HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BỘ MÔN XỬ LÝ ẢNH

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: GHÉP ẢNH PANORAMA

Giảng viên : **PHẠM HOÀNG VIỆT**
Nhóm lớp : **02**
Nhóm bài tập : **05**

Thành viên

Đỗ Văn An	**B22DCCN002**
Đào Mạnh Đạt	**B22DCCN182**

Hà Nội – 2025

MỤC LỤC

PHÂN CÔNG NHIỆM VỤ

Họ tên	Mã sinh viên	Đóng góp	Nhiệm vụ
Đỗ Văn An	B22DCCN002	50%	- Ghép nối các keypoints và tính ma trận homography. - Biến đổi ảnh, thực hiện ghép ảnh Panorama. - Thiết kế slide. - Viết báo cáo.
Đào Mạnh Đạt	B22DCCN182	50%	- Phát hiện các keypoints bằng thuật toán SIFT. - Xây dựng vector mô tả (descriptor) cho các keypoints. - Thiết kế slide. - Viết báo cáo.

PHẦN 1. GIỚI THIỆU BÀI TOÁN

1.1. Đặt vấn đề

Trong nhiếp ảnh và thị giác máy tính, việc thu nhận những bức hình có góc nhìn rộng để bao quát toàn bộ khung cảnh, không gian lớn là nhu cầu rất quan trọng. Tuy nhiên, hầu hết thiết bị chụp ảnh thông thường đều bị giới hạn bởi tiêu cự và góc mở của ống kính. Để tạo ra ảnh góc rộng, người ta thường phải dùng các loại ống kính chuyên dụng, giá thành cao (như ống kính mắt cá – fisheye), nhưng lại dễ gặp nhược điểm là hình bị méo ở các vùng rìa.

Một hướng tiếp cận bằng phần mềm để khắc phục hạn chế này là kỹ thuật ghép ảnh (Image Stitching). Kỹ thuật này cho phép kết hợp nhiều ảnh có vùng chồng lấp, được chụp từ cùng một vị trí nhưng với các hướng nhìn khác nhau, thành một bức ảnh duy nhất có trường nhìn rộng hơn – gọi là ảnh toàn cảnh (Panorama).

Vấn đề then chốt của bài toán ghép ảnh là làm sao để hệ thống có thể tự động xác định tương quan vị trí giữa các bức ảnh, tìm được chính xác các vùng chung để ghép lại mà không xuất hiện vết nối, bóng mờ, kể cả khi ảnh đầu vào có thay đổi về tỷ lệ, góc quay hay điều kiện chiếu sáng. Để làm được điều đó, cần phát hiện và mô tả được các đặc trưng cục bộ (keypoints). Trong số các phương pháp trích xuất đặc trưng, SIFT (Scale-Invariant Feature Transform) là một trong những thuật toán mạnh và ổn định nhất, với khả năng phát hiện điểm đặc trưng bất biến theo tỉ lệ, phép quay và khá bền vững trước thay đổi độ sáng hay nhiễu.

Vì vậy, đề tài "Ghép ảnh Panorama, sử dụng thuật toán SIFT để phát hiện keypoints giữa các ảnh" được lựa chọn nhằm áp dụng SIFT để xây dựng một công cụ ghép ảnh tự động, cho ra ảnh Panorama có độ chính xác và chất lượng cao.

1.2. Mục tiêu của đề tài

Đề tài tập trung vào việc nghiên cứu lý thuyết và triển khai thực nghiệm quy trình ghép ảnh Panorama tự động, với các mục tiêu cụ thể như sau:

a. Về mặt lý thuyết

- Nghiên cứu tổng quan về bài toán Image Stitching và quy trình ghép ảnh.

- Tìm hiểu về thuật toán SIFT (Scale-Invariant Feature Transform): Cách thức phát hiện cực trị trong không gian tỉ lệ, xác định các đặc trưng (keypoints) và tạo bộ mô tả (descriptor).
- Tìm hiểu về các kỹ thuật bổ trợ: Matching (khớp điểm), RANSAC (loại bỏ điểm ngoại lai để tìm ma trận Homography chính xác) và các kỹ thuật Blending (trộn ảnh) để làm mượt đường ghép.

b. Về mặt thực nghiệm

- Cài đặt chương trình ghép ảnh sử dụng ngôn ngữ lập trình Python.
- Thực hiện trích xuất các đặc trưng trên tập ảnh đầu vào sử dụng thuật toán SIFT.
- Thực hiện tìm kiếm các cặp điểm tương đồng và ước lượng ma trận biến đổi hình học (Homography).
- Biến đổi (Warp) và ghép nối các ảnh thành phẩm Panorama hoàn chỉnh.

PHẦN 2. CƠ SỞ LÝ THUYẾT

2.1. Quy trình ghép ảnh tổng quát (Image Stitching Pipeline)

Để ghép nhiều ảnh thành một ảnh Panorama hoàn chỉnh, công cụ cần thực hiện theo một quy trình tuần tự bao gồm các bước chính sau:

1) **Thu nhận ảnh đầu vào:** Tập hợp tối thiểu 2 ảnh chụp cùng cảnh, có vùng chồng lấp, góc nhìn gần giống nhau.
2) **Trích xuất các đặc trưng:** Phát hiện các keypoints trong ảnh và mô tả chúng bằng các vector đặc trưng (descriptor).
3) **Làm khớp ảnh:** Sau khi các đặc trưng đã được phát hiện, tiến hành tìm các cặp điểm tương đồng giữa hai ảnh liên tiếp dựa trên sự giống nhau của vector đặc trưng.
4) **Ước lượng phép biến đổi:** Sau khi tìm các cặp điểm tương đồng, thực hiện tính toán ma trận biến đổi hình học (ví dụ: Homography) giữa các ảnh. Thường sử dụng thuật toán RANSAC để lọc ra các điểm khớp sai và ước tính mô hình biến đổi chính xác nhất.
5) **Biến đổi ảnh:** Phép biến đổi đã ước tính được áp dụng để làm biến đổi các ảnh (warp) về cùng một mặt phẳng tọa độ. Bước này căn chỉnh hình học các ảnh để các đặc trưng tương ứng chồng khớp chính xác.
6) **Trộn ảnh:** Kết hợp các ảnh đã được biến đổi thành một bức ảnh ghép duy nhất, liền mạch.

2.2. Thuật toán SIFT (Scale-Invariant Feature Transform)

2.2.1. Khái niệm

- SIFT là một thuật toán phát hiện các điểm đặc trưng (keypoints) trên ảnh và tính toán các vector mô tả (descriptor) cho những điểm này trong các điều kiện khác nhau.
- SIFT được thiết kế để tìm và mô tả các đặc trưng trong hình ảnh, chẳng hạn như các góc, cạnh hoặc hoa văn vẫn có thể nhận dạng được ngay cả khi hình ảnh được thay đổi kích thước, xoay hoặc chiếu sáng khác nhau.

2.2.2. Ưu điểm

SIFT có một số ưu điểm chính:

- Tính bất biến tỷ lệ:
    + SIFT phát hiện đặc trưng ở nhiều tỉ lệ khác nhau, giúp thuật toán vẫn hoạt động tốt ngay cả khi kích thước ảnh thay đổi.
    + SIFT có thể đạt được điều này bằng cách sử dụng lý thuyết không gian tỷ lệ (scale-space theory): Hình ảnh bị làm mờ ở các mức độ khác nhau để tạo ra nhiều phiên bản. Sau đó, SIFT xem xét các phiên bản này để tìm các mẫu và chi tiết giữ nguyên, bất kể hình ảnh thay đổi về kích thước hoặc độ sắc nét như thế nào.
- Tính bất biến góc xoay:
    + Đối với mỗi điểm đặc trưng được phát hiện, thuật toán SIFT sẽ gán một hướng nhất quán dựa trên độ dốc hình ảnh cục bộ.
    + Bằng cách này, thuật toán vẫn có thể cho phép khớp các đặc trưng ngay cả khi ảnh bị xoay.
- Khả năng phục hồi trước các biến thể hình ảnh khác:
    + Ngoài kích thước và độ xoay, hình ảnh cũng có thể thay đổi theo những cách khác, chẳng hạn như thay đổi độ chiếu sáng. Ánh sáng trên một vật thể có thể chuyển từ sáng sang tối, góc máy ảnh có thể thay đổi một chút hoặc hình ảnh có thể bị mờ hoặc nhiễu.
    + SIFT được xây dựng để xử lý các loại biến thể này. Nó thực hiện điều này bằng cách tập trung vào các keypoint có tính đặc biệt và độ tương phản cao, vì các đặc điểm này ít bị ảnh hưởng bởi sự thay đổi về ánh sáng hoặc sự thay đổi nhỏ về góc nhìn. Do đó, SIFT có xu hướng đáng tin cậy hơn các phương pháp phát hiện cạnh hoặc góc đơn giản, thường không thành công khi điều kiện thay đổi.
- Tính phân biệt cao: Các descriptor của SIFT mô tả những đặc điểm riêng biệt của từng keypoint, giúp việc khớp đặc trưng đáng tin cậy hơn.

2.2.3. Các bước hoạt động của thuật toán

- Phát hiện cực trị trong không gian tỷ lệ:
    + Để phát hiện các đặc trưng không phụ thuộc vào kích thước ảnh, thuật toán xây dựng một không gian tỷ lệ (Scale-space). Đây là một tập hợp các hình ảnh được tạo bằng cách làm mờ dần hình ảnh gốc bằng bộ lọc Gaussian, một kỹ thuật làm mịn và nhóm các kết quả thành các lớp được gọi là quãng tám. Mỗi quãng tám chứa cùng

một hình ảnh ở mức độ mờ tăng dần, trong khi quãng tám tiếp theo là một phiên bản nhỏ hơn của hình ảnh.

+ Không gian tỷ lệ của một hình ảnh $I(x, y)$ được định nghĩa là một hàm $L(x, y, \sigma)$, là kết quả của phép tích chập (convolution) giữa bộ lọc Gaussian $G(x, y, \sigma)$ với ảnh đầu vào:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Trong đó, $G(x, y, \sigma)$ là Gaussian kernel với độ lệch chuẩn σ:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

+ Để phát hiện các điểm keypoint ổn định một cách hiệu quả, SIFT sử dụng sai số Gaussian (Difference of Gaussian - DoG). Hàm này được tính bằng hiệu của hai ảnh được làm mờ bởi Gaussian ở hai tỷ lệ lân cận nhau (cách nhau một hằng số k):

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

+ Bằng cách trừ một ảnh bị mờ khỏi một ảnh khác, SIFT tính toán Sai số Gaussian (DoG), làm nổi bật các khu vực có độ sáng thay đổi mạnh. Các khu vực này được chọn làm điểm đặc trưng tiềm năng vì chúng vẫn nhất quán khi hình ảnh được phóng to hoặc thu nhỏ.

- Định vị các điểm đặc trưng:
  + Các điểm cực trị tìm được ở bước trên là rời rạc. Để xác định vị trí chính xác của keypoint, SIFT sử dụng khai triển Taylor của hàm không gian tỷ lệ $D(x, y, \sigma)$ tại điểm mẫu $\mathbf{x} = (x, y, \sigma)^T$:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

+ Lấy đạo hàm của phương trình trên theo **x** và cho bằng 0, ta tìm được vị trí cực trị tinh chỉnh:

$$\hat{\mathbf{x}} = -\left(\frac{\partial^2 D}{\partial \mathbf{x}^2}\right)^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

+ Loại bỏ các điểm nằm trên biên: Hàm DoG rất nhạy cảm với các biên, nhưng biên không phải là điểm đặc trưng tốt để khớp ảnh. Để loại bỏ biên, SIFT sử dụng ma trận Hessian **H** kích thước 2x2:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Giả sử *a* là giá trị riêng lớn nhất và *b* là giá trị riêng nhỏ nhất của **H**.

+ Gọi $r = a/b$ là tỷ lệ giữa hai giá trị riêng chính. Để điểm đặc trưng được giữ lại, tỷ lệ này phải nhỏ hơn một ngưỡng nhất định. Công thức kiểm tra dựa trên Vết (Trace) và Định thức (Det) của ma trận:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

- Gán hướng cho các keypoint:
  + Sau khi các điểm đặc trưng ổn định được xác định, SIFT làm cho chúng bất biến với phép quay, có nghĩa là chúng vẫn có thể được khớp ngay cả khi hình ảnh bị xoay ngang hoặc lộn ngược. Để thực hiện việc này, SIFT phân tích cách độ sáng thay đổi xung quanh mỗi điểm đặc trưng, được gọi là gradient. Gradient cho thấy cả hướng và cường độ thay đổi về cường độ pixel và cùng nhau chúng nắm bắt cấu trúc cục bộ xung quanh điểm.
  + Độ lớn gradient $\boldsymbol{m(x, y)}$ và hướng $\boldsymbol{\theta(x, y)}$ được tính toán tại mỗi điểm ảnh **(*x*, *y*)** trong ảnh đã làm mờ $\boldsymbol{L}$ có tỷ lệ $\boldsymbol{\sigma}$ gần nhất với tỷ lệ của keypoint:

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

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

  + Một biểu đồ (histogram) hướng gồm 36 bin (mỗi bin bao phủ 10 độ) được xây dựng từ các gradient của các điểm lân cận keypoint. Đỉnh cao nhất của histogram tương ứng với hướng chính của keypoint.
  + Nếu có các đỉnh khác gần bằng độ mạnh, SIFT sẽ gán nhiều hướng cho cùng một điểm đặc trưng. Điều này ngăn chặn các đặc điểm quan trọng bị mất khi các đối tượng xuất hiện ở các góc bất thường. Bằng cách căn chỉnh mỗi điểm đặc trưng với hướng của nó, SIFT đảm bảo rằng các descriptor được tạo ra trong bước tiếp theo vẫn nhất quán.
- Mô tả đặc trưng:
  + Sau khi đã có vị trí, tỷ lệ và hướng, bước cuối cùng là tạo ra vector đặc trưng.
  + SIFT đạt được điều này bằng cách xem xét một vùng vuông nhỏ xung quanh mỗi keypoint, kích thước khoảng 16x16 pixel. Vùng này trước tiên được căn chỉnh theo hướng của keypoint để độ xoay không ảnh hưởng đến nó. Sau đó, vùng này được chia thành một lưới gồm 4x4 ô vuông nhỏ hơn.
  + Trong mỗi ô vuông nhỏ, SIFT đo lường sự thay đổi độ sáng theo các hướng khác nhau. Những thay đổi này được lưu trữ trong một histogram. Mỗi ô vuông có histogram riêng và cùng nhau 16 ô vuông tạo ra 16 histogram.
  + Cuối cùng, các histogram này được kết hợp thành một danh sách duy nhất có độ dài là 128. Danh sách này được gọi là vector đặc trưng.

2.3. Ghép cặp keypoint (Keypoint/Feature Matching)

Ghép cặp keypoint là bước thiết lập tương ứng hình học giữa hai ảnh thông qua sự tương đồng của descriptor. Về mặt lý thuyết, mục tiêu là tìm ánh xạ giữa các keypoint của ảnh thứ nhất và ảnh thứ hai sao cho các cặp được ghép đại diện cho

cùng một cấu trúc vật lý trong cảnh. Chất lượng của bước matching quyết định trực tiếp độ ổn định của việc ước lượng mô hình hình học ở bước sau.

2.3.1. So sánh descriptor và hàm đo tương đồng

Giả sử fi, gi lần lượt là descriptor của keypoint i trong ảnh A và keypoint j trong ảnh B. Với SIFT, descriptor là vector thực (128 chiều), do đó hàm đo phổ biến là khoảng cách Euclid (L2):

$$d(\mathbf{f}_i, \mathbf{g}_j) = \sqrt{\sum_{k=1}^{128} (f_{ik} - g_{jk})^2}$$

Khoảng cách càng nhỏ thì mức độ tương đồng càng lớn. Bài toán matching có thể xem như bài toán tối ưu cục bộ: với mỗi fi, cần tìm gi sao cho d(fi, gi) là nhỏ nhất.

2.3.2. KNN matching và vấn đề mơ hồ của đặc trưng

Trong ảnh thực, nhiều vùng có thể tạo descriptor tương tự nhau (ví dụ hoạ tiết lặp). Khi đó, nearest neighbor đơn thuần dễ dẫn đến match sai. Vì vậy, thực hành tiêu chuẩn là sử dụng K-Nearest Neighbors matching với k=2 với mỗi descriptor ở ảnh A, tìm hai descriptor gần nhất ở ảnh B (theo L2), lần lượt có khoảng cách d1 và d2 với d1 <= d2

2.3.3. Lowe’s ratio test

Để loại bỏ match mơ hồ, Lowe đề xuất ratio test:

$$\frac{d_1}{d_2} < r$$

với r là ngưỡng (thường trong khoảng 0.6–0.8). Ratio test dựa trên lập luận:

- Nếu match tốt nhất thực sự đúng, nó sẽ nổi bật hơn đáng kể so với match tốt nhì, tức d1 < d2, ratio nhỏ.

- Nếu descriptor không phân biệt tốt (nhiều ứng viên tương tự), khi đó d1 ~ d2, ratio gần 1 và match bị loại.

Ratio test do vậy đóng vai trò như một bộ lọc thống kê giúp giảm outlier trước khi đưa vào RANSAC.

2.3.4. Tập correspondences và đặc tính inlier/outlier

Sau khi matching và lọc, ta thu được tập correspondences:

$$\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_i')\}_{i=1}^{N}$$

trong đó xi = (xi, yi) là điểm trong ảnh A, xi′=(xi′,yi′) là điểm trong ảnh B. Tập C bao gồm:

- Inliers: các correspondences phù hợp với một mô hình hình học nhất quán (ví dụ homography).
- Outliers: các correspondences sai do nhiễu, lặp họa tiết, thay đổi chiếu sáng mạnh, vật thể chuyển động hoặc parallax.

Vì outlier là khó tránh khỏi, việc ước lượng mô hình hình học phải là ước lượng bền vững (robust estimation), dẫn đến việc sử dụng RANSAC.

2.4. RANSAC (Random Sample Consensus)

RANSAC là phương pháp ước lượng tham số mô hình trong điều kiện dữ liệu có chứa tỉ lệ outlier đáng kể. Trong ghép ảnh, RANSAC thường được sử dụng để ước lượng **homography** từ tập correspondences sao cho mô hình thu được được hỗ trợ bởi nhiều inlier nhất.

2.4.1. Nguyên lý

Giả sử tồn tại một mô hình M(θ) (với tham số θ) có thể giải thích đúng phần lớn dữ liệu "tốt" (inliers), trong khi outliers không tuân theo quy luật đó. RANSAC vận hành theo nguyên lý:

- Lấy mẫu ngẫu nhiên một tập con tối thiểu để ước lượng mô hình.
- Đánh giá mô hình vừa ước lượng trên toàn bộ dữ liệu bằng một tiêu chí sai số.
- Tập điểm phù hợp với mô hình (sai số nhỏ hơn ngưỡng) được xem là inlier.
- Lặp lại và chọn mô hình có consensus (số inlier) lớn nhất.

2.4.2. RANSAC cho homography

Homography có 8 bậc tự do (do ma trận 3×3 xác định tới một hệ số tỉ lệ), vì vậy cần tối thiểu **4 correspondence** để ước lượng (mỗi correspondence cung cấp hai ràng buộc độc lập). RANSAC cho homography gồm các bước:

1. **Sampling**: chọn ngẫu nhiên 4 cặp điểm từ C.
2. **Model fitting**: ước lượng homography H từ 4 cặp điểm.
3. **Scoring (đếm inlier)**: với mỗi correspondence (xi,xi′), tính điểm dự đoán xi’ từ H và sai số chiếu lại:

$$\hat{\mathbf{x}}_i' \sim H\mathbf{x}_i, \qquad e_i = \|\mathbf{x}_i' - \hat{\mathbf{x}}_i'\|$$

Nếu ei < τ (ngưỡng reprojection threshold) thì correspondence đó là inlier.
4) **Lặp**: thực hiện nhiều vòng, lưu mô hình có số inlier lớn nhất.
5) **Tối ưu lại**: sau khi có tập inlier tốt nhất, ước lượng lại H bằng toàn bộ inlier để tăng độ chính xác.

2.4.3. Vai trò của ngưỡng và số vòng lặp

- **Ngưỡng τ** điều khiển mức chặt của tiêu chí đồng thuận: τ\tauτ quá nhỏ có thể loại nhầm inlier; τ quá lớn có thể nhận cả outlier.

- **Số vòng lặp** cần đủ lớn để xác suất chọn được mẫu toàn inlier là cao. Với tỉ lệ inlier w, xác suất thành công mong muốn p, kích thước mẫu tối thiểu s=4, số vòng lặp gần đúng:

$$N = \frac{\log(1-p)}{\log(1-w^s)}$$

Khi w thấp (matching kém), N tăng nhanh, do đó chất lượng matching (ratio test) có ảnh hưởng trực tiếp đến độ hiệu quả của RANSAC.

2.5. Ma trận Homography

Trong ghép ảnh, Homography H là ma trận 3×3 ánh xạ một điểm trên ảnh 1 sang ảnh 2 trong tọa độ đồng nhất:

$$\mathbf{x}' \sim H\mathbf{x}, \quad \mathbf{x} = (x, y, 1)^T,\ \mathbf{x}' = (x', y', 1)^T$$

2.5.1. Công thức tổng quát

Giả sử một điểm P trong ảnh nguồn có tọa độ (x, y) và điểm tương ứng P’ trong ảnh đích có tọa độ (x', y'). Trong hệ tọa độ thuần nhất, chúng được biểu diễn lần lượt là x = [x, y, 1]T và x' = [x', y', 1]T

Mối quan hệ giữa chúng được định nghĩa:

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.5.2. Chuyển đổi về tọa độ Descartes

Để thu được tọa độ thực tế trên ảnh đích, ta thực hiện phép chia cho thành phần thứ 3 của vector kết quả:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

2.5.3. Các đặc tính kỹ thuật quan trọng

1. Bậc tự do (Degrees of Freedom)

   Mặc dù ma trận H có 9 phần tử, nhưng nó chỉ có **8 bậc tự do**.

   - **Lý do:** Tọa độ thuần nhất không thay đổi khi nhân với một hằng số khác 0 (tính chất bất biến tỉ lệ). Do đó, ta thường chuẩn hóa ma trận bằng cách đặt h33 = 1
   - **Hệ quả:** Cần giải hệ phương trình tìm 8 ẩn số (h11 đến h32).

2. Yêu cầu dữ liệu đầu vào

   Để tính toán duy nhất một ma trận Homography, cần ít nhất **4 cặp điểm tương ứng** giữa hai hình ảnh.

   - **Điều kiện ràng buộc:** Trong 4 điểm này, không được có 3 điểm nào thẳng hàng (collinear).

3. So sánh với biến đổi Affine

   Khác với biến đổi Affine (bảo toàn tính song song của các đường thẳng), Homography là biến đổi xạ ảnh (Projective transform). Nó có khả năng làm biến dạng hình thang, cho phép mô phỏng hiệu ứng luật xa gần (perspective distortion) – nơi các đường thẳng song song hội tụ tại một điểm.

PHẦN 3. THIẾT KẾ CHƯƠNG TRÌNH

3.1. Ngôn ngữ và Thư viện được sử dụng

- **Ngôn ngữ được sử dụng:** Python 3.
- **Một số thư viện chính:**
  + **NumPy:** Thư viện tính toán khoa học, sử dụng để xử lý mảng và ma trận ảnh, tính toán đại số tuyến tính.
  + **OpenCV:** Thư viện xử lý ảnh và thị giác máy tính, sử dụng để đọc/ghi ảnh, chuyển đổi không gian màu, triển khai SIFT, so khớp đặc trưng.
  + **PIL:** Thư viện xử lý ảnh, hỗ trợ chuyển đổi định dạng, resize ảnh, là cầu nối giữa OpenCV và giao diện.
  + **Tkinter:** Thư viện tiêu chuẩn của Python để xây dựng giao diện đồ họa (GUI).

3.2. Cấu trúc của dự án

Cấu trúc tổng quan của dự án được chia thành các phần chính như sau:

```
BTL_XLA/
├── Panorama.py          # Xử lý core: SIFT, matching, homography, ghép panorama
├── app.py               # Ứng dụng web Flask, gọi hàm panorama/panorama_multiple
│
├── static/              # Tài nguyên tĩnh cho giao diện web
│   ├── css/
│   │   └── style.css
│   └── js/
│       └── script.js
│
├── templates/           # Giao diện HTML
│   └── index.html
│
├── uploads/             # Thư mục lưu ảnh người dùng upload
```

- **Module Panorama.py (core xử lý ảnh)**: Chứa toàn bộ logic xử lý ảnh, phát hiện và mô tả đặc trưng, so khớp đặc trưng, tính homography và ghép ảnh Panorama. Các chức năng chính gồm:
  - **Phát hiện & mô tả đặc trưng SIFT**: Tìm keypoints và xây dựng vector mô tả (descriptor) cho từng điểm đặc trưng.

- **Matching đặc trưng**: So khớp các descriptor giữa hai ảnh để tìm cặp điểm tương ứng.
- **Stitching (ghép Panorama)**: Tính ma trận Homography bằng RANSAC, warp ảnh và ghép thành ảnh Panorama.
- **Xử lý hậu kỳ**: Blend vùng chồng lấp và cắt bỏ các vùng đen để cải thiện chất lượng ảnh cuối.

- **Module app.py (giao diện web)**: Xây dựng giao diện web cho công cụ ghép ảnh Panorama sử dụng Flask.
  - Nhận ảnh người dùng upload, tiền xử lý và gọi các hàm trong Panorama.py.
  - Trả về ảnh gốc, ảnh so khớp và ảnh Panorama dạng base64 để hiển thị trên trình duyệt.
  - **Thư mục static/ và templates/**:
  - templates/index.html: Giao diện web chính cho phép người dùng chọn và upload ảnh.
  - static/css/style.css, static/js/script.js: Định nghĩa giao diện và tương tác phía client cho ứng dụng ghép ảnh Panorama.

3.3. Xây dựng các hàm xử lý ảnh

3.3.1. Phát hiện và mô tả các đặc trưng trong ảnh

- Chuyển ảnh về ảnh Grayscale:

```
187     scr_gray = cv2.cvtColor(img1, cv2.COLOR_RGB2GRAY)
188     tar_gray = cv2.cvtColor(img2, cv2.COLOR_RGB2GRAY)
```

- Hàm nhận một ảnh dạng mảng NumPy (numpy.ndarray) và trả về ảnh grayscale. Nếu ảnh đầu vào là màu (BGR hoặc BGRA), hàm sẽ chuyển đổi sang grayscale.
- Kiểm tra số chiều của mảng ảnh:
- Nếu mảng có 2 chiều (height, width): ảnh đã là grayscale, trả về ảnh gốc.
- Nếu mảng có 3 chiều (height, width, channels): ảnh màu, xác định số kênh:
- Nếu channels = 3: ảnh BGR (định dạng mặc định của OpenCV). Dùng cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) để chuyển sang grayscale.
- Nếu channels = 4: ảnh BGRA (BGR có kênh alpha). Dùng cv2.cvtColor(image, cv2.COLOR_BGRA2GRAY) để chuyển sang grayscale.
- Nếu số chiều hoặc số kênh không hợp lệ, hàm ném lỗi (ValueError) để báo không thể chuyển đổi.

- Khởi tạo mô hình phát hiện đặc trưng sử dụng SIFT

```
        # SIFT đầy đủ đã được tối ưu và test kỹ
        try:
            Sift_detect = cv2.xfeatures2d.SIFT_create()
        except:
            # Nếu không có xfeatures2d, dùng SIFT thường (OpenCV 4.5+)
            Sift_detect = cv2.SIFT_create()
        k1, d1 = Sift_detect.detectAndCompute(scr_gray, None)
        k2, d2 = Sift_detect.detectAndCompute(tar_gray, None)
```

- Hàm trả về một đối tượng cv2.SIFT. Đối tượng này cung cấp các phương thức để phát hiện điểm đặc trưng và tính descriptor cho ảnh.
- Hàm cv2.SIFT_create() được dùng để tạo đối tượng cv2.SIFT với các tham số chính:
- **nfeatures = n_features** (int): số lượng keypoint tối đa mà thuật toán giữ lại (ưu tiên theo độ tin cậy). Tham số này giúp giới hạn số điểm đặc trưng để giảm chi phí xử lý và so khớp. Nếu n_features = 0, OpenCV không giới hạn số lượng keypoint.
- **contrastThreshold = contrast_threshold** (float): ngưỡng loại bỏ các điểm có độ tương phản thấp. Giá trị càng nhỏ → càng nhiều điểm được giữ lại (nhạy hơn với vùng mờ/nhiễu). Giá trị càng lớn → chỉ các điểm có độ tương phản đủ cao mới được chấp nhận.
- **edgeThreshold = edge_threshold** (float): ngưỡng loại bỏ các điểm nằm trên biên/mép mạnh. Ngưỡng nhỏ → nhiều điểm gần biên bị loại bỏ; ngưỡng lớn → nhiều điểm ở vùng biên vẫn được giữ.
- **sigma = sigma** (float): độ lệch chuẩn của bộ lọc Gaussian dùng cho cấp độ đầu tiên trong không gian tỷ lệ (scale-space). Sigma càng lớn → ảnh được làm mượt nhiều hơn, bớt chi tiết nhỏ; sigma nhỏ → giữ lại nhiều chi tiết nhỏ hơn.

3.3.2. Xây dựng các hàm ghép panorama

- Thực hiện ghép cặp các keypoint

```
    # Sử dụng KNN matcher với Lowe's ratio test (tốt hơn match đơn giản)
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)
    knn_matches = bf.knnMatch(d1, d2, k=2)

    # Áp dụng Lowe's Ratio Test để lọc matches tốt
    ratio_thresh = 0.75
    good_matches = []
    for m, n in knn_matches:
        if m.distance < ratio_thresh * n.distance:
            good_matches.append(m)

    # Kiểm tra nếu không có đủ matches
    if len(good_matches) < 10:
        raise ValueError("Không tìm thấy đủ điểm tương đồng giữa 2 ảnh. Vui lòng chọn 2 ảnh có phần chồng lấp (overlap) với nhau.")

    matches = good_matches
```

4. Hàm `match_descriptors` nhận hai bộ descriptor và tìm các cặp điểm tương ứng giữa chúng bằng Brute-Force Matcher với khoảng cách L2 (Euclidean).
5. Sau khi kiểm tra dữ liệu hợp lệ, hàm dùng k-nearest neighbors (kNN) với k=2 để tìm hai ứng viên gần nhất cho mỗi descriptor. Tiếp theo, áp dụng Lowe's ratio test để lọc nhiễu: chỉ giữ các

cặp mà khoảng cách của ứng viên tốt nhất nhỏ hơn đáng kể so với ứng viên thứ hai (theo hệ số ratio). Các match không đạt tỷ lệ này sẽ bị loại bỏ để tăng độ chính xác.

- Hàm tính ma trận H

Panorama.py

```
k1, d1 = Sift_detect.detectAndCompute(scr_gray, None)
k2, d2 = Sift_detect.detectAndCompute(tar_gray, None)
```

2. Bước 2 - Matching descriptors:

Panorama.py

```
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)
knn_matches = bf.knnMatch(d1, d2, k=2)
ratio_thresh = 0.75
good_matches = []
for m, n in knn_matches:
    if m.distance < ratio_thresh * n.distance:
        good_matches.append(m)
if len(good_matches) < 10:
    raise ValueError(...)
matches = good_matches
```

3. Bước 3 - Lấy tọa độ từ keypoints:

Panorama.py

```
keypoint_1 = np.float32([kp.pt for kp in k1])
keypoint_2 = np.float32([kp.pt for kp in k2])
pts1 = np.float32([keypoint_1[m.queryIdx] for m in matches])
pts2 = np.float32([keypoint_2[m.trainIdx] for m in matches])
```

4. Bước 4 - RANSAC tìm Homography:

Panorama.py

```
H1, mask1 = cv2.findHomography(pts2, pts1, cv2.RANSAC, 5.0)
```

Sau khi đã so khớp được các keypoint, tiến hành ước lượng ma trận H dựa trên cặp điểm đã so khớp với thuật toán RANSAC

- Hàm ghép 2 ảnh

- Gọi hàm estimate_homography để tìm ra ma trận biến đổi H.
- Ma trận H này chứa thông tin cần thiết để wrap góc nhìn của img_right sao cho khớp với mặt phẳng của img_left.

Bước 2: Tính toán kích thước khung Canvas tổng thể

- Lấy tọa độ 4 góc của img_right, sau đó dùng hàm cv2.perspectiveTransform áp dụng ma trận H lên 4 góc này để xem sau khi biến đổi, ảnh phải sẽ nằm ở đâu trong không gian.
- Kết hợp toạ độ 4 góc mới này với 4 góc của img_left.
- Tìm ra xmin, ymin, xmax, ymax bao trùm toàn bộ cả hai ảnh. Việc này giúp xác định kích thước ảnh panorama cuối cùng để đảm bảo không phần nào bị cắt mất.

Bước 3: Xử lý toạ độ âm (Translation Matrix)

- Khi biến đổi, ảnh có thể bị lệch sang vùng toạ độ âm (ví dụ x = -100).
- Đoạn code tính toán vector dịch chuyển translation = [-xmin, -ymin] và tạo ma trận dịch chuyển T.
- Mục đích là để dời toàn bộ tọa độ về dương (bắt đầu từ 0,0) để hiển thị được trên ảnh máy tính.

Bước 4: Biến đổi ảnh phải (Warping)

- Sử dụng cv2.warpPerspective để biến đổi img_right.
- Lưu ý quan trọng: Ma trận dùng để warp là tích của T x H. Điều này có nghĩa là vừa thực hiện bẻ cong ảnh theo Homography, vừa dịch chuyển nó vào đúng khung hình dương.
- Kích thước ảnh đầu ra là (xmax - xmin, ymax - ymin).

Bước 5: Dán ảnh trái vào

- Cuối cùng, lấy dữ liệu pixel gốc của img_left dán đè lên ảnh panorama vừa tạo.
- Vị trí dán bắt đầu từ toạ độ dịch chuyển translation để khớp với hệ toạ độ mới.

- Kỹ thuật Sequential Stitching (Ghép tuần tự)

```python
# Sử dụng KNN matcher với Lowe's ratio test (tốt hơn match đơn giản)
bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=False)
knn_matches = bf.knnMatch(d1, d2, k=2)

# Áp dụng Lowe's Ratio Test để lọc matches tốt
ratio_thresh = 0.75
good_matches = []
for m, n in knn_matches:
    if m.distance < ratio_thresh * n.distance:
        good_matches.append(m)

# Kiểm tra nếu không có đủ matches
if len(good_matches) < 10:
    raise ValueError("Không tìm thấy đủ điểm tương đồng giữa 2 ảnh. Vui lòng chọn 2 ảnh có phần chồng lấp (overlap) với nhau.")

matches = good_matches
```

```python
        keypoint_1 = np.float32([kp.pt for kp in k1])
        keypoint_2 = np.float32([kp.pt for kp in k2])
        pts1 = np.float32([keypoint_1[m.queryIdx] for m in matches])
        pts2 = np.float32([keypoint_2[m.trainIdx] for m in matches])

        # Phân tích vị trí keypoints để xác định thứ tự
        # Nếu ảnh 1 ở bên trái, các điểm matching ở ảnh 1 sẽ ở bên phải
        # và các điểm matching ở ảnh 2 sẽ ở bên trái
        h1, w1 = img1.shape[:2]
        h2, w2 = img2.shape[:2]

        # Tính trung bình vị trí X của các điểm matching
        avg_x1 = np.mean(pts1[:, 0])
        avg_x2 = np.mean(pts2[:, 0])

        # Tỷ lệ vị trí so với chiều rộng ảnh
        ratio1 = avg_x1 / w1  # Tỷ lệ trong ảnh 1
        ratio2 = avg_x2 / w2  # Tỷ lệ trong ảnh 2

        # Nếu ảnh 1 ở trái: ratio1 cao (điểm ở bên phải ảnh 1), ratio2 thấp (điểm ở bên trái ảnh 2)
        # Nếu ảnh 1 ở phải: ratio1 thấp, ratio2 cao
```

```
    h1, w1 = img1.shape[:2]
    h2, w2 = img2.shape[:2]

    # Tính trung bình vị trí X của các điểm matching
    avg_x1 = np.mean(pts1[:, 0])
    avg_x2 = np.mean(pts2[:, 0])

    # Tỷ lệ vị trí so với chiều rộng ảnh
    ratio1 = avg_x1 / w1  # Tỷ lệ trong ảnh 1
    ratio2 = avg_x2 / w2  # Tỷ lệ trong ảnh 2

    # Nếu ảnh 1 ở trái: ratio1 cao (điểm ở bên phải ảnh 1), ratio2 thấp (điểm ở bên trái ảnh 2)
    # Nếu ảnh 1 ở phải: ratio1 thấp, ratio2 cao
    is_img1_left = (ratio1 > 0.5 and ratio2 < 0.5) or (ratio1 > ratio2 + 0.1)

    # Thử cả 2 cách và chọn kết quả tốt hơn
    results = []

    # Cách 1: img1 trái, img2 phải
    try:
        H1, mask1 = cv2.findHomography(pts2, pts1, cv2.RANSAC, 5.0)
```

```
ratio1 = avg_x1 / w1  # Tỷ lệ trong ảnh 1
ratio2 = avg_x2 / w2  # Tỷ lệ trong ảnh 2

# Nếu ảnh 1 ở trái: ratio1 cao (điểm ở bên phải ảnh 1), ratio2 thấp (điểm ở bên trái ảnh 2)
# Nếu ảnh 1 ở phải: ratio1 thấp, ratio2 cao
is_img1_left = (ratio1 > 0.5 and ratio2 < 0.5) or (ratio1 > ratio2 + 0.1)

# Thử cả 2 cách và chọn kết quả tốt hơn
results = []

# Cách 1: img1 trái, img2 phải
try:
    H1, mask1 = cv2.findHomography(pts2, pts1, cv2.RANSAC, 5.0)
    if H1 is not None:
        inliers1 = np.sum(mask1) if mask1 is not None else 0
        # Kiểm tra homography hợp lệ (không bị biến dạng quá mức)
        if inliers1 > 10:  # Ít nhất 10 inliers
            img_result1 = cv2.warpPerspective(img2, H1, (w1 + w2, max(h1, h2)))
            # Blend thay vì ghi đè để tránh viền đen
            img_result1 = blend_images(img1, img_result1, h1, w1, h2, w2, True)
```

```
360     # Ghép từng ảnh một
361     img_soKhop_list = []
362     current_panorama = images[0]
363     failed_images = []
364     successful_count = 1  # Đếm số ảnh đã ghép thành công (bắt đầu với ảnh đầu tiên)
365
366     for i in range(1, len(images)):
367         try:
368             img_soKhop, panorama_result = panorama(current_panorama, images[i])
369             img_soKhop_list.append(img_soKhop)
370             current_panorama = panorama_result
371             successful_count += 1
372         except Exception as e:
373             # Track ảnh không ghép được
374             failed_images.append(i + 1)  # +1 vì index bắt đầu từ 0, nhưng số thứ tự từ 1
375             # Nếu quá nhiều ảnh không ghép được, raise error
376             if successful_count < 2:
377                 raise ValueError(f"Không thể ghép ảnh {i+1} với panorama hiện tại. {str(e)}")
378             # Nếu chỉ một vài ảnh không ghép được, tiếp tục với các ảnh còn lại
379             continue
380
```

Sequential Stitching (Ghép ảnh tuần tự) là kỹ thuật ghép nhiều ảnh bằng cách ghép từng ảnh một theo thứ tự từ trái sang phải (hoặc ngược lại), thay vì chọn ảnh giữa làm gốc như Center-Based Stitching.

2. Nguyên lý của Sequential Stitching (Ghép tuần tự)

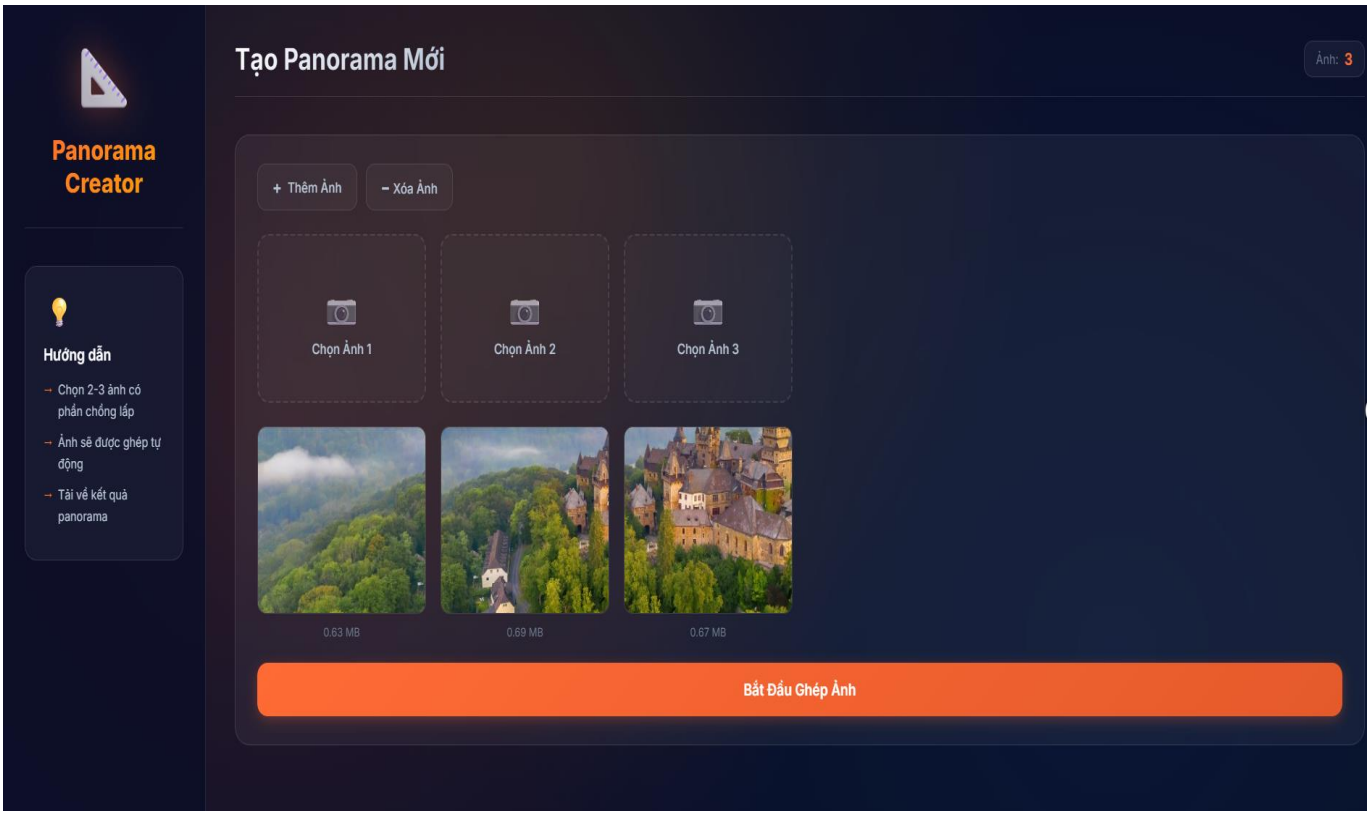
Phương pháp này ghép ảnh theo thứ tự từ trái sang phải (hoặc ngược lại).Ví dụ với 5 ảnh: I_1, I_2, I_3, I_4, I_5:

- Bước 1: Ghép $I_1 + I_2 \rightarrow \text{panorama}_{12}$
- Bước 2: Ghép $\text{panorama}_{12} + I_3 \rightarrow \text{panorama}_{123}$
- Bước 3: Ghép $\text{panorama}_{123} + I_4 \rightarrow \text{panorama}_{1234}$
- Bước 4: Ghép $\text{panorama}_{1234} + I_5 \rightarrow$ panorama cuối cùng

Mỗi bước tính ma trận homography giữa panorama hiện tại và ảnh tiếp theo, rồi warp ảnh tiếp theo về panorama hiện tại.**3. Cách tính ma trận H**Với mỗi cặp ảnh (panorama hiện tại, ảnh tiếp theo), ta tính ma trận homography H bằng RANSAC từ các cặp điểm tương ứng đã match, sau đó dùng cv2.warpPerspective để biến đổi ảnh tiếp theo về hệ tọa độ của panorama hiện tại.

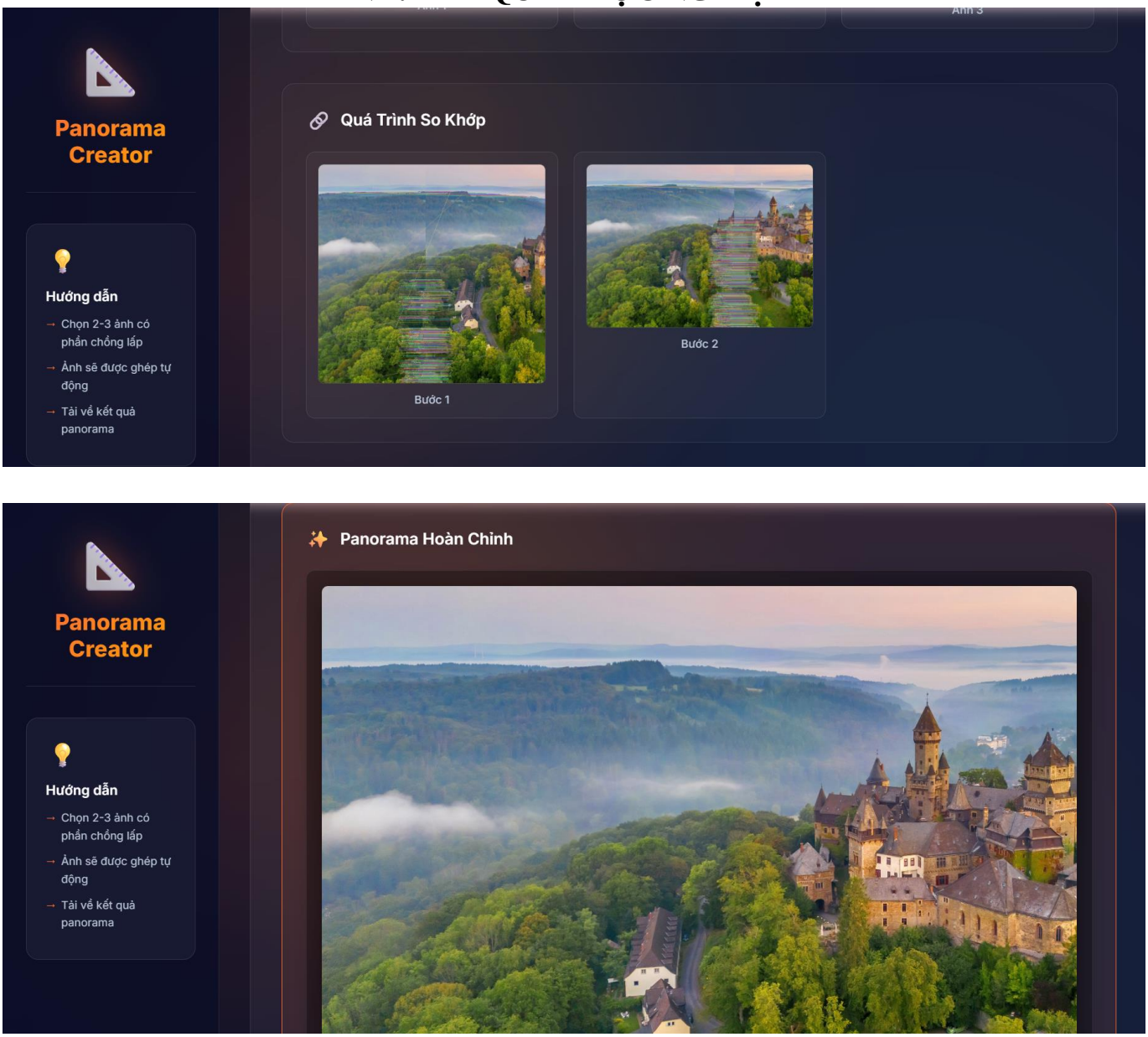
5.2. Xây dựng giao diện cho công cụ

- **Tùy chỉnh các tham số thuật toán:**
- Điều chỉnh số lượng keypoints tối đa (n_features)
- Thay đổi ngưỡng tương phản (contrast_threshold) để kiểm soát độ nhạy phát hiện đặc trưng
- Điều chỉnh ngưỡng biên (edge_threshold) để lọc keypoints ở vùng biên
- Tùy chỉnh tham số sigma của Gaussian trong scale-space
- Điều chỉnh tỷ lệ trong Lowe's ratio test (ratio) để lọc matches
- Thay đổi ngưỡng RANSAC (ransac_thresh) để tối ưu độ chính xác tính homography
- **Preview các ảnh trước khi ghép:**
- Hiển thị danh sách ảnh đã chọn với thumbnail
- Cho phép xem từng ảnh ở kích thước lớn hơn
- Hiển thị thứ tự ảnh sẽ được ghép
- Cảnh báo nếu ảnh không đủ overlap hoặc không phù hợp
- **Xử lý và hiển thị kết quả:**
- Hiển thị ảnh so khớp (matches) giữa các cặp ảnh liên tiếp
- Hiển thị ảnh Panorama cuối cùng sau khi ghép
- Cho phép lưu ảnh Panorama ra file
- Hiển thị thông báo lỗi/nhắc nhở nếu quá trình ghép không thành công

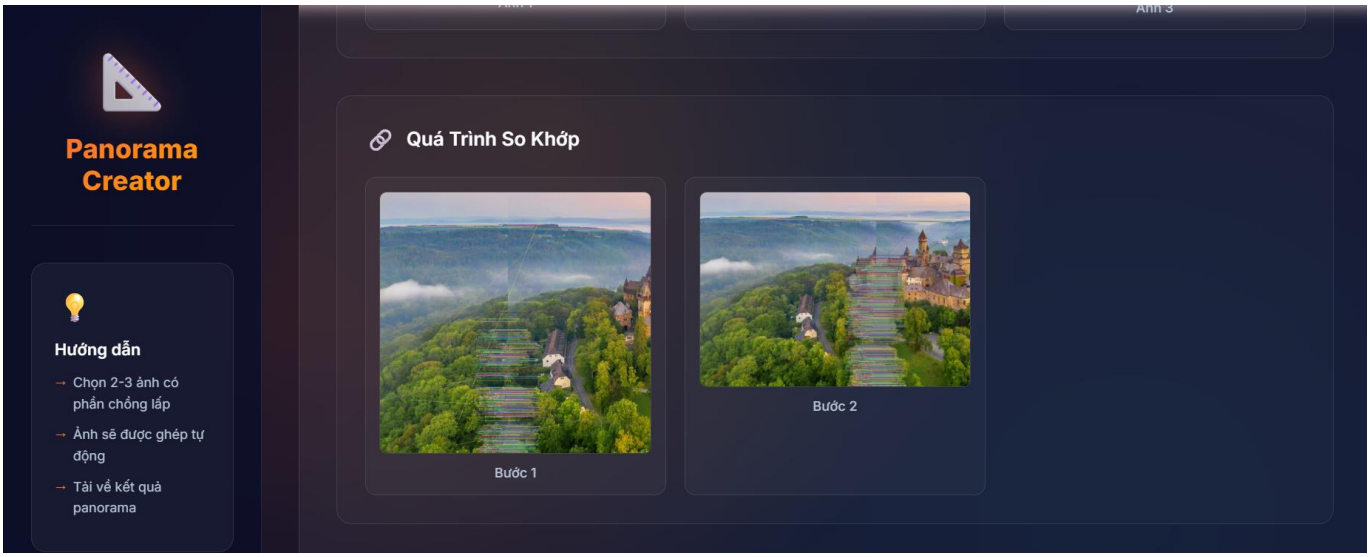


Giao diện công cụ

PHẦN 4. KẾT QUẢ THỰC NGHIỆM



Ghép 2-3 ảnh có chồng lấp



So khớp các keypoint

PHẦN 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được

Đề tài "Ghép ảnh Panorama, sử dụng thuật toán SIFT để phát hiện keypoints giữa các ảnh" đã hoàn thành các mục tiêu nghiên cứu và phát triển đã đề ra. Thông qua quá trình nghiên cứu lý thuyết và triển khai thực nghiệm, nhóm đã hoàn thành các mục tiêu đề ra ban đầu với các kết quả cụ thể như sau:

- Xây dựng công cụ ghép ảnh Panorama: Đã phát triển hoàn thiện một công cụ phần mềm sử dụng ngôn ngữ Python và thư viện OpenCV, có giao diện đồ họa (GUI) thân thiện dựa trên Tkinter. Công cụ cho phép người dùng nhập ảnh, tùy chỉnh tham số và xem trước kết quả.
- Ứng dụng thuật toán SIFT và Homography: Đã ứng dụng thuật toán SIFT để phát hiện các điểm đặc trưng bất biến với tỷ lệ và góc xoay. Hệ thống thực hiện ghép cặp chính xác thông qua phương pháp KNN kết hợp với kiểm chứng Lowe's ratio test.
- Xử lý nhiễu và biến đổi: Việc tích hợp giải thuật RANSAC đã giúp loại bỏ hiệu quả các cặp điểm khớp sai, đảm bảo ma trận Homography tìm được là tối ưu cho phép biến đổi.

Triển khai kỹ thuật ghép ảnh tuần tự: Phương pháp này lấy ảnh đầu tiên làm base (panorama ban đầu), với mỗi ảnh tiếp theo, Tính ma trận homography giữa panorama hiện tại và ảnh tiếp theo bằng RANSAC từ các cặp điểm đã match, Warp ảnh tiếp theo về hệ tọa độ của panorama hiện tại bằng cv2.warpPerspective. Blend hai ảnh ở vùng overlap bằng linear gradient để tránh viền đen.Panorama mới trở thành base cho bước tiếp theo. Quá trình lặp lại cho đến khi ghép hết tất cả ảnh, tạo ra panorama cuối cùng.
Ưu điểm:Đơn giản, dễ triển khai, phù hợp khi ảnh được chụp tuần tự theo một hướng.Xử lý nhanh với số lượng ảnh vừa phải.
Nhược điểm: Biến dạng có thể tích tụ ở các ảnh xa ảnh đầu tiên, đặc biệt khi ghép nhiều ảnh, không tối ưu bằng center-based khi có nhiều ảnh (5+ ảnh).

5.2. Hạn chế

Mặc dù công cụ đã hoạt động ổn định trên các tập dữ liệu thử nghiệm, nhưng vẫn tồn tại một số hạn chế nhất định:

- Tốc độ xử lý: Do thuật toán SIFT có độ phức tạp tính toán cao (phải xây dựng không gian tỷ lệ, vector mô tả 128 chiều,...), thời gian xử lý sẽ tăng lên đáng kể khi kích thước ảnh đầu vào lớn hoặc số lượng ảnh cần ghép nhiều.
- Vấn đề về đường biên: Hiện tại công cụ tập trung vào việc căn chỉnh hình

học Trong các trường hợp ánh sáng thay đổi mạnh giữa các bức ảnh hoặc có vật thể chuyển động, đường ghép nối có thể vẫn bị lộ hoặc không mượt mà hoàn toàn do chưa áp dụng các thuật toán pha trộn phức tạp (như Multi-band Blending).

- Biến dạng phối cảnh: Ma trận Homography thực hiện biến đổi xạ ảnh phẳng. Khi góc chụp quá rộng (ví dụ gần 180 độ hoặc hơn), các vùng biên của ảnh Panorama sẽ bị kéo giãn và méo nghiêm trọng.

5.3. Hướng phát triển trong tương lai

Để khắc phục các hạn chế trên và nâng cao chất lượng của công cụ, hướng phát triển tiếp theo của đề tài bao gồm:

- Tối ưu hóa thuật toán và hiệu năng: Nghiên cứu áp dụng các thuật toán trích xuất đặc trưng nhanh hơn như SURF (Speeded Up Robust Features) hoặc ORB (Oriented FAST and Rotated BRIEF) để hướng tới khả năng ghép ảnh thời gian thực (real-time).
- Cải thiện kỹ thuật trộn ảnh: Tích hợp kỹ thuật Gain Compensation để cân bằng độ sáng giữa các ảnh và Pyramid Blending để loại bỏ hoàn toàn các vết nối, giúp ảnh đầu ra liền mạch tự nhiên hơn.
- Hỗ trợ phép chiếu hình trụ hoặc hình cầu: Thay vì chỉ sử dụng Homography phẳng, sẽ chuyển sang chiếu ảnh lên mặt trụ hoặc mặt cầu trước khi ghép. Điều này sẽ giải quyết triệt để vấn đề méo hình khi ghép ảnh toàn cảnh 360 độ.
- Tự động hóa hoàn toàn: Bổ sung tính năng tự động sắp xếp thứ tự ảnh đầu vào (không yêu cầu người dùng phải chọn theo thứ tự) và tự động gợi ý các tham số tối ưu dựa trên phân tích dữ liệu ảnh.

TÀI LIỆU THAM KHẢO

[1] SIFT: Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

[2] RANSAC: Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395.

[3] Warping - Blending: Image Alignment and Stitching: A Tutorial. Foundations and Trends in Computer Graphics and Vision, 2(1), 1-104.

[4] OpenCV: https://opencv.org/

[5] NumPy: https://numpy.org/

[6] Pillow: https://python-pillow.github.io/

[7] Tkinter: https://docs.python.org/3/library/tkinter.html