ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Học phần: Xử Lý Ảnh Và Video Số

Mã học phần: CSC16005

DÁN ẢNH: STITCHING IMAGES

GIẢNG VIÊN HƯỚNG DẪN: PGS. TS LÝ QUỐC NGỌC

SINH VIÊN THỰC HIỆN:

- 1712641 TRẦN NGUYỄN NHU
- 1712787 NGUYỄN VĂN THÌN
- 1712713 LÊ BÁ QUYỀN
- 1712770 TRƯƠNG THỊ LỆ THANH
- 1712771 BÙI THÁI TẤN THÀNH

TP. HỒ CHÍ MINH, NGÀY 28 THÁNG 12 NĂM 2019

1. GIỚI THỆU ĐỀ TÀI

1.1 TỔNG QUAN VỀ DÁN ẢNH (STITCHING IMAGES)

Ghép ảnh là quá trình kết hợp nhiều hình ảnh nhỏ xếp chồng lên nhau để tạo ra một bức ảnh lớn có độ phân giải cao hơn. Thông thường việc ghép ảnh được thực hiện bằng việc sử dụng các phần mềm máy tính. Ghép ảnh có rất nhiều ứng dụng khác nhau. Ứng dụng truyền thống nhất là tạo nên ảnh không gian rộng và ảnh vệ tinh từ một tập các ảnh, dùng để xây dựng bản đồ địa lý, ghép các tấm ảnh chụp được trên bề mặt của một ngôi sao thành một tấm ảnh có độ phân giải lớn hơn, vv… Các vấn đề chính trong ghép ảnh là sắp xếp các ảnh thành phần, nắn chỉnh biến dạng, biến đổi màu sắc và làm mờ đường biên giữa các ảnh. Tất cả các thao tác này nhằm làm cho bức ảnh ghép trông giống như là một ảnh liền chứ không phải là được ghép từ nhiều ảnh nhỏ.

1.2 YÊU CẦU ĐỀ TÀI

Input: Các ảnh có liên quan với nhau cần khâu lại thành ảnh lớn

Output: Một phân đoạn ảnh toàn cảnh (panorama) được khâu từ các ảnh đầu vào

2. MỘT SỐ CÔNG TRÌNH NGHIÊN CỨU LIÊN QUAN

2.1 Ổn định hình ảnh

Là một nhóm các kỹ thuật làm giảm mờ liên quan đến chuyển động của máy ảnh hoặc thiết bị hình ảnh khác trong khi phơi sáng.

Figure 1. So sánh các bức ảnh cận cảnh của bàn phím máy tính có và không có ổn định hình ảnh quang học

2.2 Photomosaics

Trong lĩnh vực hình ảnh nhiếp ảnh, thuật ngữ **Photomosaic** là một hình ảnh đã được chia thành phần nhỏ(lát gạch), mỗi phần trong số đó là thay thế bằng một bức ảnh khác phù hợp với ảnh mục tiêu.[1] Khi được xem ở độ phóng đại thấp, các pixel riêng lẻ xuất hiện dưới dạng hình ảnh chính, trong khi kiểm tra kỹ cho thấy hình ảnh trên thực tế được tạo thành từ hàng trăm hoặc hàng nghìn hình ảnh nhỏ hơn

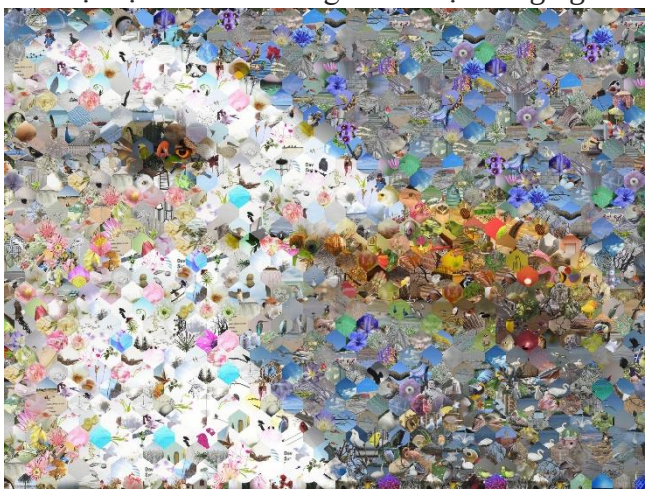


Figure 2. Một bức ảnh khảm của một con mòng biển được làm từ hình ảnh của các loài chim và các bức ảnh thiên nhiên khác bằng cách sử dụng gạch hình lục giác

2.3 Hình ảnh y tế

Đôi khi, sử dụng một hình ảnh X quang là không đủ để phát hiện bất kỳ sự bất thường nào trong cơ thể con người. Để tạo ra hình ảnh chứa toàn bộ các bộ phận cơ thể, có thể lắp ráp hình ảnh số hóa từ các băng cassette và phim có chứa các phần của các bộ phận cơ thể. Ví dụ, trong chụp X quang thông thường, một hình ảnh lớn có thể được lắp ráp từ các hình ảnh X quang của nhiều mức phơi sáng với sự chồng lấp không gian nhỏ. Kỹ thuật này thường được gọi là khâu.[2]

2.4 Hình ảnh siêu phân giải

Hình ảnh siêu phân giải (**SR**) là một lớp các kỹ thuật giúp tăng cường (tăng) độ phân giải của hệ thống hình ảnh.[3]

Figure 3. Super-resolution Imaging

3. CÁC KỸ THUẬT GHÉP ẢNH PANORAMA
GHÉP ẢNH PANORAMA DỰA TRÊN ĐẶC TRƯNG BẤT BIẾN CỦA ẢNH

3.1 Trích chọn đặc trưng bất biến của ảnh

Một trong những phương pháp tìm kiếm nổi bật cơ bản nhất chính là phương pháp tìm kiếm các điểm nổi của đường biên, các điểm nằm trên đường cong mà có độ cong tối đa hay còn gọi là các điểm góc.

Ban đầu các điểm này được phát hiện nhờ vào "độ nhọn" (sharp) của đường biên: biên của đối tượng được lưu dưới dạng mã xích, góc được phát hiện thông qua việc tìm kiếm những vị trí trên biên bị uốn một cách đáng kể. kỹ thuật phát hiện góc này rất phức tạp và triển khai trên nhiều bước.

Thuật toán Harris sử dụng một cửa sổ có thể trượt theo bất ký hướng nào bằng cách sử dụng hàm Gaussian và phép khai triển Taylor.

Về mặt ý tưởng, thuật toán Harris sẽ tìm kiếm sự thay đổi lớn về cường độ xám theo các hướng khác nhau bằng cách dùng một cửa sổ nhỏ để làm nhiệm vụ rà soát và phát hiện những điểm định nghĩa là góc.

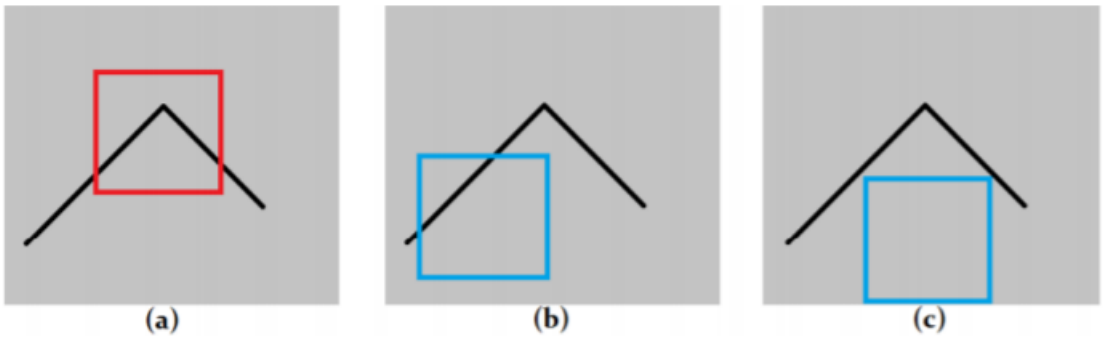


Figure 4. Cửa sổ trượt phát hiện góc Harris

Trong Hình 14(a) cửa sổ trượt nằm trong vùng hình ảnh có chứa góc, khi di chuyển theo bất kỳ hướng nào đều có sự thay đổi về cường độ xám.

Trong Hình 14(b) cửa sổ trượt nằm trên vùng hình ảnh có chứa cạnh, khi di chuyển cửa sổ trượt theo hai hướng của cạnh sẽ không có sự thay đổi về cường độ xám.

Trong Hình 14(c) cửa sổ trượt nằm trên vùng hình ảnh ko có góc cạnh, sau khi di chuyển cửa sổ trượt sẽ ko có sự thay đổi về cường độ xám.

Dựa vào điều này ta có thể phát hiện ra điểm nào là điểm góc, điểm nào không phải.

3.1.1 Thuật toán trích chọn đặc trưng

Giả sử cho một ảnh xám **(I)**, với mỗi điểm **(u, v)** và độ dịch chuyển **(x, y)** ta có thể tính toán được sự thay đổi trung bình cường độ xám bằng một cửa sổ dịch chuyển từ **(u, v)** tới **(u + x, v+ y)** như sau:

$$S(x,y)=\sum_u\sum_v w(u,v)\,(I(u+x,v+y)-I(u,v))^2$$

Trong đó:

S(x, y) là tổng bình phương giá trị độ lệch hay còn gọi là sự thay đổi cường độ xám tại **(x, y)**

w(u, v) là cửa sổ tại **(u, v)**

I(u, v) và **I(u + x, v + y)** là giá trị cường độ xám của pixel tại các vị trí **(u, v)**

và **I(u + x, v + y)**

Giá trị **I(u + x, v + y)** có thể được khai triển theo công thức Taylor như sau:

$$I(u+x,v+y)\approx I(u,v)+I_x(u,v)x+I_y(u,v)y$$

Với Ix, Iy là đạo hàm theo thành phần x, y

Từ đó, có thể được viết lại như sau:

$$S(x,y)=\sum_u\sum_v w(u,v)\left(I_x(u,v)x-I_y(u,v)y\right)^2$$

Biểu diễn dưới dạng ma trận thì S(x, y) ta có:

$$S(x,y)\approx(x,y)\,A\begin{pmatrix}x\\y\end{pmatrix}$$

Trong đó, A là một cấu trúc như sau:

$$S(x,y)=\sum_u\sum_v w(u,v)\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix}=\begin{bmatrix}\langle I_x^2\rangle & \langle I_xI_y\rangle\\ \langle I_xI_y\rangle & \langle I_y^2\rangle\end{bmatrix}$$

Gọi **λ1** và **λ2** là các giá trị riêng của **A, k** là hằng số được xác định thông qua thực nghiệm, thường có giá trị trong khoảng **[0. 04, …, 0. 15]**.

Khi đó, biểu thức dưới đây sẽ quyết định xem cử sổ **w** có chứa góc hay là không:

$$M_c=\lambda_1\lambda_2-k(\lambda_1+\lambda_2)^2=\det(A)-k\left(trace^2(A)\right)$$

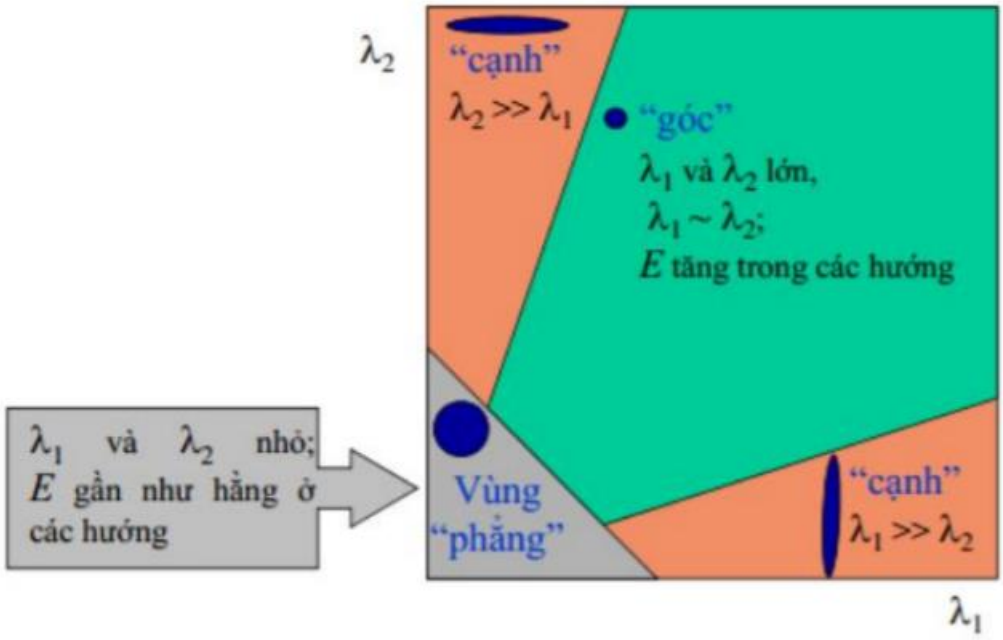


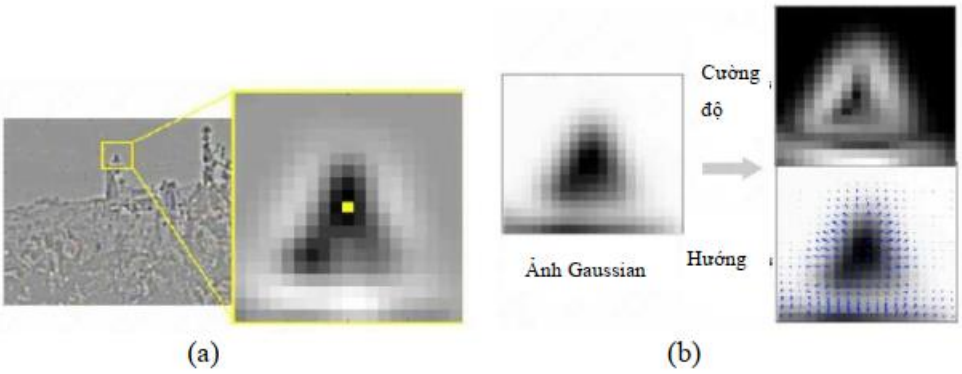
Figure 5. Minh họa các trường hợp λ1 và λ2

Nếu cả **λ1** và **λ2** đều nhỏ. Có nghĩa là hàm **S(x, y)** gần như không thay đổi theo bất kỳ hướng nào. Khi đó vùng ảnh nằm trong cửa sổ gần như không có sự thay đổi về cường độ. Tức là trường hợp này không tìm thấy điểm góc.

Nếu **λ1** là lớn và **λ2** là nhỏ hoặc ngược lại. Có nghĩa là hàm **S(x, y)** có sự thay đổi nhỏ nếu cửa sổ trượt theo một hướng, và có sự thay đổi đáng kể nếu dịch chuyển theo hướng trực giao. Điều này cho thấy tồn tại một cạnh.

Nếu cả **λ1** và **λ2** đều lớn. Có nghĩa là hàm **S(x, y)** có sự thay đổi đáng kể về cường độ xám khi dịch chuyển cửa sổ trượt theo bất kỳ hướng nào. Điều này cho thấy có tồn tại một điểm góc.

3.1.2 Định hướng cho các điểm đặc trưng



Hình 3. Định hướng cho điểm đặc trưng [3].
a) Vùng cực trị và b) Xác định cường độ và hướng.

Các bước được trình bày trong Hình 3. Ở Hình 3 (a) vị trí điểm đặc trưng được xác định là

vùng cực trị. Trong Hình 3 (b), chúng tôi tính toán hướng và độ lớn của ảnh Gaussian:

$$m(x,y)=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2}$$

$$\theta(x,y)=\tan^{-1}\left(\frac{\left(L(x,y+1)-L(x,y-1)\right)}{\left(L(x,y+1)-L(x-1,y)\right)}\right)$$

Với: **M(x, y)** là độ lớn của vector định hướng.

ɵ(x, y) là hướng của vector định hướng (biểu diễn qua góc ɵ)

L(x, y) là ảnh Gaussian ở tỷ lệ nhỏ nhất

3.1.3 Mô tả các điểm đặc trưng

Các phép xử lý trên đã thực hiện dò tìm và gán tọa độ, kích thước và hướng cho mỗi điểm đặc trưng. Các tham số đó yêu cầu một hệ thống tọa độ cục bộ 2D có thể lặp lại được để mô tả vùng ảnh cục bộ và nhờ vậy tạo ra sự bất biến đối với các tham số đó. Bước này sẽ tính toán một bộ mô tả cho một vùng ảnh cục bộ mà có tính đặc trưng cao (bất biến với các thay đổi khác nhau về độ sáng, thu – phóng ảnh, xoay).

Một cách tiếp cận đơn giản đó là lấy mẫu mật độ ảnh cục bộ lân cận điểm đặc trưng ở một độ đo thích hợp, và đối sánh các mật độ này sử dụng độ đo tương quan chuẩn.

Cách tiếp cận tốt hơn được đưa ra bởi Edelman, Intrator và Poggio (1997). Cách tiếp cận này dựa trên một mô hình thị giác sinh học, cụ thể là mô hình noron phức tạp trong hệ thống não bộ. Các noron sẽ tương ứng với một gradient tại một hướng và tần số không gian cụ thể, nhưng vị trí của gradient trên võng mạc được phép trượt trên một phạm vi nhỏ của khung nhìn.

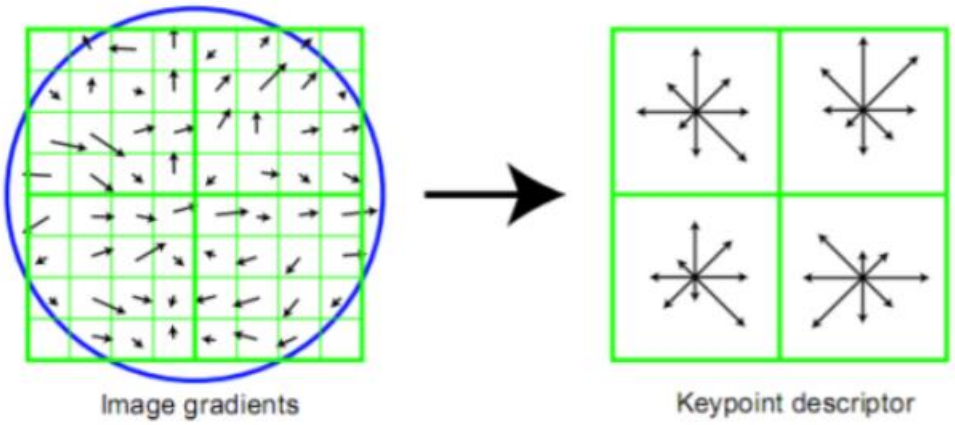


Figure 6. Bộ mô tả cục bộ

Ảnh trái là mô phỏng biên độ gradient và hướng tại mỗi mẫu ảnh trong một vùng lân cận với điểm nổi bật. Các giá trị đó tập trung trong một cửa sổ Gaussian (nằm bên trong vòng tròn). Các mẫu này sau đó được gom lại thành một lược đồ hướng mô tả vắn tắt nội dung

trong 4x4 vùng con như được mô tả bên phải với độ dài của mỗi hàng tương ứng với tổng biên độ gradient gần hướng đó bên trong một vùng.

Điểm hấp dẫn sau khi được xác định hướng sẽ được biểu diễn dưới dạng các vector 4x4x8 = 128 chiều (Số chiều = 8 hướng x (4x4) điểm hấp dẫn = 128 chiều) bằng cách tổng hợp các vector định hướng của các điểm trong khu vực, các vector này có đặc điểm:

– Chung gốc.
– Độ dài mỗi vector tương ứng với độ lớn gradient m của nó.

3.2 Đối sánh các đặc trưng bất biến

3.2.1 Độ đo khoảng cách và độ đo tương tự

Độ đo tương tự là một trong những phương pháp tốt để máy tính phân biệt được các hình ảnh qua nội dung của chúng. Việc đối sánh theo nội dung sẽ truy vấn hình ảnh bằng phương pháp đo tương tự dựa trên các đặc trưng, việc xác định nó có thể dưới nhiều hình thức như phát hiện biên, màu sắc, vị trí điểm ảnh …, các phương pháp như histogram, màu sắc và phân tích sử dụng biểu đồ để xác định độ tương tự.

Do đó, độ đo có ý nghĩa quan trọng trong đối sánh ảnh dựa trên nội dung. Độ đo mang ý nghĩa quyết định kết quả đối sánh sẽ như thế nào, mức độ chính xác ra sao. Nhiều phép đo khoảng cách đã được khai thác trong việc đối sánh ảnh bao gồm: khoảng cách Euclide, khoảng cách Cousin, khoảng cách giao nhau của biểu đồ histogram, khoảng cách Minkowshi ….

3.2.2 Đối sách đặc trưng cục bộ bất biến

Việc đối sánh sẽ được thực hiện trên các tập keypoint tìm được. Bước chính trong kỹ thuật đối sánh sẽ thực hiện tìm tập con keypoint so khớp nhau ở hai ảnh, để thực hiện việc này sẽ tìm các cặp keypoint trùng nhau lần lượt ở hai ảnh. Tập con các keypoint so khớp chính là vùng ảnh tương đồng. Việc đối sánh hai tập đặc trưng quy về bài toán tìm láng giềng gần nhất của mỗi điểm đặc trưng.

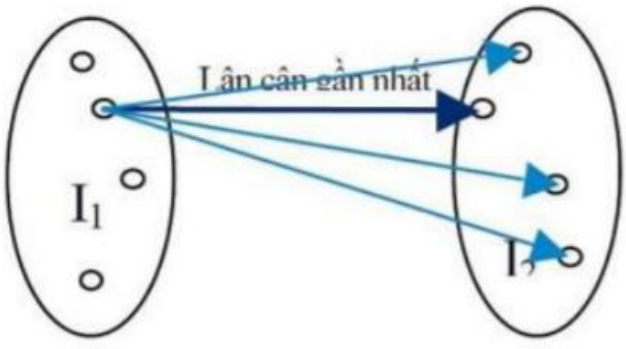


Figure 7. Ví dụ về đối sánh hai tập đặc trưng

Có 2 vấn đề cần được quan tâm:

– Tổ chức tập hợp điểm cho phép tìm kiếm láng giềng một cách hiệu quả.

– Việc đối sánh phải đạt độ chính xác nhất định.

Một phương pháp được đề xuất bởi D. Mount cho phép tìm kiếm nhanh các điểm lân cận được sử dụng, ANN được viết tắt của Approximative Nearest Neibour. Nó cho phép tổ chức dữ liệu dưới dạng kd-tree. Cụ thể là hai điểm trong không gian đặc trưng được coi là giống nhau nếu khoảng cách Euclide giữa hai điểm là nhỏ nhất và tỷ số giữa khoảng cách gần nhất với khoảng cách gần nhì phải nhỏ hơn một ngưỡng nào đó.

Giả sử cặp keypoint có bộ mô tả lần lượt là:

$A = (a_1, a_2, a_3, \ldots, a_{128})$ và $B = (b_1, b_2, b_3, \ldots, b_{128})$

Thì khoảng cách Euclide giữa A và B được tính bằng công thức: $D(A, B) = \Sigma(ai - bi)^2$

3.3 Tính toán ma trận Homography

3.3.1 Vài nét về Homography

Trong toán học, Homography là sự dịch chuyển sử dụng phép chiếu hình học, hay nói cách khác nó là sự kết hợp của cặp điểm trong phép chiếu phối cảnh. Ảnh thực trong không gian ba chiều có thể biến đổi về không gian ảnh bằng phép chiếu thông qua ma trận biến đổi Homography hay còn gọi là ma trận H. Các phép chiếu biến đổi thông qua ma trận Homography không đảm bảo về kích thước và góc của vật được chiếu nhưng lại đảm bảo về tỷ lệ.

Trong lĩnh vực thị giác máy, Homography là một ánh xạ từ mặt phẳng đối tượng đến mặt phẳng ảnh. Ma trận homography thường có liên quan đến các công việc xử lý giữa hai ảnh bất kỳ và có ứng dụng rất sộng rãi trong các công tác sửa ảnh, ghép ảnh, tính toán sự chuyển động, xoay hay dịch chuyển giữa hai ảnh.

Ta có công thức sau:

$$\mathbf{HX = sX'}$$

Trong đó:

S là hằng số tỉ lệ của phép chiếu và khác 0
X’ là kết quả của phép ánh xạ
H là ma trận Homography, là một ma trận khả nghịch
Vì H là một ma trận khả nghịch, cho nên trong trường hợp muốn tái tạo ảnh X từ X’, chỉ cần xác định được ma trận Homography là được.

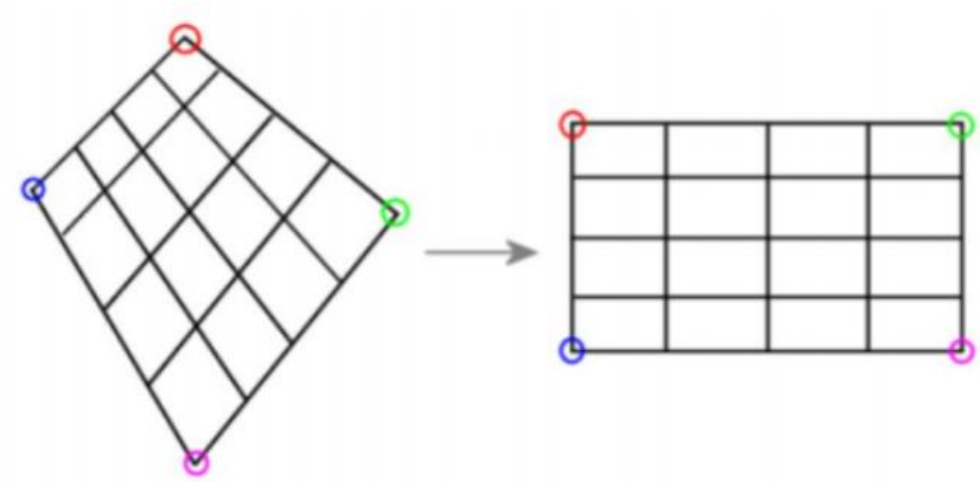


Figure 8. Phép chiếu Homography

3.3.2 Tính toán Homography

Homography là một định nghĩa toán học. Đó là sự dịch chuyển sử dụng phép chiều hình học, hay nói cách khác nó là sự kết hợp của cặp điểm trong phép chiếu phối cảnh. Ảnh thực trong không gian ba chiều có thể biến đổi về không gian ảnh bằng phép chiếu thông qua ma trận biến đổi Homography H. Các phép chiếu biến đổi thông qua ma trận Homography tuy không đảm bảo về kích thước và góc của vật được chiếu, nhưng đảm bảo về tỷ lệ.

Tính ma trận Homography bằng phương pháp Direct Linear Transform

Để tính ma trận Homography từ các cặp điểm tương ứng, người ta dùng phương pháp DLT (Direct Linear Transform), gồm có 2 bước: Đầu tiên, từ các cặp điểm tương ứng, ta chuyển về dạng ma trận $\mathbf{A_i h = 0}$. Sau đó, áp dụng phân rã SVD để tính ma trận **H**.

Giải thuật SVD hay được gọi là giải thuật phân rã giá trị đơn (Singular Value Decomposition) được Golub và Kahan công bố năm 1965 [5], đó là một kỹ thuật phân rã được sử dụng để giảm hạng (hay số chiều) của ma trận. SVD cho phép phân tích một ma trận phức tạp thành ba ma trận thành phần. Mục đích nhằm đưa ra việc giải quyết bài toán liên quan đến ma trận lớn, phức tạp về những bài toán nhỏ hơn.

$$A = USV^T$$

Trong đó:

U là ma trận trực giao cấp m*n (m số từ chỉ mục) các vector dòng của U là các vector từ chỉ mục.
S là ma trận đường chép cấp r*r có giá trị suy biến σ1≥ σ2≥ …≥ σr với r= rank(A)
V là ma trận trực giao cấp r*n – các vector cột của V là các vector văn bản
Hạng của ma trận A là các số dương trên đường chéo của ma trận S.
Phương pháp DLT với các điểm nổi bật được tìm thấy từ thuật toán Harris:
Trong tọa độ không đồng nhất, ta có công thức:

$$c\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Lần lượt chia dòng thứ nhất của công thức trên cho dòng thứ ba và dòng thứ hai cho dòng thứ ba, ta có:

$$-h_1x - h_2y - h_3 + \left(h_7x + h_8y + h_9\right)_u = 0$$

$$-h_4x - h_5y - h_6 + \left(h_7x + h_8y + h_9\right)_u = 0$$

Viết dưới dạng ma trận ta có:

$$A_ih = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & v \end{bmatrix} \left(h_1h_2 \ldots h_9\right)^T$$

Áp dụng công thức phân rã SVD cho ma trận **[A]** ta có:

$$A = U\sum V^T = \sum_{i=1}^{9} s_iu_iv_i^T$$

Với si là các giá trị đơn và được sắp xếp nhỏ dần, nên s9 là giá trị nhỏ nhất. Khi đó, giá trị của hi bằng giá trị cuối cùng của cột vi.

Trong thực tế các ảnh đầu vào có thể có gốc tọa độ ở góc trái của ảnh, cũng có thể gốc tọa độ nằm ở tâm ảnh. Nếu để tình trạng như vậy sẽ ảnh hưởng đến các kết quả biến đổi về sau như khi nhân ảnh với một hệ số hay các biến đổi tương tự affin. Các ảnh cần phải chuẩn hóa bằng phép biến đổi quay và dịch chuyển.

Thuật toán RANSAC [7]

RANSAC (RANdom SAmple Consensus) được công bố bởi Fischler và Boller vào năm 1981. Ý tưởng chính của RANSAC như sau: Từ tập dữ liệu ban đầu, ta sẽ có hai loại dữ liệu nhiễu và không nhiễu (outlier và inlier), vì thế ta phải đi tính toán để tìm ra mô hình tốt nhất cho tập dữ liệu. Việc tính toán và chọn ra mô hình tốt nhất sẽ được lặp đi lặp lại k lần, với giá trị k được chọn sao cho đủ lớn để đảm báo xác suất p (thường rơi vào giá trị 0. 99) của tập dữ liệu mẫu ngẫu nhiên không chứa dữ liệu nhiễu.

Quá trình thực hiện thuật toán RANSAC được mô tả như dưới đây:

Từ tập dữ liệu đầu vào gồm có nhiễu và không có nhiễu ta chọn từ n dữ liệu ngẫu nhiên, tối thiểu để xây dựng mô hình:

– Tiến hành xây dựng mô hình với n dữ liệu đó, sau đó ra một ngưỡng dùng để kiểm chứng mô hình.

– Gọi tập dữ liệu ban đầu trừ đi tập n dữ liệu để xây dựng mô hình tập dữ liệu kiểm chứng. Sau đó, tiến hành kiểm chứng mô hình đã xây dựng bằng tập dữ

liệu kiểm chứng. Nếu kết quả thu được từ mô hình vượt quá ngưỡng, thì điểm đó là nhiễu, còn không đó sẽ là ngược lại.
– Quá trình này sẽ được lặp lại trong k lần. Với k được tính theo công thức trên. Tại mỗi vòng lặp giá trị của k sẽ được tính lại.
– Kết quả là mô hình nào có số dữ liệu không nhiễu nhiều nhất sẽ được chọn là mô hình tốt nhất.

Trong bài toán tạo ảnh Panorama, ma trận Homography được tính từ tập các cặp điểm nổi bật tương ứng của hai ảnh ban đầu đã được so sánh đối chiếu ở bước hai. Khi đó bốn cặp điểm nổi bật tương ứng không thẳng hàng, phương trình Ah = 0 theo phương pháp DLT chuẩn hóa đã trình bày ở phần trên. Trong đó, A là ma trận có kích thước 8 * 9. Từ đó, ta xác định được ma trận h.

Với ma trận Homography được tính từ bốn cặp điểm ngẫu nhiên, ta có d là khoảng cách đo mức độ gần nhau của các cặp điểm đã được so sánh đối chiếu. Với cặp điểm nổi bật tương đồng (x, x') và d($\vec{a}$, $\vec{b}$) là khoảng cách của hai vector, ta có công thức khoảng cách như sau:

$$d = d\left(\vec{x}, H\vec{x'}\right) + d\left(\vec{x'}, H\vec{x}\right)$$

Thuật toán chi tiết:

– Khởi tạo số vòng lặp **k**, ngưỡng **distance**, **maxinlier**, và **p = 0.**

– **for(i = 1:k)**, thực hiện các bước sau:

– **Bước 1**: Chọn 4 cặp điểm tương đồng ngẫu nhiên

– **Bước 2**: Kiểm tra xem các điểm có nằm trên cùng một dường thẳng hay ko.

– **Bước 3**: Tính ma trận Homography H từ 4 điểm sử dụng phương pháp DLT chuẩn hóa.

– **Bước 4**: Tính khoảng cách d của các cặp điểm nổi bật tương đồng

– **Bước 5:** Tính số lượng m các cặp điểm không ngẫu nhiên (inlier) thỏa điều kiện: di < distance

– **Bước 6**: Nếu m > maxinlier thi maxinlier = m, ma trận Homography H= Hcurr.

– Tiếp tục tính lại ma trận H cho tất cả các cặp điểm tương đồng được coi là không nhiễu (inlier) bằng phương pháp DLT.

3.4 Ghép hai ảnh dựa trên ma trận Homography

Sau khi ma trận Homography được tính toán, bước cuối cùng trong việc tạo ảnh Panorama là hòa trộn hai bức ảnh lại với nhau. Ý tưởng của bước này là sử dụng một ảnh làm trung tâm, sau đó sử dụng ma trận Homography để chiếu ảnh còn lại tới mặt phẳng ảnh trung tâm.

Như trong hình 2.18, bên trái là ảnh trung tâm được sử dụng là mặt phẳng chiếu. Bên phải là ảnh được chiếu lên mặt phẳng ảnh thứ nhất sử dụng ma trận Homography. Phần màu xám là phần riêng của mỗi bức ảnh, còn phần màu đen là phần chung của hai bức ảnh. Để thực hiện

được kỹ thuật này ta dùng phép chiếu phối cảnh để chiếu các hình ảnh đầu vào ban đầu sử dụng ma trận homography làm ma trận biến đổi để chiếu lên mặt phẳng chiếu.

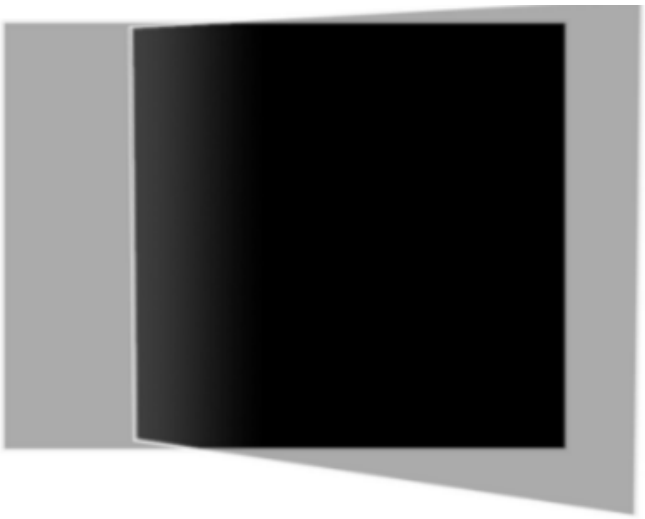


Figure 9. Minh họa ghép nối ảnh

Phép chiếu là phép chuyển đổi những điểm trong hệ tọa độ n chiều thành những điểm trong hệ tọa độ có số chiều ít hơn n. Một điểm P trên đối tượng được xác định trong tọa độ thế giới thực tại vị trí (x, y, z) qua phép chiếu phối cảnh t được điểm P' có tọa độ (x', y', z') nằm trên mặt phẳng chiếu.

4. Cài đặt chương trình và thực thi

Chương trình thực thi chạy trên môi trường python với sự hỗ trợ của thư viện opencv

Cài đặt môi trường python:

```
pip install opencv-contrib-python==3.4.2.16
```

Tiếp đến là them thư viện vào code:

```
import cv2
import numpy as np
```

Danh sách các bước chúng ta nên làm để thực hiện khâu ảnh là:

1. Tính toán các điểm đặc trưng cho hình ảnh bên trái và bên phải.
2. Tính khoảng cách giữa mỗi đặc trưng trong một hình với mọi đặc trưng trong hình ảnh khác.
3. Chọn các kết quả phù hợp nhất cho mỗi mô tả của một hình ảnh.
4. Chạy RANSAC để ước tính homography.
5. Căn chỉnh, blend màu cho stitch.
6. Cuối cùng stitch chúng lại với nhau.

Bắt đầu từ bước đầu tiên, chúng tôi đang nhập hai hình ảnh này và chuyển đổi chúng thành thang độ xám, nếu bạn đang sử dụng hình ảnh lớn, tôi khuyên bạn nên sử dụng cv2.resize vì nếu bạn có máy tính cũ hơn thì có thể rất chậm và mất nhiều thời

gian. Nếu bạn muốn thay đổi kích thước kích thước hình ảnh, tức là bằng 50%, chỉ cần thay đổi từ fx = 1 thành fx = 0,5.

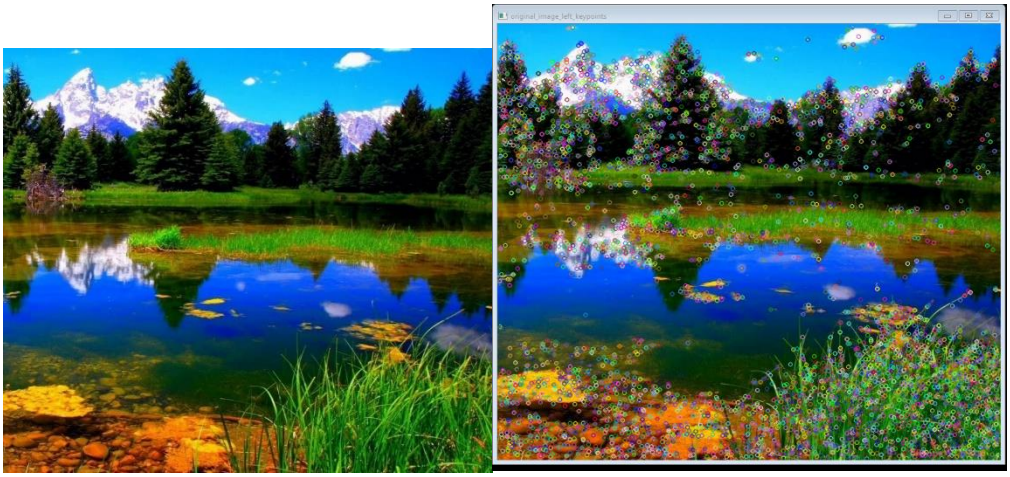
```
img_ = cv2.imread ( 'original_image_left.jpg')
img_ = cv2.resize (img_, (0,0), fx = 1, fy = 1)
img1 = cv2.cvtColor (img_, cv2.COLOR_BGR2GRAY)
img = CV2. imread ('original_image_right.jpg')
img = cv2.resize (img, (0,0), fx = 1, fy = 1)
img2 = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
```

Chúng tôi sẽ sử dụng bộ mô tả Sift của opencv_contrib. SIFT (Scale Invariant Feature Transform) là một thuật toán OpenCV rất mạnh[8]. Những tính năng phù hợp nhất này đóng vai trò là cơ sở để khâu. Chúng tôi trích xuất các điểm chính và mô tả sàng lọc cho cả hai hình ảnh như sau:

```
sift = cv2.xfeatures2d.SIFT_create ()
# tìm các điểm chính và mô tả với SIFT
kp1, des1 = sift.detectAndCompute (img1, none)
kp2, des2 = sift.detectAndCompute (img2, s2)
```

kp1 và kp2 là các điểm chính, des1 và des2 là các mô tả của các hình ảnh tương ứng. Nếu chúng ta vẽ hình ảnh này với các tính năng, thì đây là hình ảnh:

```
cv2.imshow ('original_image_left_keypoint', cv2.drawKeypoint
(img_, kp1, Node))
```



Khi bạn đã có các mô tả và các điểm chính của hai hình ảnh, Sẽ tìm thấy sự tương ứng giữa chúng. Tại sao chúng ta làm việc này ? Để nối bất kỳ hai hình ảnh nào thành một hình ảnh lớn hơn, chúng ta phải tìm các điểm chồng chéo. Những điểm chồng chéo này sẽ cho chúng ta một ý tưởng về sự định hướng của hình ảnh thứ hai theo hình thứ nhất. Và dựa trên những điểm chung này, chúng ta có một ý tưởng cho dù hình ảnh thứ hai lớn hơn hay nhỏ hơn hay nó đã được xoay và sau đó chồng lên nhau, hoặc có thể thu nhỏ xuống / sau đó được trang bị. Tất cả các thông tin như vậy được mang lại bằng cách thiết lập thư tín. Quá trình này được gọi là đăng ký.

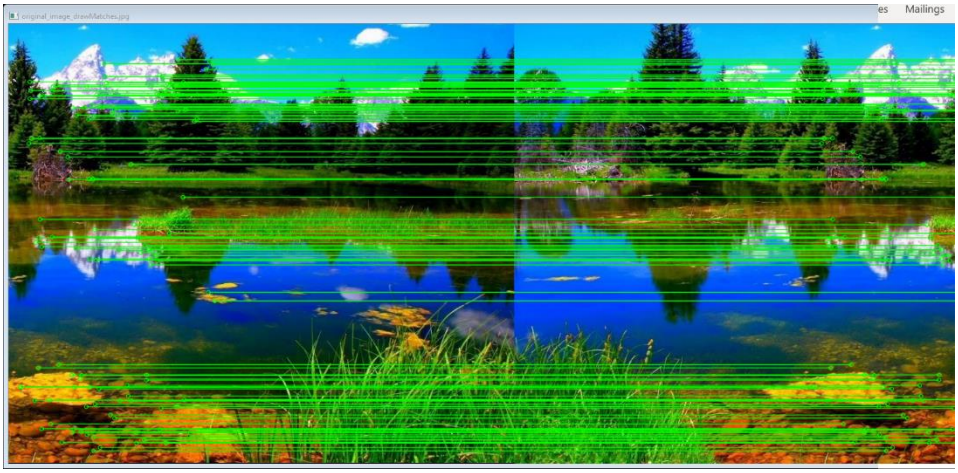
```
match = cv2.BFMatcher()
matches = match.knnMatch(des1,des2,k=2)
```

Thông thường trong hình ảnh có thể có nhiều khả năng các điểm chồng chéo tồn tại ở nhiều nơi của hình ảnh. Vì vậy, chúng tôi lọc qua tất cả các matches để có được những cái tốt nhất. Vì vậy, chúng tôi áp dụng kiểm tra tỷ lệ bằng cách sử dụng 2 matches đầu thu được ở trên. Xem xét một match nếu tỷ lệ được xác định dưới đây lớn hơn tỷ lệ được chỉ định.

```
good = []
for m,n in matches:
      if m.distance < 0.03*n.distance:
                  good.append(m)
```

Bây giờ xác định các tham số của các đường vẽ trên hình ảnh và đưa ra đầu ra để xem nó trông như thế nào khi chúng tôi tìm thấy tất cả các kết quả khớp trên hình ảnh:

```
draw_params = dict(matchColor = (0,255,0), #đường nối vẽ màu lục
singlePointColor = None,flags = 2)
img3 = cv2.drawMatches(img_,kp1,img,kp2,good,None,**draw_params)
cv2.imshow("original_image_drawMatches.jpg", img3)
```



Vì vậy, một khi đã thu được kết quả khớp tốt nhất giữa các hình ảnh, bước tiếp theo của chúng tôi là tính toán ma trận homography. Như chúng tôi đã mô tả trước đây, ma trận homography sẽ được sử dụng với các điểm phù hợp nhất, để ước tính một chuyển đổi định hướng tương đối trong hai hình ảnh.

Với OpenCV ta có:

```
H_= cv2.findHomography(srcPoints, dstPoints, cv2.RANSAC, 5)
```

Trước khi bắt đầu thuật toán khâu mã hóa, chúng ta cần trao đổi đầu vào hình ảnh. Vì vậy, ngay bây giờ, img_viết sẽ chụp ảnh phải và còn img sẽ chụp ảnh trái.

```
MIN_MATCH_COUNT = 10
if len(good) > MIN_MATCH_COUNT:
    src_pts = np.float32([ kp1[m.queryIdx].pt for m in     good
]).reshape(-1,1,2)
    dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good
]).reshape(-1,1,2)    M, mask = cv2.findHomography(src_pts, dst_pts,
cv2.RANSAC,5.0)    h,w = img1.shape
    pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-
1,1,2)
    dst = cv2.perspectiveTransform(pts,M)    img2 =
cv2.polylines(img2,[np.int32(dst)],True,255,3, cv2.LINE_AA)
    cv2.imshow("original_image_overlapping.jpg", img2)
else:
    print ("Not enough matches are found - %d/%d" %
(len(good),MIN_MATCH_COUNT))
```

Vì vậy, lúc đầu, chúng tôi đặt số điều kiện khớp tối thiểu của chúng tôi là 10 (được xác định bởi MIN_MATCH_COUNT) và chúng tôi chỉ thực hiện khâu nếu kết quả khớp tốt của chúng tôi vượt quá yêu cầu phù hợp. Mặt khác chỉ đơn giản là hiển thị một thông báo nói rằng không có đủ kết quả phù hợp.

Vì vậy, trong câu lệnh if, chúng ta đang chuyển đổi các Điểm chính (từ danh sách khớp) thành đối số cho hàm findHomography ().

```
cv2.imshow(“original_image_overlapping.jpg”, img2)
```

Vì vậy, một khi chúng ta đã thiết lập được một homography, chúng ta cần làm warp điểm nhìn, về cơ bản thay đổi phạm vi quan sát, chúng ta áp dụng ma trận đồng nhất sau cho hình ảnh:

```
warped_image = cv2.warpPerspective(image, homography_matrix,
                                   dimension_of_warped_image)
```

Vì vậy, chúng tôi sử dụng như sau:

```
dst = cv2.warpPerspective(img_,M,(img.shape[1] + img_.shape[1],
                                  img.shape[0]))
dst[0:img.shape[0], 0:img.shape[1]] = img
```

Trong hai dòng mã trên, chúng tôi đang lấy vùng chồng lấp từ hai hình ảnh đã cho. Sau đó, trong "dst", chúng tôi chỉ nhận được phần bên phải của hình ảnh không bị chồng chéo, vì vậy trong dòng mã thứ hai, chúng tôi sẽ đặt hình ảnh bên trái của mình vào hình ảnh cuối cùng. Vì vậy, tại thời điểm này, chúng tôi đã khâu đầy đủ hình ảnh:

Vì vậy, từ thời điểm này, phần còn lại là loại bỏ mặt tối của hình ảnh, vì vậy chúng tôi sẽ viết mã sau để xóa phông đen khỏi tất cả các viền hình ảnh:

```
def trim(frame):
#crop top
if not np.sum(frame[0]):
return trim(frame[1:])
#crop bottom
elif not np.sum(frame[-1]):
return trim(frame[:-2])
#crop left
elif not np.sum(frame[:,0]):
return trim(frame[:,1:])
#crop right
elif not np.sum(frame[:,-1]):
return trim(frame[:,:-2])
return frame
```

Và đây là chức năng được xác định cuối cùng mà chúng ta gọi để cắt viền và đồng thời chúng ta hiển thị ảnh đó trong màn hình của chúng ta.

```
cv2.imshow("original_image_stiched_crop.jpg", trim(dst))
```

Ngoài ra, nhóm cũng sử dụng thư viện OPENCV để thực hiện stitching image

Thư viện opencv đã cho ra kết quả tốt hơn nhiều so với cách cài đặt trên

TÀI LIỆU KHAM KHẢO

[1] https://en.wikipedia.org/wiki/Photographic_mosaic

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3597956/

[3] https://en.wikipedia.org/wiki/Super-resolution_imaging

[4] https://vi.wikipedia.org/wiki/Panorama

[5] http://www.researchpublications.org/IJCSA/NCAICN-13/230.pdf

[6] Herbert Bay, Andreas Es, Tinne Tuytelaars and Luc Van Gool, “Speeded-Up Robust Features (SURF) ”.

[7] Martin F., Robert B. – “Random sample consensus”- A paradigm for model fitting with application to image analysis and automated cartography, Communications of the ACM 24 (1981) 381–395.