HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

—----------

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: XỬ LÝ ẢNH

Đề tài 5: Ghép ảnh Panorama

Giảng viên: Phạm Hoàng Việt

Lớp: D22HTTT05 - XLA 12

Thành viên nhóm:

Nguyễn Duy Hoàng - B22DCCN334

Trần Trọng Đại - B22DCCN178

Hà Nội - 2025

Danh sách thành viên nhóm

Họ và tên	MSV	% Công việc
Nguyễn Duy Hoàng	B22DCCN334	50
Trần Trọng Đại	B22DCCN178	50

Mục lục

Chương I. Giới thiệu đề tài

I.1. Đặt vấn đề

Ghép ảnh luôn là một trong những thủ thuật được sử dụng rộng rãi trong những lĩnh vực về công nghệ (đồ họa, thị giác máy tính, xử lý ảnh,…) và có sự liên kết chặt chẽ với cuộc sống hằng ngày như chụp những bức ảnh toàn cảnh có tầm nhìn rộng bằng smartphone, thực hiện các video giám sát, hỗ trợ xe tự lái. Với các tính năng ghép nhiều hình ảnh chồng chéo để tạo nên một tấm ảnh lớn có góc nhìn toàn cảnh rộng rãi, thì ngoài các máy ảnh toàn phương 360 độ, các điện thoại thông minh cũng có thể xây dựng các bức tranh toàn cảnh từ nhiều các chuỗi hình ảnh. Có rất nhiều các ứng dụng nổi tiếng để có thể tạo nên các bức ảnh ghép như Adobe Photoshop, Image Composite Editor, …

Các thuật toán dùng để sắp xếp hình ảnh và ghép chúng lại thành một bức ảnh lớn liền mạch đã có từ lâu và được áp dụng phổ biến. Căn chỉnh tốc độ khung ảnh được sử dụng trong mọi máy quay có tính năng ổn định hình ảnh. Thuật toán ghép ảnh tạo ra các bức ảnh ghép có độ phân giải cao chính là tiền đề để có được những bức ảnh vệ tinh như bây giờ. Chính vì sự thú vị, sự tiện lợi, áp dụng tốt vào được thực tế, việc tạo nên một bức ảnh Panorama từ các ảnh nhỏ rất đáng để cho những người có niềm đam mê,yêu thích ảnh, xử lý ảnh và các công việc, học tập liên quan có những tìm hiểu, nghiên cứu sâu hơn nữa.

I.2. Mục đích, yêu cầu

I.2.1. Mục đích

Mục đích của đề tài là tìm hiểu được phương pháp ghép ảnh Panorama sử dụng thuật toán SIFT; Xây dựng được chương trình đơn giản thực hiện việc ghép ảnh Panorama; Đưa ra được những đánh giá, kết luận về thuật toán SIFT

và chương trình ghép ảnh Panorama.

I.2.2. Yêu cầu

- Nghiên cứu các khái niệm và đặc trưng về ảnh Panorama.
- Nghiên cứu phương pháp ghép ảnh Panorama sử dụng SIFT.
- Xây dựng chương trình Python đơn giản để ghép ảnh Panorama.

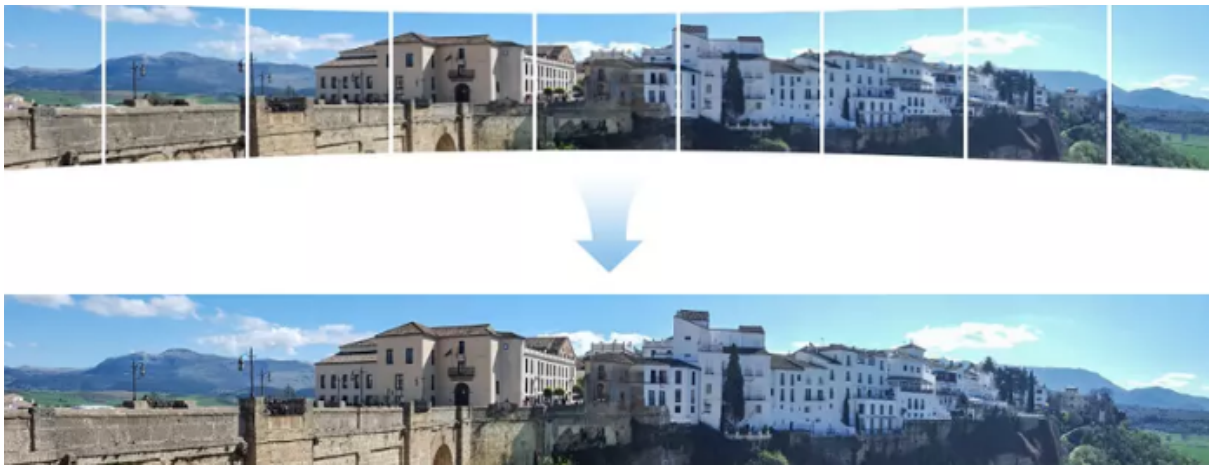
Chương II. Ảnh Panorama

II.1. Khái niệm

Ảnh Panorama (Ảnh toàn cảnh) là một bức ảnh có tầm nhìn rộng, bao quát được mọi vật trong cùng một khung hình. Nói đơn giản là xem hình ảnh với góc nhìn rộng hơn ảnh bình thường, tức là những khung ảnh cực kỳ lớn mà một khung hình chụp bằng máy ảnh thông thường không thể hiện hết.

Hình 1: Ví dụ về ảnh Panorama

Có thể hiểu rằng panorama là chế độ chụp ảnh khổ rộng bằng cách chụp nhiều tấm ảnh liên tiếp, với thông tin tấm ảnh trước được thể hiện một phần trong tấm ảnh sau. Sau đó với sự trợ giúp của phần mềm xử lý ảnh thì ta sẽ có được một tấm ảnh khổ rộng.



Hình 2: Xử lý ảnh Panorama

II.2. Phương pháp ghép ảnh

Để tạo nên một bức ảnh Panorama hoàn chỉnh thì cần những giai đoạn như: Thu nhận ảnh, Ghép ảnh, Trộn ảnh, Cắt ảnh,.... Tuy nhiên trong bài báo cáo này sẽ chỉ tập trung vào phương pháp ghép ảnh là chính.

Kỹ thuật ghép ảnh Panorama có rất nhiều cách khác nhau, và chúng đều có những đặc trưng cơ bản:

- Trích chọn điểm đặc trưng giữa hai ảnh.
- Tìm những điểm tương đồng giữa hai ảnh.
- Tìm ma trận Homography và ghép nối hai bức ảnh với nhau.

Thay vì phân tích toàn bộ hình ảnh, chỉ cần tập trung vào một vài điểm nhất định trên ảnh đó và thực hiện phép phân tích cục bộ gọi là "trích chọn đặc trưng dựa trên điểm nổi bật". Phương pháp này sẽ hoạt động tốt nếu như trên ảnh có tồn tại một số lượng vừa đủ các điểm nổi bật bất biến và ổn định có thể thực hiện việc phân tích cục bộ một cách chính xác.

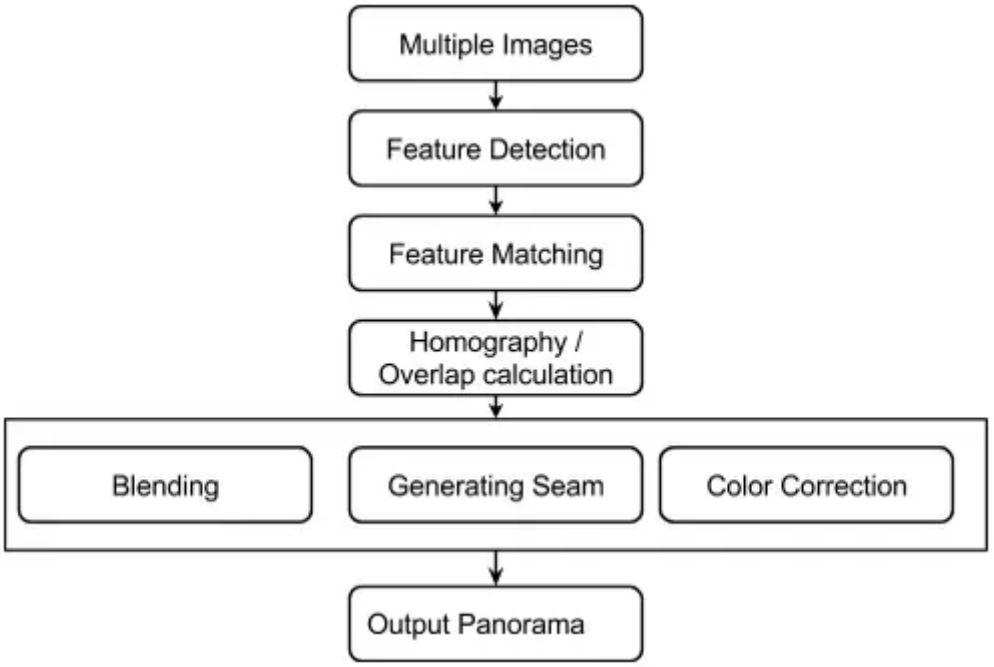
Điểm nổi bật trong ảnh là điểm ảnh có chứa nhiều thông tin hơn các điểm ảnh lân cận. Biểu diễn ảnh theo điểm nổi bật sẽ cô đọng hơn, giảm được không gian tìm kiếm trong các bài toán ứng dụng.

Các phương pháp trích chọn đặc trưng từ các điểm nổi bật như là: Thuật toán tìm kiếm góc Harris; Trích chọn đặc trưng cục bộ bất biến SIFT (Scale Invariant Feature Transform); Trích chọn đặc trưng SURF (Speed Up Robust Features),... và việc lựa chọn thuật toán, phương pháp nào tùy thuộc vào từng bài toán cụ thể. Trong bài báo cáo này, ta sẽ tập trung vào thuật toán SIFT.

Chương III. Phương pháp ghép ảnh

III.1. Luồng xử lý

Dưới đây là luồng xử lý cho bài toán ghép ảnh (trường hợp chỉ có 2 ảnh):



Hình 3: Luồng xử lý

Mô tả tổng quan của luồng này như sau:

- Sử dụng các thuật toán chuyên biệt để detect tập các keypoint cho từng ảnh. Những keypoint này là những điểm đặc biệt, mang tính chất đặc trưng và không bị ảnh hưởng (hoặc ảnh hưởng ít) bởi độ sáng, các phép xoay, (zoom)...

- Tìm cách so khớp (matching) 2 tập keypoint này, tìm ra các cặp keypoint tương ứng.

- Dựa vào các cặp keypoint đó để tìm ra cách biến đổi -> ghép 2 ảnh lại với nhau. Như vậy ra đã thu được ảnh Panorama.

III.2. Thuật toán SIFT

Thuật toán SIFT có thể phát hiện và mô tả điểm đặc trưng không ảnh hưởng đến tỷ lệ và góc quay của hình ảnh.

III.2.1. Không gian tỷ lệ

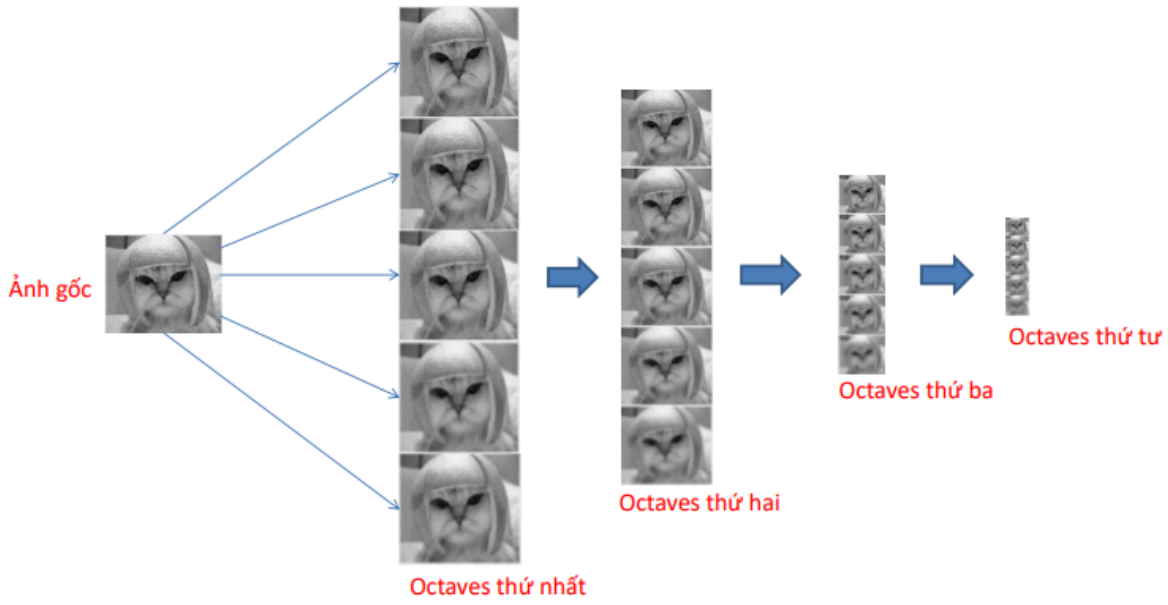
Không gian tỷ lệ của một ảnh là một hàm L(x,y,σ) được tạo ra từ tích chập của bộ lọc Gaussian để làm mờ ở các tỷ lệ khác nhau với ảnh đầu vào.

Không gian tỷ lệ bao gồm các Octaves, mỗi Octaves là ảnh làm mờ dần từ ảnh đầu vào. Số lượng Octaves và tỷ lệ phụ thuộc kích thước của ảnh gốc:

- Mỗi Octaves bước sau có kích thước bằng một nửa ảnh trước đó.
- Chỉ cần Octaves thứ tư hoặc thứ năm là phù hợp.

Làm mờ (làm mịn): L(x,y,σ) = G(x,y,σ) * I(x,y)

- L(x,y,σ) là một ảnh mờ
- G (x,y,σ) là toán tử Gaussian: $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$
- I(x,y) là ảnh đầu vào
- σ là tham số "tỷ lệ" -> lượng mờ -> giá trị càng lớn, độ mờ càng lớn.



Hình 4: Ví dụ về Không gian tỷ lệ và các Octaves

III.2.2. Mô tả thuật toán SIFT

Chi tiết của các bước trong thuật toán SIFT như sau:

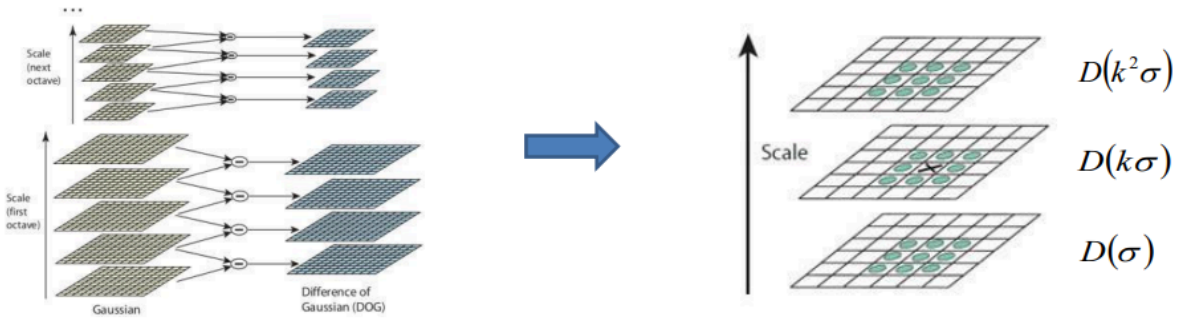
+ Bước 1: Tìm kiếm điểm keypoint tiềm năng

- Gọi σ là hệ số "tỷ lệ" -> lượng mờ -> Lượng mờ trong ảnh tiếp theo sẽ là kσ (k là hằng số bất kỳ)

-> L(x,y,kσ) = G(x,y,kσ) * I(x,y)

-> Tìm các điểm cực trị của hàm DoG: D(x,y,σ) = (G(x,y,kσ) - G(x,y,σ))*I(x,y)

- Tìm điểm cực trị: so sánh các điểm trên ảnh DoG so sánh với 8 điểm lân cận và 9 điểm ảnh DoG trước và 9 điểm ảnh DoG sau -> Điểm cực trị là điểm keypoint tiềm năng.



Hình 5: Tìm kiếm điểm keypoint tiềm năng

+ Bước 2: Định vị điểm keypoint

Mỗi điểm keypoint tiềm năng sẽ được đánh giá xem có giữ lại không:

- Loại bỏ các điểm keypoint có độ tương phản thấp.

- Loại bỏ các điểm keypoint dọc theo các biên (cạnh) do dễ bị nhiễu.

+ Bước 3: Xác định hướng cho điểm keypoint: Tính toán biểu đồ hướng Gradient trong vùng láng giềng của điểm keypoint.

Độ lớn và hướng của các điểm keypoint xác định theo công thức:

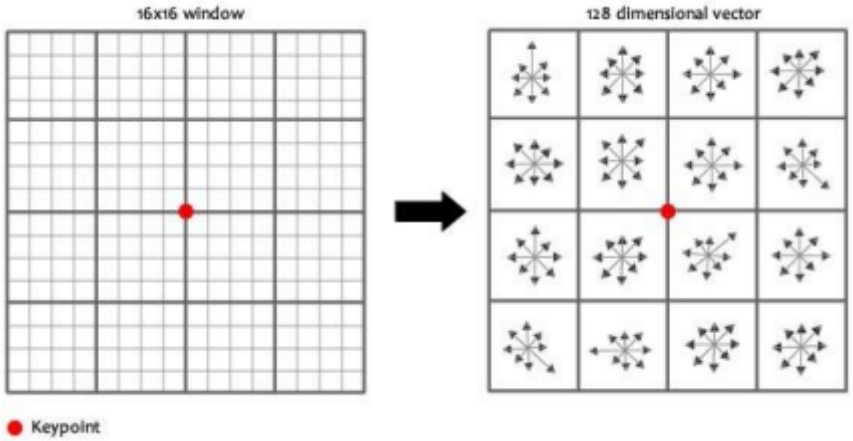
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

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

+ Bước 4: Tạo bộ mô tả điểm keypoint:

Mô tả các điểm keypoint dưới dạng vectơ đặc trưng nhiều chiều.

- Một cửa sổ 16x16 xung quanh keypoint và chia thành 16 khối phụ có kích thước 4x4.
- Mỗi khối phụ có 8 hướng -> 4x4x6=128 chiều -> Mỗi điểm keypoint được biểu diễn bằng 1 vector đặc trưng 128 chiều.



Hình 6: Biểu diễn các keypoint

Ví dụ về kết quả trên ảnh sau khi sử dụng thuật toán:

Hình 7: Ví dụ về thuật toán SIFT

III.3. Thuật toán RANSAC

Sau khi tìm được các tập keypoint của 2 ảnh, để có thể tìm ra các bộ mô tả khớp nhau, ta biểu diễn các bộ này dưới dạng 1 ma trận đồng nhất H 3x3 (Hormography), sử dụng thuật toán RANSAC (Random Sample Consensus) được biểu diễn như sau:

- Từ các cặp điểm so khớp giữa 2 ảnh: (k1, k'1), (k2, k'2)…(kn, k'm)
- Chọn 4 cặp điểm và tính ma trận H bằng cách tính hàm mất (loss):

$$\text{Loss} = \sum_{i=0}^{n} (\text{disstance}(H * ki, k'_i))$$

- Lặp lại quá trình với số lần lặp đủ lớn. Sau đó chọn H có Loss bé nhất.

Ví dụ về kết quả trên ảnh sau khi sử dụng thuật toán:

Hình 8: Ví dụ về thuật toán RANSAC

Sau khi thực hiện thuật toán trên, ta có thể thực hiện biến đổi phối cảnh cho ảnh nguồn để được ảnh nguồn đã được biến đổi tọa độ sang tọa độ ảnh đích.

Chương IV. Cài đặt, kết quả thực nghiệm

IV.1. Cài đặt chương trình

Ta sẽ viết chương trình sử dụng Python để hiển thị ảnh qua từng giai đoạn. Các bước thực hiện ghép ảnh (đối với 2 ảnh nguồn và đích) được mô tả như sau:

+ Bước 1: Phát hiện các điểm keypoint và trích xuất các bộ mô tả keypoint của hai ảnh:

- Tập điểm keypoint, tập bộ mô tả điểm keypoint của ảnh nguồn:

S1 = {k1, k2, ... kn}; DS1 = {dk1, dk2, ... dkn}

- Tập điểm keypoint, tập bộ mô tả điểm keypoint của ảnh đích:

S2 = {k'1, k'2, ... k'm}; DS2 = {dk'1, dk'2, ... dk'm}

+ Bước 2: So khớp các bộ mô tả giữa hai ảnh DS1 và DS2.

+ Bước 3: Sử dụng thuật toán RANSAC tìm ma trận đồng nhất H.

+ Bước 4: Dựa trên ma trận H thực hiện biến đổi phối cảnh cho ảnh nguồn để được ảnh nguồn đã được biến đổi tọa độ sang tọa độ ảnh đích.

+ Bước 5: Ghép ảnh nguồn đã được biến đổi tọa độ với ảnh đích để có được ảnh Panorama.

IV.2. Xây dựng thuật toán, chương trình

Ta xây dựng các hàm thuật toán trong file **panorama.py** ở class tên là Stitcher như sau:

IV.2.1. Phát hiện các keypoint

Ta sử dụng hàm detectAndDescribe() để phát hiện các keypoint trong ảnh:

```python
def detectAndDescribe(self, image):
    # Chuyển đổi hình ảnh sang thang độ xám
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Kiểm tra xem ta có đang sử dụng OpenCV v3.X hay không
    if self.isv3:
        # Phát hiện và trích xuất các đặc điểm từ hình ảnh
        descriptor = cv2.xfeatures2d.SIFT_create()
        (kps, features) = descriptor.detectAndCompute(image, None)
    # Nếu không, ta đang sử dụng OpenCV 2.4.X
    else:
        # Phát hiện các keypoint trong hình ảnh
        detector = cv2.FeatureDetector_create("SIFT")
        kps = detector.detect(gray)
        # Trích xuất các feature từ hình ảnh
        extractor = cv2.DescriptorExtractor_create("SIFT")
        (kps, features) = extractor.compute(gray, kps)
    # Chuyển đổi các keypoint từ đối tượng KeyPoint sang mảng NumPy
    kps = np.float32([kp.pt for kp in kps])
    # Trả về một bộ các keypoint và các feature
    return (kps, features)
```

Trong hàm trên, ta phát hiện các keypoint sử dụng Difference of Gaussian (DoG) keypoint detector và SIFT feature extractor. Kết quả cuối cùng được trả về là các đối tượng KeyPoint đã được chuyển đổi sang mảng NumPy để thực hiện tính toán.

IV.2.2. Liên kết các keypoint

Sau khi đã tìm ra được các keypoint của mỗi ảnh, ta sử dụng hàm matchKeypoints() để tìm ra điểm tương đồng giữa các keypoint trong 2 ảnh:

```python
def matchKeypoints(self, kpsA, kpsB, featuresA, featuresB, ratio, reprojThresh):
    # Tính toán các kết quả khớp thô (raw matches) và khởi tạo danh sách các kết quả khớp thực tế
    matcher = cv2.DescriptorMatcher_create("BruteForce")
    rawMatches = matcher.knnMatch(featuresA, featuresB, 2)
    matches = []
    # Lặp qua các kết quả khớp thô
    for m in rawMatches:
        # Đảm bảo khoảng cách nằm trong một tỷ lệ nhất định của mỗi kết quả khác (Ví dụ: Kiểm tra tỷ lệ Lowe)
        if len(m) == 2 and m[0].distance < m[1].distance * ratio:
            matches.append((m[0].trainIdx, m[0].queryIdx))
    # Tính toán phép đồng dạng (homography) cần ít nhất 4 phép khớp
    if len(matches) > 4:
        # Xây dựng hai tập hợp điểm
        ptsA = np.float32([kpsA[i] for (_, i) in matches])
        ptsB = np.float32([kpsB[i] for (i, _) in matches])
        # Tính toán phép đồng dạng (homography) giữa hai tập hợp điểm
        (H, status) = cv2.findHomography(ptsA, ptsB, cv2.RANSAC, reprojThresh)
        # Trả về các kết quả khớp cùng với ma trận đồng dạng và trạng thái của mỗi điểm khớp
        return (matches, H, status)
    # Nếu không, không thể tính toán được sự đồng dạng
    return None
```

Để ghép các đặc điểm lại với nhau, ta chỉ cần lặp qua các mô tả từ cả hai ảnh, tính toán khoảng cách và tìm khoảng cách nhỏ nhất cho mỗi cặp. OpenCV có một hàm tích hợp gọi là cv2.DescriptorMatcher_create() xây dựng bộ so khớp feature. BruteForce cho biết rằng ta sẽ tính toán một cách đầy đủ khoảng cách Euclide giữa tất cả các vectơ đặc trưng từ cả hai hình ảnh và tìm ra cặp mô tả có khoảng cách nhỏ nhất. Hàm knnMatch() thực hiện khớp k-NN giữa hai tập vectơ đặc trưng bằng cách sử dụng k=2 (chỉ ra hai kết quả khớp hàng đầu cho mỗi vectơ đặc trưng được trả về), sau đó thực hiện kiểm tra tỷ lệ của David Lowe để loại bỏ kết quả khớp dương tính giả.

Việc tính toán phép đồng dạng (homography) sử dụng RANSAC giữa hai tập hợp điểm yêu cầu tối thiểu bốn phép trùng khớp ban đầu. Để ước tính phép đồng dạng đáng tin cậy hơn thì ta cần nhiều hơn bốn điểm trùng khớp.

Tiếp theo, ta sử dụng hàm drawMatches() để trực quan hóa những keypoint đã được liên kết giữa 2 bức ảnh:

```
def drawMatches(self, imageA, imageB, kpsA, kpsB, matches, status):
    # Khởi tạo hình ảnh trực quan đầu ra
    (hA, wA) = imageA.shape[:2]
    (hB, wB) = imageB.shape[:2]
    vis = np.zeros((max(hA, hB), wA + wB, 3), dtype="uint8")
    vis[0:hA, 0:wA] = imageA
    vis[0:hB, wA:] = imageB
    # Lặp lại các matches
    for ((trainIdx, queryIdx), s) in zip(matches, status):
        # Chỉ xử lý kết quả khớp nếu keypoint đã được xác định khớp thành công
        if s == 1:
            ptA = (int(kpsA[queryIdx][0]), int(kpsA[queryIdx][1]))
            ptB = (int(kpsB[trainIdx][0]) + wA, int(kpsB[trainIdx][1]))
            cv2.line(vis, ptA, ptB, (0, 255, 0), 1)
    # Trả về hình ảnh trực quan
    return vis
```

Từ 2 hình ảnh gốc, ta tập hợp các điểm chính liên quan đến từng hình ảnh, các kết quả khớp ban đầu sau khi áp dụng thử nghiệm tỷ lệ Lowe và cuối cùng là trạng thái danh sách được cung cấp bởi phép tính đồng dạng. Sử dụng các biến này, ta có thể hình dung các điểm chính "nội tại" bằng cách vẽ một

đường thẳng từ điểm chính N trong ảnh đầu tiên đến điểm chính M trong ảnh thứ hai.

IV.2.3. Biểu diễn và ghép ảnh

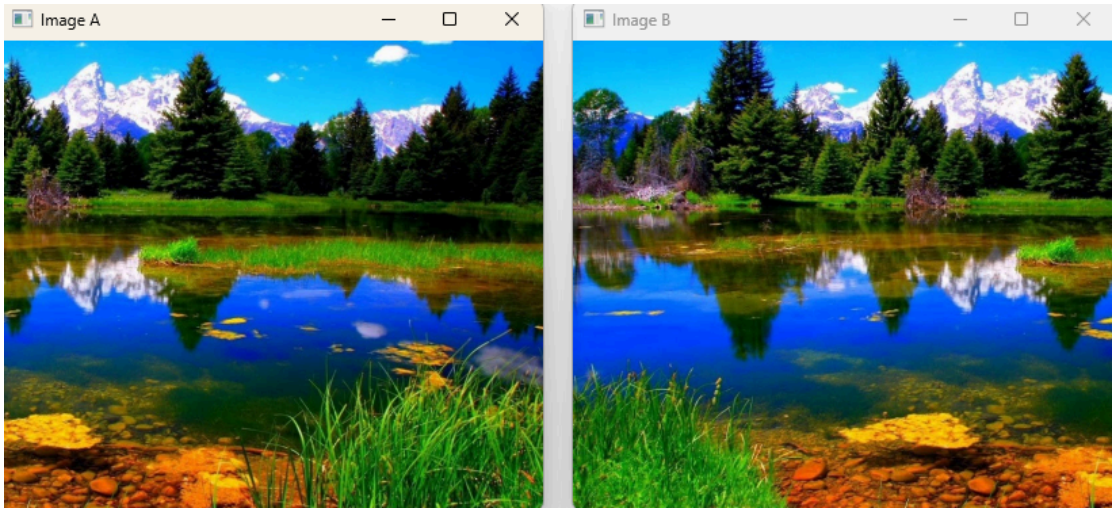
Sau khi hoàn thiện 3 hàm trên, ta tổng hợp lại bằng hàm stitch():

```
def stitch(self, images, ratio=0.75, reprojThresh=4.0, showMatches=False):
    # Giải nén hình ảnh, sau đó phát hiện các điểm chính (keypoint) và trích xuất các mô tả bất biến cục bộ
    (imageA, imageB) = images
    (kpsA, featuresA) = self.detectAndDescribe(imageA)
    (kpsB, featuresB) = self.detectAndDescribe(imageB)
    # Khớp các tính năng (feature) giữa hai hình ảnh
    M = self.matchKeypoints(kpsA, kpsB, featuresA, featuresB, ratio, reprojThresh)
    # Nếu kết quả khớp là None thì không có đủ kết quả khớp keypoint để tạo ra một bức tranh toàn cảnh
    if M is None:
        return None
    # Nếu không, hãy áp dụng hiệu ứng cong vênh để ghép các hình ảnh cùng nhau
    (matches, H, status) = M
    result = cv2.warpPerspective(imageA, H, (imageA.shape[1] + imageB.shape[1], imageA.shape[0]))
    result[0:imageB.shape[0], 0:imageB.shape[1]] = imageB
    # Kiểm tra xem các keypoint có khớp nhau hay không để trực quan hóa
    if showMatches:
        vis = self.drawMatches(imageA, imageB, kpsA, kpsB, matches, status)
        # Trả về một bộ hình ảnh được khâu và hình dung
        return (result, vis)
    # Trả về một bộ hình ảnh đã khâu
    return result
```

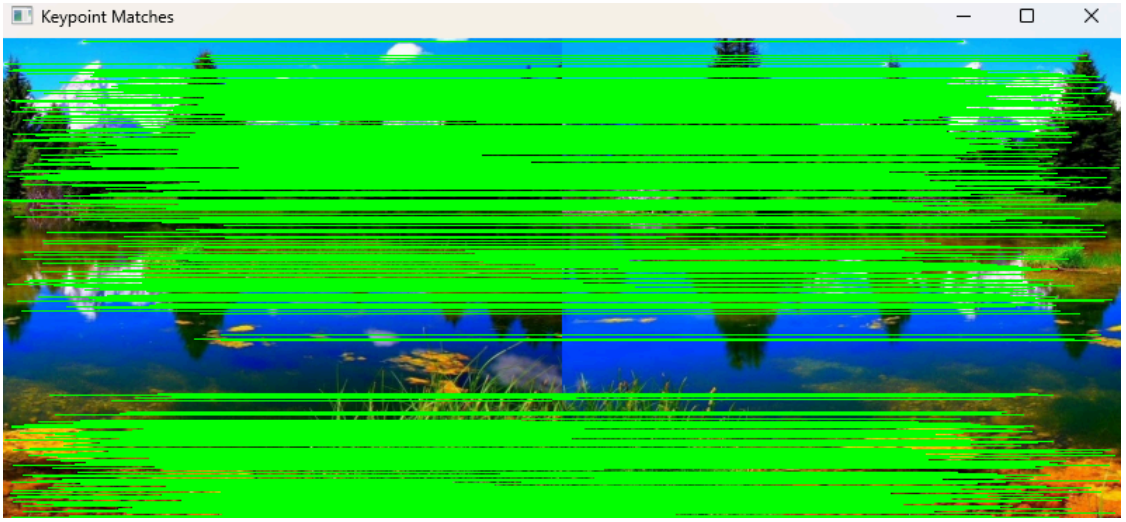
Đây là hàm sẽ thực hiện chương trình chính để hiển thị ảnh Panorama.

IV.3. Kết quả thực nghiệm

+ Kết quả thực nghiệm:



Hình 9: 2 ảnh đầu vào (Input)



Hình 10: Match Keypoint

Hình 11: Kết quả ghép ảnh (Output)

Kết luận

Bài báo cáo đã trình bày các khái niệm và phương pháp, thuật toán về việc tạo ảnh Panorama. Đồng thời cũng tạo một chương trình đơn giản để kiểm thử thuật toán và đưa ra được kết quả là bức ảnh Panorama như mong muốn. Mặc dù có rất nhiều các phương pháp, thuật toán khác nhau về việc tạo ảnh Panorama nhưng báo cáo mới chỉ dừng lại ở một phương pháp và thuật toán. Chương trình tạo ra cũng khá là đơn giản, chưa có tính ứng dụng cao, các tính năng cần được cải thiện nếu như có thêm thời gian, có thể phát triển sâu rộng hơn trong việc khắc phục ảnh Panorama khi thực hiện ghép các ảnh con bị mờ, nhiễu, các góc cạnh chưa có sự tương xứng. Khi khắc phục được những hạn chế cơ bản trên và phát triển chương trình hoàn thiện hơn, chúng em có thể sử dụng đề tài này để thực hiện đồ án tốt nghiệp.

Lời cuối cùng, nhóm em xin chân thành cảm ơn thầy đã dành thời gian!

Tài liệu, đường liên kết

- OpenCV, docs.opencv.org (2020).

  OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform)

- Trung Thành Nguyễn, Viblo (16/10/2020).

  Image Stitching - thuật toán đằng sau công nghệ ảnh Panorama.