|  |
| --- |
| Trường Đại học Phenikaa |
| Báo Cáo Bài Tập Lớn Học Phần  “THỊ GIÁC MÁY TÍNH” |
| **Đề tài: Panorama Stitching** |
|  |
| **Thành Viên:**   * 19010064 - Đậu Bá Nhật Minh (Nhóm Trưởng) * 19010062 - Phạm Bá Khương * 19010056 - Đường Ngọc Hà * 19010023 - Lê Công Nhân * 19010070 - Nguyễn Quốc Thịnh   **Giảng viên hướng dẫn: Nguyễn Văn Tới** |
|  |

|  |
| --- |
|  |

**Mục Lục**

Table of Contents

**Type chapter title (level 1)1**

Type chapter title (level 2)2

Type chapter title (level 3)3

**Type chapter title (level 1)4**

Type chapter title (level 2)5

Type chapter title (level 3)6

**Chương 1: Giới thiệu đề tài Bài Tập Lớn**

* 1. **Giới thiệu đề tài**

Panorama Stitching, hay còn gọi là image stitching là quá trình kết hợp hai hoặc nhiều bức ảnh có các trường chồng lên nhau để tạo ra một bức ảnh toàn cảnh hoặc ảnh có độ phân giải cao hơn.

* 1. **Nội dung nghiên cứu và mục tiêu**
     1. Nội dung nghiên cứu
* Ngôn ngữ lập trình Python
* Thư viện OpenCV
* Feature Descriptor SIFT

1.2.2 Mục tiêu nghiên cứu

Với đầu vào là 2 bức ảnh có vùng overlap lên nhau và xuất ra ảnh đã ghép 2 bức ảnh trên thành một.

* 1. **Thành viên trong nhóm và đóng góp**
     1. Phân công công việc

|  |  |
| --- | --- |
| **Họ và tên** | **Nhiệm vụ được phân công** |
| Đậu Bá Nhật Minh | Tìm hiểu về thuật toán SIFT để tìm và phát hiện Keypoint |
| Phạm Bá Khương | Tìm hiểu về thuật toán FLANN để nối các KeyPoint |
| Đường Ngọc Hà | Tìm hiểu về phép biến đổi hình ảnh Perspective Transform |
| Lê Công Nhân | Tìm hiểu về thuật toán RANSAC để tính ma trận Homography |
| Nguyễn Quốc Thịnh | Chỉnh màu sau khi đã ghép xong ảnh |

* + 1. Đánh giả của trưởng nhóm
* Đậu Bá Nhật Minh:
  + Đã tìm hiểu và hiểu rõ một cách chi tiết SIFT và đã trình bày thành công trước lớp.
  + Nhiệt tình đóng góp ý kiến trong quá trình làm dự án
  + Có tìm hiểu đầy đủ tất cả các phần của các thành viên và giúp đỡ các bạn hiểu rõ các kiến thức sử dụng trong bài tập lớn.
* Phạm Bá Khương
  + Đã tìm hiểu sơ lược tổng quát về thuật toán FLANN
  + Tham gia đầy đủ các buổi họp nhóm tuy nhiên vẫn chưa nhiệt tình đóng góp ý kiến
* Đường Ngọc Hà
  + Đã tìm hiểu đầy đủ về phép biến đổi hình ảnh Perspective Transform
  + Tham gia đầy đủ các buổi họp nhóm và có đóng góp ý kiến trong các buổi họp
* Lê Công Nhân
  + Đã tìm hiểu chi tiết về thuật toán RANSAC
  + Tham gia đầy đủ các buổi họp nhóm tuy nhiên vẫn chưa nhiệt tình đóng góp ý kiến
* Nguyễn Quốc Thịnh
  + Đã tìm hiểu đầy đủ về bước hậu xử lý chỉnh màu sau khi ghép ảnh
  + Tham gia đầy đủ các buổi họp nhóm và có khả năng tiếp thu kiến thức tốt
  1. **Phương thức thực hiện và kết quả**
     1. Phương thức thực hiện
* Nắm chắc kiến thức cơ bản về ngôn ngữ lập trình Python
* Áp dụng thư viện OpenCV vào xử lý ảnh
* Tìm hiểu và áp dụng thuật toán SIFT và các thuật toán liên quan để thực hiện ghép ảnh
  + 1. Kết quả

Đã tìm ra các keypoint trong một bức ảnh và thành công trong công việc matching, thực hiện các phép biến đổi hình ảnh và ghép các bức ảnh lại với nhau.

* 1. **Tài liệu**

<https://drive.google.com/drive/folders/1SaZN6oPjMmqMo2gO1aIXwIQEy6eX-p6i?usp=sharing>

**Chương 2: Nội dung tổng quan**

* 1. **Ngôn ngữ lập trình Python**
* Ngôn ngữ lập trình Python được Guido van Rossum tạo ra cuối năm 1990.
* Python là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) bậc cao cho các mục đích lập trình đa năng
* Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu
* Ban đầu, Python được phát triển để chạy trên nền [Unix](https://vi.wikipedia.org/wiki/Unix). Nhưng rồi theo thời gian, Python dần mở rộng sang mọi [hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) từ [MS-DOS](https://vi.wikipedia.org/wiki/MS-DOS) đến [Mac OS](https://vi.wikipedia.org/wiki/Mac_OS), OS/2, [Windows](https://vi.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://vi.wikipedia.org/wiki/Linux) và [các hệ điều hành khác thuộc họ Unix](https://vi.wikipedia.org/wiki/T%C6%B0%C6%A1ng_t%E1%BB%B1_Unix).
* Cú pháp Python giúp dễ dàng diễn đạt các khái niệm toán học, vì vậy ngay cả những người không quen thuộc với ngôn ngữ này cũng có thể bắt đầu xây dựng các mô hình toán học một cách dễ dàng
* Python được thiết kế để dễ sử dụng, học nhanh và có cú pháp dễ tiếp cận
* Python thích hợp để viết mã và triển khai hợp tác, vì mã của nó dễ đọc và dễ truyền tải cho người khác
* Python có một cộng đồng lớn hỗ trợ ngôn ngữ này
  1. **Thư viện OpenCV**

OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV được viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực.

Opencv có rất nhiều ứng dụng:

* Nhận dạng ảnh
* Xử lý hình ảnh
* Phục hồi hình ảnh/video
* Thực tế ảo
* Các ứng dụng khác
  1. **Những thư viện khác được sử dụng trong bài tập lớn**
* Thư viện Numpy
* Thư viện Matplotlib
  1. **Giới thiệu về thuật toán SIFT**

SIFT – Scale Invariant Feature Transform là thuật toán được dùng để tìm kiếm các Keypoint trong một bức ảnh. Những Keypoint này là những điểm có tính chất đặc biệt, giàu tính năng và đặc trưng. Với từng keypoint, SIFT trả về tọa độ (x,y) kèm Descriptor – Vector 128 chiều đại diện cho các đặc trưng trong Keypoint đó.

Các Descriptor này hầu như không hoặc ít bị ảnh hưởng bởi độ xoay, ánh sáng, scale… Điều đó có nghĩa là cùng 1 điểm xuất hiện trên 2 ảnh thì sẽ có descriptor xấp xỉ nhau cho dù góc chụp hay độ sáng có thể khác nhau.



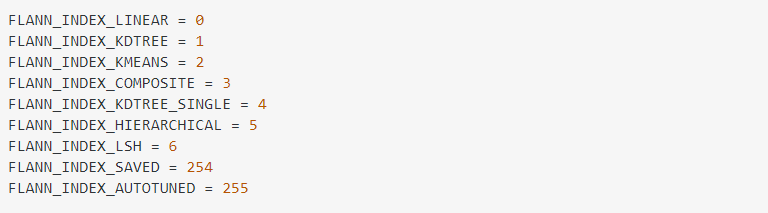
*Mỗi vòng tròn là 1 keypoint với một đường thẳng biểu thị hướng Gradient của nó*

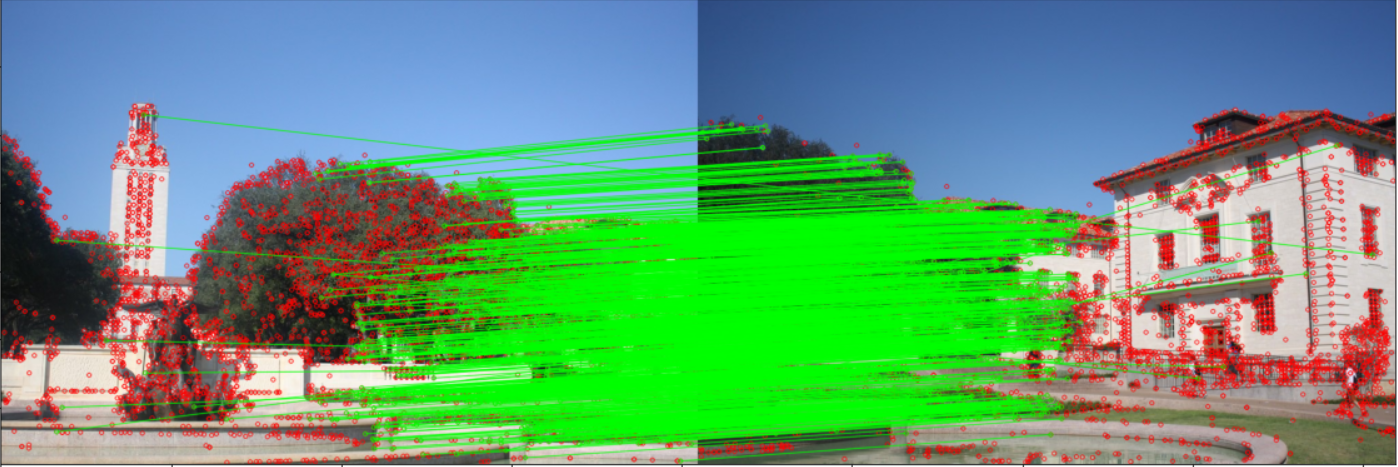
* 1. **Giới thiệu về thuật toán FLANN**

Sau khi đã có 2 tập keypoint *S1* và *S2* trên 2 bức ảnh, chúng ta cần tính khoảng cách Euclid giữ một Keypoint *ki*trong tập *S1* với tất cả cá Keypoint trong tập *S2*, đo độ sai khác với nhau để từ đó tìm ra các cặp điểm match nhau nhất giữa 2 tập. Tuy nhiên chúng ta có thể dùng FLANN để đảm bảo tốc độ và hiệu năng.

FLANN (Fast Library for Approximate Nearest Neighbors) là một thư viện chứa một tập hợp các thuật toán được đánh giá là tốt nhất trong việc tìm kiếm vùng lân cận gần nhất và nó còn là hệ thống giúp chọn thuật toán tốt nhất trong việc matching keypoint dựa vào tập dự liệu.

Các thuật toán có trong FLANN trong Python:

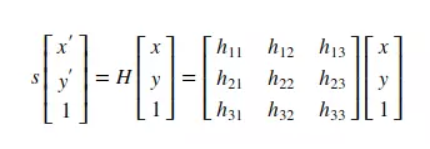




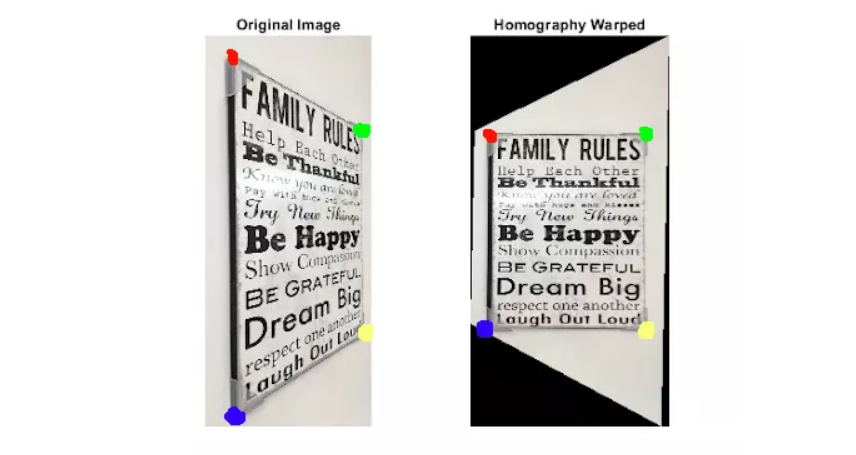
*Các keypoint được nối với nhau 1 cách khá chính xác và đầy đủ*

* 1. **Giới thiệu về phép biến đổi Perspective Transform và thuật toán RANSAC**

Trong xử lý ảnh, các phép biến đổi ảnh sẽ được thực hiện bằng cách nhân ma trận trên hệ tọa độ đồng nhất. Giả sử 1 điểm ảnh có tọa độ (x,y) sẽ được viết thành (x,y,1). Tọa độ (x,y,1) cũng tương đương với (xz, yz, z). Phép biến đổi 1 ảnh tương đương với việc nhân tọa độ các điểm với 1 ma trận H kích thước 3x3.

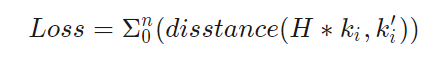


Trong các phép biến đổi ảnh, Perspective transform là phép biến đổi không bảo toàn góc, độ dài, tính song song... mà chỉ bảo toàn đường thẳng. Cũng nhờ tính chất này, perspective transform giúp chúng ta biến đổi ảnh từ 1 góc chiếu này sang 1 góc chiếu khác hẳn.



Như vậy, nếu có được 4 điểm trong ảnh input và 4 điểm tương ứng trong ảnh target, ta có thể tính được giá trị từng phần tử trong matrix H. Tuy nhiên, phương pháp keypoint matching trên ta sẽ thu được hàng chục, thậm chí hàng trăm cặp điểm tương ứng [(*k*1​,*k*1′​),(*k*2​,*k*2′​)...(*kn*​,*kn*′​)]. Vậy nên chọn 4 cặp nào trong hằng trăm cặp điểm kia để tính H. Khi đó ta sử dụng thuật toán RANSAC.

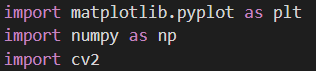
RANSAC - Random Sample Consesus, là 1 thuật toán khá đơn giản. Trong bài toán này, RANSAC chỉ đơn giản là lấy mẫu bất kì 4 cặp điểm ngẫu nhiên, tính matrix H. Tính tổng độ sai khác giữa các điểm target và các điểm input sau khi biến đổi bằng H. Ta có công thức:



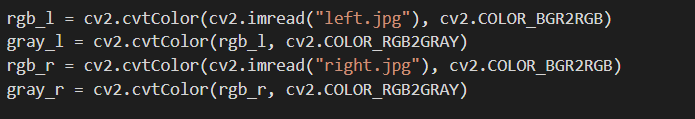
Trong đó, *ki*​ và *ki*′​ là 1 cặp điểm tương ứng. Lặp lại quá trình lấy mẫu - tính loss này với sô lần đủ lớn. Sau đó chọn H có Loss bé nhất. Như vậy ta đã thu được Homography matrix H dùng để biến đổi tọa độ các điểm trong ảnh input sang ảnh target.

P.S: *Tài liệu chi tiết về các thuật toán trên đã được chúng em tổng hợp tại mục* ***1.5. Tài Liệu***

**Chương 3: Triển khai bài toán**

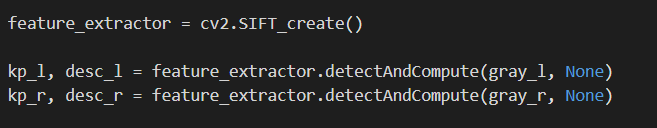


Đầu tiên chúng ta cần khởi tạo những thư viện cần thiết, đặc biệt là OpenCV – cv2.



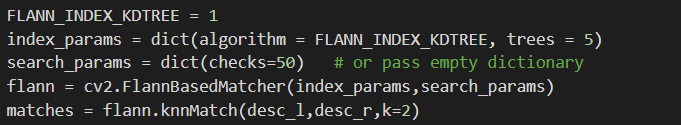
Tiếp theo chúng ta sẽ thực hiện cụ thể bài toán trên 2 bức ảnh (Do ảnh ban đầu được định dạng BGR nên chúng ta sẽ dùng hàm cv2.cvtcolor() để chuyển bức ảnh sang định dạng RGB để dễ nhìn hơn).

Chúng ta cũng phải cần chuyển bức ảnh về hệ màu Gray vì SIFT feature yêu cầu đầu vào là ảnh với hệ màu Gray.



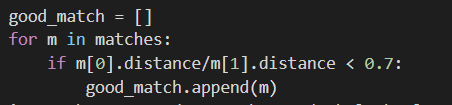
Thực hiện khởi tạo SIFT bằng cv2.SIFT\_create().

Trích xuất Keypoint và Descriptor từ SIFT bằng hàm detectAndCompute() với đầu vào là ảnh với hệ màu Gray.

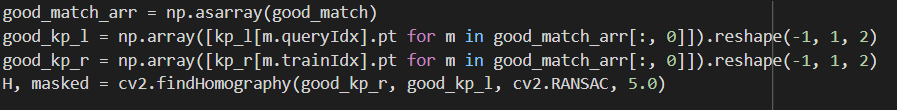


Bắt đầu triển khai thuật toán FLANN, ban đầu chúng ta khởi tạo 2 tham số cho thuật toán bằng hàm dict() - hàm tạo dictionary. Tham số index\_params sẽ bao gồm thuật toán ở đâu chúng ta lựa chọn là FLANN\_INDEX\_KDTREE với số lượng cây = 5. Tham số search\_params sẽ chỉ định số lần các cây trong index\_params được duyệt đệ quy, giá trị càng cao thì độ chính xác cũng sẽ càng cao, tuy nhiên cũng sẽ tốn nhiều thời gian hơn.

Tiếp theo chúng ta khởi tạo thuật toán FLANN bằng hàm cv2.FlannBasedMatcher(index\_params,search\_params). Cuối cùng, tìm các cặp keypoint matching với nhau bằng hàm knnMatch() với đầu vào là các Descriptor.

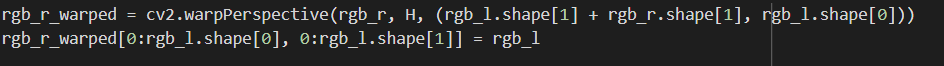


Do sẽ có hàng chục, thậm chí hàng trăm các cặp keypoint cho nên chúng ta sẽ dùng một ratio để lọc những cặp tốt nhất (ở đây chúng ta sẽ sử dụng giá trị 0.7 theo tài liệu của David G.Lowe).



Do các good\_match được định dạng tuple nên chúng ta cần đứa chúng về thành array bằng hàm numpy.asarray()

Tiếp theo chúng ta sẽ lấy những cặp matching chia chúng lại trở thành những điểm keypoint ở 2 bức ảnh. Sau đó chúng ta sẽ tính ma trận Homography bằng hàm cv2.findHomography() với đầu vào các điểm keypoint tốt đã matching ở 2 tập và thuật toán cv2.RANSAC.



Cuối cùng, chúng ta sẽ thực hiện công việc ghép các bức ảnh với hàm cv2.warpPerspective() bằng ma trận Homography H mà chúng ta đã tính được từ thuật toán RANSAC, thực hiện reshape hình ảnh và chúng ta sẽ có được kết quả cuối cùng là bức ảnh Panorama.

**Chương 4: Kết Luận**

Đầu vào:





Kết quả:



Do bài toán chỉnh sửa màu sau khi thực hiện ghép ảnh là một bài toán khá khó khăn và chúng em đã cố gắng tìm hiểu thử một số thuật toán nhưng kết quả trả lại vẫn không phải là bức ảnh được chỉnh sửa màu hài hòa hơn. Vì vậy chúng em xin phép được kết thúc chương trình tại đây.