 + ŵ
2);
[T1, coor_shift02] = coortransf (T0, f, coor_shift);
toc
%%%%%%%%%%%%% hợp nhất các phần chồng chéo
tic
fprintf ('ghép các phần chồng lên nhau, ghép ảnh ...');
panorama1 = T1 (:,:,:, 1);
cho N = 1: M
nếu N <M panorama1 = mosaic (panorama1, T1 (:,:,:, N + 1), coor_shift02 (N, 1), coor_shift02 (N, 2)); kết thúc

kết thúc
toc
%%%%%%%%%%%%%%%%%% dựng lại hình ảnh
tic
fprintf ('lưu và hiển thị kết quả ...');
imwrite (panorama1, 'pic1.jpg', 'jpg');
imshow (panorama1, []);
toc
```

Ảnh hoàn chỉnh thu được sau khi sắp xếp các ảnh đầu vào.

8.3.4 Đăng ký điểm Harris dựa trên thuật toán RANSAC

Đăng ký điểm Harris dựa trên Đồng thuận mẫu ngẫu nhiên (RANSAC) là một loại phương pháp đối sánh dựa trên các tính năng. Trước tiên, việc phát hiện điểm Harris được thực hiện và sau đó thực hiện đối sánh thô theo đặc điểm cục bộ của các điểm được trích xuất để tìm ra sự tương ứng giữa các tập hợp điểm được so khớp. Sau khi đối sánh thô, hầu hết các cặp điểm khớp sai được loại bỏ, nhưng vẫn còn nhiều điểm thiếu so với yêu cầu. Các cặp điểm này có sai số lớn trong mối quan hệ hình học chủ yếu vẫn là do sự giống nhau về thông tin thang xám. Những điểm này là

được gọi là các cặp đối sánh giả. Thuật toán RANSAC được sử dụng để loại bỏ các cặp đối sánh giả.

RANSAC là một loại thuật toán lặp để ước tính các tham số mô hình toán học. Ý tưởng chính là tính toán các thông số để làm cho phần lớn các mẫu (điểm đặc trưng) có thể thiết lập mô hình toán học. Tại lần lặp lại, số lượng mẫu tối thiểu được sử dụng để lấy mẫu mô hình và tính toán các thông số và số lượng mẫu chuẩn bị cho mô hình được đếm. Và các tham số mẫu tối đa được coi là giá trị của mô hình cuối cùng. Điểm mẫu phù hợp với mô hình được gọi là phần trong và điểm mẫu không phù hợp với mô hình được gọi là điểm ngoài hoặc điểm hoang dã.

Các ý tưởng cơ bản của RANSAC như sau:

Hãy xem xét một mô hình yêu cầu một bộ lấy mẫu tối thiểu với *n* mẫu (*n* cho số lượng mẫu tối thiểu cần thiết để khởi tạo các thông số mô hình) và một bộ mẫu *P,* số mẫu của tập P # (P)> n. Tập hợp con *S* với *n* các mẫu được trích xuất ngẫu nhiên từ *P* được sử dụng để khởi tạo mô hình *M:*

Các mẫu trong bộ bổ sung *SC* *P / S* có lỗi nhỏ hơn ngưỡng đã đặt *t,* cùng với bộ *S* cấu tạo S_*. S_* được coi là bộ nội tại thiết lập và xây dựng *S* Bộ đồng thuận của.

Nếu #(S_*) * *N,* các tham số phù hợp được coi là thu được và phương pháp Bình phương nhỏ nhất, v.v. được sử dụng để ước tính mô hình mới M_* trên bộ nội quy $S_;$ hoặc lấy lại mẫu mới *S* và lặp lại.

Sau một số lần lấy mẫu nhất định, nếu không tìm thấy tập hợp nhất quán nào thì thuật toán không thành công, nếu không thì tập hợp đồng thuận thập phân thu được sau khi lấy mẫu và thuật toán kết thúc. Mã được hiển thị trong CHƯƠNG TRÌNH 8.7.

CHƯƠNG TRÌNH 8.7: Đăng ký điểm Harris dựa trên thuật toán RANSAC

```
điểm chức năng = kp_harris (im)
    % Trích xuất các điểm chính bằng thuật toán Harris (với phiên
    bản% cải tiến)
    % Tác giả :: Vincent Garcia%
    Ngày :: 05/12/2007
    % ĐẦU VÀO
  % tôi : hình ảnh graylevel
    % ĐẦU RA
    % điểm: điểm lãi suất được trích ra
```

```
% NGƯỜI GIỚI THIỆU
% CG Harris và MJ Stephens. "Máy dò góc và cạnh có kết cấu",% Kỷ yếu Hội
nghị Tầm nhìn Alvey lần thứ tư, Manchester. % trang 147-151, 1988.

% Alison Noble, "Mô tả các bề mặt hình ảnh", Luận án Tiến sĩ, Khoa Khoa học
Kỹ thuật, Đại học Oxford 1989, tr45.
% C. Schmid, R. Mohrand và C. Bauckhage, "Đánh giá công cụ dò tìm điểm quan tâm",%
Int. Tạp chí Thị giác Máy tính, 37 (2), 151-172, 2000.
% THÍ DỤ
% điểm = kp_harris (im)%
chỉ giá trị độ sáng im =
double (im (:,:, 1));
sigma = 1,5;
% mặt nạ phái sinh
s_D = 0,7 * sigma;
x = -round (3 * s_D): round (3 * s_D);
dx = x. * exp (-x. * x / (2 * s_D * s_D)) ./ (s_D * s_D * s_D * sqrt (2 * pi)); dy
= dx ';
% dẫn xuất hình ảnh
Ix = đối tượng (im, dx, 'giống nhau'); Iy =
convert2 (im, dy, 'giống nhau');
% tổng của ma trận Tương quan tự động
s_I = sigma;
g = fspecial ('gaussian', max (1, fix (6 * s_I + 1)), s_I);
Ix2 = var (Ix. ^ 2, g, 'giống nhau'); % Các dẫn xuất hình ảnh bình phương được làm mịn Iy2 =
convert2 (Iy. ^ 2, g, 'same');
Ixy = conc2 (Ix. * Iy, g, 'giống nhau');
% điểm lãi suất phản hồi
cim = (Ix2. * Iy2 - Ixy. ^ 2) ./ (Ix2 + Iy2 + eps);                    Số đo% Alison Noble.
% k = 0,06; cim = (Ix2. * Iy2 - Ixy. ^ 2) - k * (Ix2 + Iy2). ^ 2; % tìm       % Harris ban đầu đo.
thấy cực đại địa phương trên 3x3 hàng xóm
[r, c, max_local] = findLocalMaximum (cim, 3 * s_I);
% đặt ngưỡng 1% của giá trị lớn nhất t = 0,1
* max (max_local (:));
% tìm cực đại cục bộ lớn hơn ngưỡng [r, c] =
find (max_local> = t);
% xây dựng điểm quan
tâm = [r, c];
kết thúc
```

```
function [row, col, max_local] = findLocalMaximum (val, radius)
    % Xác định mức tối đa cục bộ của một giá trị nhất
    định%
    % Tác giả :: Vincent Garcia%
    Ngày :: 09/02/2007
    %
    % ĐẦU VÀO
    % val : ma trận NxM chứa các giá trị
    % radius: bán kính của vùng lân cận%
    OUTPUT
    % hàng : vị trí hàng của cực đại cục bộ
    % col : vị trí cột của cực đại cục bộ
    % max_local: ma trận NxM chứa các giá trị của val trên% tối đa cục bộ duy nhất
    EXAMPLE
    % [l, c, m] = findLocalMaximum (img, radius);
    % TÌM KIẾM DILATION LOCALMAXIMABY (NHANH CHÓNG) /! \ KHÔNG DUY NHẤT /! \
    % mask = fspecial ('disk', radius)> 0; %
    val2 = imdilate (val, mặt nạ);
    % chỉ số = val == val2;
    % [row, col] = find (index == 1); %
    max_local = zeros (size (val)); %
    max_local (chỉ số) = val (chỉ số);
    % TÌM KIẾM BỘ LỌC ĐỊA PHƯƠNG DUY NHẤT (NHANH CHÓNG) mask
    = fspecial ('disk', radius)> 0;
    nb = sum (mặt nạ (:));
    cao nhất = ordfilt2 (val, nb, mặt nạ);
    second_highest = ordfilt2 (val, nb-1, mặt nạ);
    mục lục = cao nhất == val & cao nhất ~ = thứ_highest;
    max_local = số không (kích thước (val));
    max_local (chỉ số) = val (chỉ số);
    [hàng, cột] = find (chỉ số == 1);
    % TÌM LOCALMAXIMA DUY NHẤT (NHANH
    CHÓNG)% val_height = size (val, 1);
    % val_width = kích thước (val, 2);
    % max_local = số không (val_height, val_width);
    % val_enlarge = zeros (val_height + 2 * radius, val_width + 2 * radius); %
    val_mask = số không (val_height + 2 * radius, val_width + 2 * radius);
    % val_enlarge ((1: val_height) + radius, (1: val_width) + radius) = val; %
    val_mask ( (1: val_height) + radius, (1: val_width) + radius) = 1;
    % mask = fspecial ('disk', radius)> 0;
```

```
% row = zeros (val_height * val_width, 1);
% col = zeros (val_height * val_width, 1); %
chỉ số = 0;
% cho l = 1:
val_heightlt% c = 1:
val_width%    val_ref = val (l, c);
%             lân_val = val_enlarge (l: l + 2 * bán kính, c: c + 2 * bán kính);
%             ne_mask = val_mask (          l: l + 2 * bán kính, c: c + 2 * bán kính). * mặt nạ;
%             ne_sort = sort (ne_val (ne_mask == 1));
%             if val_ref == ne_sort (end) && val_ref> ne_sort (end-1)
%                 mục lục                  = chỉ số + 1;
%                 hàng (chỉ mục, 1)        = l & agrave;
%                 col (chỉ mục, 1)         = c;
%                 max_local (l, c) = val_ref;
%             kết thúc
%         kết thúc
% kết thúc
    % hàng (chỉ mục + 1: end, :) =
    []; % col (index + 1: end, :) = [];
kết thúc
```

Mã RANSAC:

```
function [final_inaries flag bestmodel] = AffinePairwiseRansac (frames_a1, frames_a2, all_matches)%
iterations = 0
% bestfit = nil
% besterr = cái gì đó thực sự
lớn% trong khi lặp lại <k {
%       có thể có = n giá trị được chọn ngẫu nhiên từ dữ liệu%
        maybemodel = các thông số mô hình được trang bị cho có
thể%   alsoinlier = tập hợp
rỗng%  đối với mọi điểm trong dữ liệu không nằm trong các giá trị có thể {%
            nếu điểm phù hợp với mô hình maybemodel với sai số nhỏ hơn t
%               thêm điểm cho các ngoại lệ
%       }
%       nếu số lượng phần tử trong ngoại lệ> d {
%           điều này ngụ ý rằng chúng tôi có thể đã tìm thấy một mô hình tốt hiện đang
%           kiểm tra xem nó tốt như thế nào
%           goodmodel = thông số mô hình được trang bị cho tất cả các điểm trong các điểm có thể và ngoại lệ thiserr = một
%           thước đo về mức độ phù hợp của mô hình với các điểm này
%           nếu thiserr <besterr {
%               bestfit = mô hình tốt hơn
%               besterr = thiserr
```

```
%                }
%        }
%        lặp lại gia tăng
%}
% trở lại bestfit
%
% đầu tiên quyết định xem chúng tôi có bao nhiêu
trận đấu MIN_START_VALUES = 4;
num_matches = size (all_matches, 2);
nếu (num_matches <MIN_START_VALUES)
      cuối cùng = [];
      mô hình tốt nhất = [];
      cờ = -1;
      trở về
kết thúc
% Làm? Chúng có thể phải được thay đổi nếu các giá trị khác nhau.
Z_OFFSET = 640;
COND_THRESH = 45;
% RANSAC thông số
NUM_START_VALUES = 3; % chỉ 3 corrospondences cần thiết để xác định mô hình affine K =
50;
ERROR_THRESHOLD = 10; % khá cao - đây là số pixel D = 1; % điểm bổ sung
phải phù hợp với bất kỳ mô hình liên kết nhất định nào
N = NUM_START_VALUES;
RADIUS = 30; % đã thay đổi thành 30 bởi
vijay MIN_NUM_OUTSIDE_RADIUS = 1;
% lỗi tốt nhất, lần lặp
phù hợp nhất = 0;
besterror = inf;
mô hình tốt nhất = [];
cuối cùng = [];
max_inlier = 0;
trong khi (lần lặp <K)
      % bắt đầu bằng NUM_START_VALUES giá trị duy nhất
      uniqueValues   = [];
      max_index = size (all_matches, 2);
      while (length (uniqueValues) <NUM_START_VALUES)
            value = ceil (max_index * rand (1,1));
            if (length (find (value == uniqueValues)) == 0)
                  % giá trị khác 0 duy nhất
```

```
            uniqueValues   = [giá trị uniqueValues];
        kết thúc
    kết thúc
    % uniqueValues   là các chỉ số trong all_matches có
thểinaries = all_matches (:, uniqueValues); giá trị ngẫu       % bắt đầu với NUM_START_VALUES duy nhất
nhiên
    % đảm bảo điểm được phân phối tốt
    point_matrix = [frames_a1 (:, có thể có (1, :)); Z_OFFSET * cái (1, NUM_START_VALUES)]; if (cond
    (point_matrix)> COND_THRESH)
        lần lặp lại = iteration + 1;
        tiếp tục;
    kết thúc
    M_maybemodel = getModel (có thể, khung_a1, khung_a2); if
    (prod (size (M_maybemodel)) == 0)
        lần lặp lại = iteration + 1;
        tiếp tục;
    kết thúc
    cũng có = [];
    % tìm ra các nội dung khác
    cho i = 1: size (all_matches, 2)
        temp = find (all_matches (1, i) == mightinlier (1, :)); if
        (length (temp) == 0)
            % điều này có nghĩa là, trỏ không phải trong
            couldinlier a1 = frames_a1 (1: 2, all_matches (1,
            i)); a2 = frames_a2 (1: 2, all_matches (2, i));
            if (getError (M_maybemodel, a1, a2) <ERROR_THRESHOLD)
                alsoinlier = [alsoinaries all_matches (:, i)];
            kết thúc
        kết thúc
    kết thúc
    if (size (alsoinlier, 2)> 0)
        num = 0;
        dist = [];
        cho i = 1: NUM_START_VALUES
            diff =          frame_a1 (1: 2, alsoinlier (1, :)) - ...
                repmat (frames_a1 (1: 2, mightinlier (1, i)), [1, size (alsoinlier, 2)]); dist
            = [dist; sqrt (sum (diff. ^ 2))];
        kết thúc
        num = sum (sum (dist> RADIUS) == NUM_START_VALUES); nếu
        (num <MIN_NUM_OUTSIDE_RADIUS)
            lần lặp lại = iteration + 1;
```

```
            tiếp tục;
        kết thúc
    kết thúc
    % xem mô hình tốt như thế nào
    % fprintf ('Số phần tử trong các phần tử cũng nằm trong số các phần tử% d \ n', size (các phần tử cũng trong
    các phần tử, 2)); if (size (alsoinlier, 2)> D)
        % điều này ngụ ý rằng chúng tôi đã tìm thấy một mô hình tốt%
        bây giờ hãy xem nó tốt như thế nào
        % tìm thấy mô hình mới
        all_inlier = [có thể còn các yếu tố khác];
        M_bettermodel = getModel (all_inlier, khung_a1, khung_a2); %
        mô hình mới có thể là xấu
        if (prod (size (M_bettermodel)) == 0)
            lần lặp lại = iteration + 1;
            tiếp tục;
            kết thúc
            % tìm thấy lỗi cho mô hình
            thiserror = getModelError (M_bettermodel, all_inlier, frames_a1, frames_a2);
            nếu max_inlier <size (all_inlier, 2) | (thiserror <besterror & max_inaries == size (all_inlier, 2))
                bestmodel = M_bettermodel;
                besterror = thiserror;
                cuối cùng = tất cả các số lượng;
                max_iniers = size (final_inlier, 2);
            kết thúc
        kết thúc
        % làm điều đó K lần
        lần lặp lại = iteration + 1;
    kết thúc
    % bestmodel có Mô hình tốt nhất nếu (sản
    phẩm (kích thước (mô hình tốt nhất)) ~ = 0)
        % một mô hình đã được tìm thấy
        fprintf ('Lỗi của best_model ~% f pixel \ n', besterror);
        cờ = 1;
    khác
        cờ = -1;
        cuối cùng = [];
        mô hình tốt nhất = [];
        fprintf ('Không tìm thấy mô hình tốt! \ n');
    kết thúc
kết thúc
```

```
function error = getModelError (M, so khớp, khung_a1, khung_a2)
        nummatches = size (so khớp, 2);
        lỗi = 0;
        cho i = 1: nummatches
                a1 = frames_a1 (1: 2, so khớp (1, i));
                a2 = frames_a2 (1: 2, so khớp (2, i));
                error = error + getError (M, a1, a2);
        kết thúc
        error = error / nummatches;
kết thúc
function M = getModel (so khớp, khung_a1, khung_a2)
        % hãy chuyển từ 1 thành 2 - đã thay đổi vào ngày 28 tháng 4 để phù
        hợp% với các mô hình biểu đồ và phối cảnh
        singular_thresh = 1e-6;
        tỉ_lệ_hình_hình = 5;
        scale_thresh = 0,005;
        % gần đúng M
        M = số không (3,3);
        Y = []; X = [];
           for i = 1: size (khớp, 2)
              a1 = frames_a1 (1: 2, so khớp (1, i));
              a2 = frames_a2 (1: 2, so khớp (2, i)); Y
              = [Y; a2];
              X = [X; a1 (1) a1 (2) 1 0 0 0; 0 0 0 a1 (1) a1 (2) 1];
        kết thúc
          % để kiểm tra xem ma trận có phải là số ít
        không nếu (1 / cond (X) <singular_thresh)
              M = [];
              trở về
        kết thúc
          M = X \ Y;
        % chúng ta cần trả về M - một ma trận 3X3, trong đó hàng cuối cùng là (0 0 1)
        M = [reshape (M, 3,2) '; 0 0 1];
        % hãy thêm một số quy tắc để loại bỏ bất kỳ bản đồ điên rồ nào mà
        chúng tôi chắc chắn không thể có phản ánh
        [u, s, v] = svd (M (1: 2,1: 2)); if
        (det (u * v ') <0)
              % ==> có phản xạ M = [];
```

```
%                   fprintf ('Trường hợp đặc biệt để tránh phản chiếu \ n');
                    trở về
                kết thúc
                % chúng ta không thể có tỷ lệ nhân rộng điên rồ trong hai thứ nguyên. if
                (cond (M (1: 2,1: 2))> scaling_ratio_thresh)
                    % ==> các trận đấu là xấu M
                    = [];
%                   fprintf ('Trường hợp đặc biệt để tránh tỷ lệ chia tỷ lệ điên rồ \ n');
                    trở về
                kết thúc


                % kiểm tra thu phóng điên rồ
                if (s (1,1) <scale_thresh | s (2,2) <scale_thresh)
                    M = [];
                kết thúc
kết thúc
lỗi hàm = getError (M, a1, a2)
                % a2_model là giá trị của a2 đến từ lỗi ánh xạ tính toán
                mô hình%
                a2_mẫu = M * ([a1; 1]); % 3x1 vector, chỉ có hai giá trị đầu tiên problem error =
                dist (a2, a2_model (1: 2));
kết thúc
hàm d = dist (một, hai)
    d = sqrt (sum ((một-hai). ^ 2));
kết thúc
function [final_inaries flag bestmodel] = PerspectivePairwiseRansac (frames_a1, frames_a2, all_matches)
    % lần lặp = 0%
    bestfit = nil
    % besterr = cái gì đó thực sự lớn%
    trong khi lặp lại <k {
    %       có thể có = n giá trị được chọn ngẫu nhiên từ dữ liệu%
            maybemodel = các thông số mô hình được trang bị cho có thể%
            alsoinlier = tập hợp rỗng%
            đối với mọi điểm trong dữ liệu không nằm trong các giá trị có thể {%
                nếu điểm phù hợp với mô hình maybemodel với sai số nhỏ hơn t
    %                   thêm điểm cho các ngoại lệ
    %       }
    %       nếu số lượng phần tử trong ngoại lệ> d {
    %           điều này ngụ ý rằng chúng tôi có thể đã tìm thấy một mô hình tốt hiện đang
    %           kiểm tra xem nó tốt như thế nào
```

```
        %               goodmodel = thông số mô hình được trang bị cho tất cả các điểm trong các điểm có thể và ngoại lệ
        %               thiserr = một thước đo về mức độ phù hợp của mô hình với các điểm này
        %               nếu thiserr <besterr {
        %                       bestfit = mô hình tốt hơn
        %                       besterr = thiserr
        %               }
        %       }
        %       lặp lại gia tăng
        %}
        % trở lại bestfit
        %
        % tìm mô hình từ hình ảnh đầu tiên đến hình ảnh thứ hai
%
%       [XW] = [abc] [x]
%       [YW] = [def] [y]
%       [W] = [gh 1] [1]
% (x, y, 1) là các điểm trong hình ảnh đầu tiên và chúng ánh xạ tới (XW, YW, W)
trong% hình ảnh thứ hai
% trước tiên hãy quyết định xem chúng ta có bao nhiêu
trận đấu có MIN_START_VALUES = 20;
num_matches = size (all_matches, 2);
nếu (num_matches <MIN_START_VALUES)
        cuối cùng = [];
        mô hình tốt nhất = [];
        cờ = -1;
        trở về
kết thúc
% RANSAC thông số
K = 150;
NUM_START_VALUES = 4; % sử dụng giải pháp 4 điểm bình phương nhỏ nhất
ERROR_THRESHOLD = 10; % đây là số lượng pixel. D = 8; % bắt đầu với 4 điểm và
phù hợp với ít nhất 8 điểm phù hợp hơn với mô hình% lỗi tốt nhất, phù hợp nhất

lần lặp = 0;
besterror = inf;
mô hình tốt nhất = [];
cuối cùng = [];
max_inlier = 0;
trong khi (lần lặp <K)
        % bắt đầu với NUM_START_VALUES giá trị duy nhất
```

```
        uniqueValues  = [];
        max_index = size (all_matches, 2);
        while (length (uniqueValues) <NUM_START_VALUES)
            value = ceil (max_index * rand (1,1));
            if (length (find (value == uniqueValues)) == 0)
                % giá trị khác 0 duy nhất
                uniqueValues  = [giá trị uniqueValues];
            kết thúc
        kết thúc
        % uniqueValues  là các chỉ số trong all_matches
        có thểinlier = all_matches (:, uniqueValues); % bắt đầu với NUM_START_VALUES giá trị ngẫu nhiên
duy nhất
        M_maybemodel = getModel (có thể, khung_a1, khung_a2); if (prod
        (size (M_maybemodel)) == 0)
            lần lặp lại = iteration + 1;
            tiếp tục;
kết thúc
cũng có = [];
% tìm ra các nội dung khác
cho i = 1: size (all_matches, 2)
    temp = find (all_matches (1, i) == mightinlier (1, :)); if
    (length (temp) == 0)
        % điều này có nghĩa là, trỏ không phải trong
        couldinlier a1 = frames_a1 (1: 2, all_matches (1,
        i)); a2 = frames_a2 (1: 2, all_matches (2, i));
        if (getError (M_maybemodel, a1, a2) <ERROR_THRESHOLD)
            alsoinlier = [alsoinaries all_matches (:, i)];
        kết thúc
    kết thúc
kết thúc
% xem mô hình tốt như thế nào
% fprintf ('Số phần tử trong các phần tử cũng nằm trong số các phần tử% d \ n', size (các phần tử cũng trong các
phần tử, 2)); if (size (alsoinlier, 2)> D)
    % điều này ngụ ý rằng chúng tôi đã tìm thấy một mô hình tốt% bây
    giờ hãy xem nó tốt như thế nào
    % tìm thấy mô hình mới
    all_inlier = [có thể còn các yếu tố khác];
    M_bettermodel = getModel (all_inlier, khung_a1, khung_a2); %
    mô hình mới có thể là xấu
    if (prod (size (M_bettermodel)) == 0)
```

```
                lần lặp lại = iteration + 1;
                tiếp tục;
            kết thúc
            % tìm thấy lỗi cho mô hình
            thiserror = getModelError (M_bettermodel, all_inlier, frames_a1, frames_a2);
            nếu max_inlier <size (all_inlier, 2) | (thiserror <besterror & max_inaries == size
(all_inlier, 2))
                bestmodel = M_bettermodel;
                besterror = thiserror;
                cuối cùng = tất cả các số lượng;
                max_iniers = size (final_inlier, 2);
            kết thúc
        kết thúc
        % làm điều đó K lần
        lần lặp lại = iteration + 1;
    kết thúc
        if (sản phẩm (kích thước (mô hình tốt nhất)) ~ = 0)
            % một mô hình đã được tìm thấy
            fprintf ('Lỗi của best_model ~% f pixel \ n', besterror);
            cờ = 1;
          khác
            cờ = -1;
            cuối cùng = [];
            mô hình tốt nhất = [];
            fprintf ('Không tìm thấy mô hình tốt! \ n');
        kết thúc
    kết thúc
    function error = getModelError (M, so khớp, khung_a1, khung_a2)
            nummatches = size (so khớp, 2);
            lỗi = 0;
              cho i = 1: nummatches
                    a1 = frames_a1 (1: 2, so khớp (1, i));
                    a2 = frames_a2 (1: 2, so khớp (2, i));
                    error = error + getError (M, a1, a2);
            kết thúc
            error = error / nummatches;
    kết thúc
    function P = getModel (so khớp, khung_a1, khung_a2)
        % đi từ 1 đến 2
        % hãy sử dụng cách tiếp cận bình phương nhỏ nhất. Được tham chiếu từ
```

```
    % http: //alumni.media.mit.edu/~cwren/interpolator
    nummatches = size (so khớp, 2);
    LHS = [];
    cho i = 1: nummatches
        a1 = frames_a1 (1: 2, so khớp (1, i)); x = a1 (1); y = a1 (2); a2
        = frames_a2 (1: 2, so khớp (2, i)); X = a2 (1); Y = a2 (2);
        LHS = [LHS; xy 1 0 0 0 -X * x -X * y; 0 0 0 xy 1 -Y * x -Y * y];
    kết thúc
    RHS = reshape (frames_a2 (1: 2, so khớp (2, :)), nummatches * 2, 1); P =
    reshape ([(LHS \ RHS); 1], 3,3) ';
    % để lấy P ở dạng
%       [abc; phản đối; gh 1];
kết thúc
lỗi hàm = getError (P, a1, a2)
    % mô hình F đi từ hình ảnh 1 đến hình ảnh
    2% điểm tương ứng cho a1
    temp = P * [a1; 1];
    a2_model (1) = temp (1) / temp (3);
    a2_model (2) = temp (2) / temp (3);
    error = dist (a2_model ', a2);
kết thúc
 hàm d = dist (một, hai)
    d = sqrt (sum ((một-hai). ^ 2));
kết thúc
```

8.4 Ghép ảnh toàn cảnh

Ghép ảnh toàn cảnh nhằm mục đích ghép nối liền mạch trên chuỗi ảnh được chụp từ cùng một cảnh, góc nhìn khác nhau, độ dài tiêu cự khác nhau, từ cùng một trung tâm quang học với một phần chồng lên nhau. Điều này có nghĩa là thuật toán đăng ký hình ảnh được sử dụng để tính toán các thông số chuyển động giữa mỗi khung hình và sau đó tổng hợp một hình ảnh góc rộng tĩnh lớn. Hơn nữa, hình ảnh đường khâu yêu cầu phải gần như cảnh thật, không có đường may rõ ràng. Theo các quan điểm khác nhau, ghép ảnh có thể được chia thành thuật toán dựa trên một góc nhìn và thuật toán dựa trên nhiều góc nhìn. Để có được chuỗi hình ảnh điểm nhìn đơn lẻ, một máy ảnh được bố trí ở một vị trí và xoay nó xung quanh; hoặc, đặt máy ảnh theo hình tròn, trục quang học của máy ảnh nằm trên cùng một mặt phẳng và giao với một điểm và video được thu theo thời gian thực.

ở các vị trí khác nhau và chụp cùng một lúc. Thuật toán ghép ảnh dựa trên nhiều điểm nhìn thường được sử dụng để ghép ảnh toàn cảnh theo dải.

Phần này giới thiệu thuật toán ghép ảnh dựa trên phép biến đổi phép chiếu ảnh mà không có đối tượng hoạt động.

Thông tin hình ảnh rời rạc chỉ có thể thể hiện thông tin trên một phần của môi trường trực quan. Chế độ xem toàn cảnh dựa trên kết xuất hình ảnh là hiển thị thông tin hình ảnh rời rạc trong một hình ảnh hoàn toàn. Xây dựng một môi trường đồ họa hoàn chỉnh để có hiệu ứng hình ảnh 3D tốt hơn.

(1) Định vị hình ảnh

Tự động tìm các vị trí chồng chéo cho hình ảnh. Đề xuất có hai trực tràng-vùng gle A và B. B chứa một khu vực A_2. A_2 và A là cùng một mô-đun, vị trí của A_2 trong B là để giải quyết. Thuật toán điển hình là tìm kiếm từ góc dưới bên trái của B, trong đó mỗi mảnh được so sánh với a có cùng diện tích C và A, và giá trị của hàm đánh giá, diện tích nhỏ nhất là A_2.

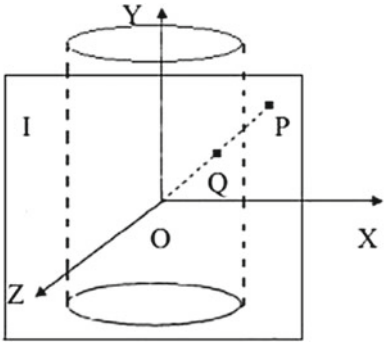
(2) Khâu hình ảnh

Sau khi định vị hình ảnh, nếu ghép đơn giản hai hình ảnh sẽ có một đường nối rõ ràng do sự chênh lệch về độ sáng. Phương pháp thử màu có thể được sử dụng để điều hòa độ sáng của các hình ảnh liền kề và tạo ra hình ảnh tổng hợp liền mạch.

(3) Thực hiện phép chiếu hình trụ

Được chụp tại cùng một điểm với camera quay, các hình ảnh toàn cảnh hình trụ không nằm trong cùng một hệ tọa độ. Có một góc nhất định trên mặt chiếu. Để tạo ra một hình ảnh toàn cảnh, chúng ta phải biến đổi những hình ảnh này thành một hệ tọa độ hình trụ duy nhất và sử dụng công nghệ ghép hình ảnh để loại bỏ sự chồng chéo của mỗi hai hình ảnh. Bằng cách này, một bức tranh toàn cảnh hình trụ hoàn chỉnh sẽ thu được.

Như thể hiện trong Hình. 8.13 , đó là sơ đồ hình chiếu dương của bề mặt hình trụ. *Tôi* là một khung được trích xuất từ video, P là bất kỳ điểm nào trên hình ảnh đã chụp, Q là điểm mà P bản đồ đến tọa độ hình trụ.



Hình 8.13 Sơ đồ hình chiếu dương của bề mặt hình trụ

Giả sử rằng W và H là tầm nhìn của hình ảnh *Tôi* tương ứng, f là bán kính của hình trụ, sao cho P Tọa độ của trong hệ tọa độ 3D được biểu diễn dưới dạng ($x - W/2$, $y - H/2$, f). Sử dụng sự kết hợp của phương trình tham số và phương trình hình trụ, giả sử Q tọa độ của là x (x', y', z') và P cũng như Q nằm trên cùng một đường thẳng, thỏa mãn phương trình tham số:

$$\begin{cases} x' & t(x - W/2) \\ y' & t(y - H/2) \\ z' & tf \end{cases} \tag{8.14}$$

Ở đâu t là tham số, cùng với phương trình bề mặt hình trụ: $x'^2 + z'^2 = f^2$.

Do đó, chúng ta có thể nhận được tọa độ của Q điểm vì tọa độ của Q điểm là ba chiều, chúng tôi chuyển đổi nó thành hai chiều để nhận được:

$$\begin{cases} x' & f * \arctan[(x - W/2)/f] + f * \arctan(W/2f) \\ y' & f(y - H/2)/\sqrt{(x - W/2)^2 + f^2} + H/2 \end{cases} \tag{8.15}$$

Sau khi chiếu ảnh xuống mặt phẳng trụ ta thu được các ảnh của cùng một hệ trục tọa độ. Sau đó, bằng cách tìm kiếm sự chuyển đổi giữa các hình ảnh liền kề, các hình ảnh chuỗi được nối với nhau để tạo thành một hình ảnh toàn cảnh hình trụ trong cùng một cảnh.

Các bước sử dụng phương pháp IBR để ghép hình ảnh toàn cảnh video như sau:

(1) Trích xuất các khung chính của video và sử dụng hình ảnh để thể hiện thông tin trong video.

(2) Tìm ra vùng chồng chéo của hình ảnh, điều này có nghĩa là trích xuất vị trí của đối tượng địa lý.

(3) Đăng ký hình ảnh, điểm tính năng phù hợp. Sử dụng thuật toán đối sánh để loại bỏ các cặp điểm sai và các cặp điểm góc di động. Chức năng chuyển đổi tọa độ có được bằng cách tính toán ma trận chuyển đổi giữa hình ảnh dữ liệu và hình ảnh được so khớp. Cuối cùng, chức năng biến đổi tọa độ được sử dụng để chuyển đổi hình ảnh sang hệ tọa độ dữ liệu và nhận ra việc đăng ký hình ảnh được khớp với hình ảnh dữ liệu trong cùng một hệ tọa độ.

(4) Bước cuối cùng là ghép ảnh, liên quan đến việc hợp nhất hai ảnh và loại bỏ các đường nối.

Mã để đọc video và trích xuất các khung hình chính được hiển thị trong CHƯƠNG TRÌNH 8.8.

CHƯƠNG TRÌNH 8.8: Đọc video và trích xuất các khung chính

```
xyloObj = VideoReader ('test.avi');% đọc video
nFrames = xyloObj.NumberOfFrames;          % số khung hình
đếm = 1;          % đếm khung hình được trích xuất
chữ cái = 'a'; %nhãn ， để làm cho các chuỗi đọc tiếp theo trở nên bình thường, hãy đặt trước các chữ số Ả Rập bằng các chữ cái
cho k = 1: 5: nFrames
    mov (k) .cdata = read (xyloObj, k);          % dữ liệu màu hình ảnh
    strtemp = strcat ('hình ảnh /', ký tự + số lượng / 10);
    strtemp = strcat (strtemp, int2str (số đếm));
    strtemp = strcat (strtemp, '. jpg');
    count = count + 1;
    imwrite (mov (k) .cdata, strtemp);% lưu dưới dạng strtemp.jpg
kết thúc
% phương pháp chuyển đổi tính năng dựa trên màu sắc —— sử dụng biểu đồ màu để đo tính năng màu% bật
lên một vài khung hình chính
filenames = dir ('images / *. jpg');          % nguồn hình ảnh
num = size (tên tệp, 1);                    % số lượng hình ảnh
key = số không (1, num);                        % (0,1) mảng khung chính%
đếm = 0;                                        lưu một vài khung chính%
ngưỡng = 0,75;                                  đặt ngưỡng
nếu num == 0
    error ('Xin lỗi, không có hình ảnh nào trong thư mục hình ảnh!');
khác                                            % đặt khung đầu tiên làm khung chính
    img = imread (strcat ('images /', filenames (1) .name));
    phím (1) = 1;
    count = count + 1;
    % có được biểu đồ RGB
    [preCountR, x] = imhist (img (:,:, 1));          % biểu đồ màu đỏ
    [preCountG, x] = imhist (img (:,:, 2));          % biểu đồ màu xanh lá cây
    [preCountB, x] = imhist (img (:,:, 3));          % biểu đồ màu xanh lam
    % hiển thị con số khung hình
    chính đầu tiên (số lượng);
    imshow ('hình ảnh / a1.jpg');
    cho k = 2: num
        img = imread (strcat ('images /', filenames (k) .name));
        [newCountR, x] = imhist (img (:,:, 1));          % biểu đồ màu đỏ
        [newCountG, x] = imhist (img (:,:, 2));          % biểu đồ màu xanh lá cây
        [newCountB, x] = imhist (img (:,:, 3));          % biểu đồ màu xanh lam
        sR = 0;
        sG = 0;
        sB = 0;
```

```
% sử dụng phương pháp biểu đồ màu
cho j = 1: 256
    sR = min (preCountR (j), newCountR (j)) + sR;
    sG = min (preCountG (j), newCountG (j)) + sG;
    sB = min (preCountB (j), newCountB (j)) + sB;
kết thúc
dR = sR / sum (newCountR);
dG = sG / sum (newCountG);
dB = sB / sum (newCountB);
            % YUV, những người nhạy cảm với Y d =
            0,30 * dR + 0,59 * dG + 0,11 * dB;
            nếu d <ngưỡng                % điểm tương đồng nhỏ, đã tìm thấy khung hình chính mới
                phím (k) = 1;                % đặt làm khung hình chính
                count = count + 1;
                hình (số đếm);
                imshow (strcat ('hình ảnh /', tên tệp (k) .name));
                % biểu đồ màu cập nhật gần nhất
                preCountR = newCountR;
                preCountG = newCountG;
                preCountB = newCountB;
            kết thúc
        kết thúc
kết thúc
keyFrameIndexes = find (key)
```

Ghép ảnh toàn cảnh dựa trên phương pháp IBR sử dụng các khung hình chính, có mã là chương trình 8.9 và các chức năng liên quan được mô tả trong chương trình 8.3.

CHƯƠNG TRÌNH 8.9: Ghép ảnh toàn cảnh dựa trên phương pháp IBR

```
rõ ràng; clc;
%%%%%%%%%%%%%%%%%% hình ảnh đầu vào, kiến   tạo hình kim tự tháp, lưu một mảng tic

fprintf ('hình ảnh đầu vào, hình ảnh kiến   tạo kim tự tháp, ...');
cấp = 3;
T0 =   uint8 ([]); T = uint8 ([]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%
i1 = imread ('1.jpg'); T0 (:,:,:, 1) = i1;
i2 = imread ('2.jpg'); T0 (:,:,:, 2) = i2;
i3 = imread ('3.jpg'); T0 (:,:,:, 3) = i3;
```

```
i4 = imread ('4.jpg'); T0 (:,:,:, 4) = i4;
i5 = imread ('5.jpg'); T0 (:,:,:, 5) = i5;
i6 = imread ('6.jpg'); T0 (:,:,:, 6) = i6;
i7 = imread ('7.jpg'); T0 (:,:,:, 7) = i7;
i8 = imread ('8.jpg'); T0 (:,:,:, 8) = i8;
[h, w, d] = size (T0 (:,:,:, 1)); %%% yêu cầu cùng kích thước hình
ảnh% [T, Terr] = multi_resolution (T0, level);
T = multi_resolution (T0, cấp độ);
toc
% %%%%%%%%%%%%%%%%%%%%%%% tính toán bù đắp thông qua tương quan pha tic

fprintf ('tính toán độ lệch thông qua tương quan pha
..'); M = 8; %% số hình ảnh
% L = M * w; %% tổng độ dài của hình ảnh
cho N = 1: M
    if N <M [i, j] = poc_2pow (T (:,:,:, N), T (:,:,:, N + 1));
    elseif N == M [i, j] = poc_2pow (T (:,:,:, N), T (:,:,:, 1));
    kết thúc
    coor_shift (N, 1) = i;
    coor_shift (N, 2) = j;
kết thúc
coor_shift = coor_shift * 2 ^ level; %%% chuyển đổi các hiệu số trong phân cấp kim tự tháp thành bù đắp
  nguyên
toc
%%%%%%%%%%%%%%%%%% Chuyển sang hệ tọa độ trụ tic

fprintf ('Biến đổi sang hệ tọa độ trụ ..'); f = sqrt (h ^ 2
+ ŵ 2);
[T1, coor_shift02] = coortransf (T0, f, coor_shift);
toc
%%%%%%%%%%%%% hợp nhất các phần chồng chéo tic

fprintf ('ghép các phần chồng lên nhau, ghép ảnh ....');
panorama1 = T1 (:,:,:, 1);
cho N = 1: M
    nếu N <M
panorama1 = mosaic (panorama1, T1 (:,:,:, N + 1), coor_shift02 (N, 1), coor_shift02 (N, 2));
    kết thúc
```

```
kết thúc
toc
%%%%%%%%%%%%%%%%% dựng lại hình ảnh
tic
fprintf ('lưu và hiển thị kết quả ..');
toc
image1 = rgb2gray (panorama1);
index = find (image1 (:, 1)> = 255);
aa = max (chỉ số);
[r, c] = size (image1)
image1 = imcrop (panorama1, [1, aa, c, r]);
imshow (image1);
imwrite (image1, 'pic1.jpg', 'jpg');
```

Các khung chính được trích xuất được hiển thị trong Hình. 8.14 . Hình ảnh đường khâu toàn cảnh được hiển thị trong Hình. 8.15 .

Hình 8.14 Các khung chính được trích xuất

Hình 8.15 Hình ảnh khâu toàn cảnh

Người giới thiệu

1. Yanyan L, Xu S (2008) Nghiên cứu thuật toán ghép ảnh dựa trên đối sánh tỷ lệ. Công nghệ đo điện tử
2. Le-Fu WU, Ding GT (2010) Thuật toán ghép ảnh dựa trên vùng. Comput Eng Design 31 (18): 4043–4044
3. Zhou DF, Ming-Yi HE, Yang Q (2009) Một thuật toán ghép ảnh liền mạch mạnh mẽ dựa trên các điểm đặc trưng. Kiểm soát biện pháp Technol 28 (6): 32–36
4. Chandratre R, Chakkarwar VA (2014) Ghép hình ảnh bằng Harris và RANSAC. Int J Comput Appl 89 (15): 14–19
5. Zhao WJ, Gong SR, Liu Q và cộng sự (2007) Một số học tự động sắp xếp cho chuỗi hình ảnh được sử dụng trong Mosaics hình ảnh. J Đồ thị hình ảnh 12 (10): 1861–1864