

# Project 5: Panoramic photos (Image Stitching)

Nguyễn Trang Sỹ Lâm

Ngày 25 tháng 11 năm 2024

Môn học: Xử lý Ảnh số và Thị giác Máy tính  
Giảng viên hướng dẫn: Võ Thanh Hùng

## 1 Giới thiệu

Trong thời đại công nghệ hiện nay, việc ghép nối các hình ảnh lại với nhau để tạo thành một bức ảnh toàn cảnh (panorama) không chỉ là một ứng dụng phổ biến trong nhiếp ảnh mà còn đóng vai trò quan trọng trong các lĩnh vực như bản đồ số, thực tế ảo (VR), hệ thống hỗ trợ lái xe tự động, và nhận dạng không gian. Quá trình này được gọi là **panorama stitching** và bao gồm việc kết hợp các hình ảnh chồng lấp sao cho bức ảnh kết quả liền mạch và tự nhiên.

Quá trình ghép ảnh toàn cảnh yêu cầu sự kết hợp của nhiều bước phức tạp trong xử lý ảnh số, bao gồm:

- **Phát hiện và mô tả đặc trưng (Feature Detection and Description):** Xác định các điểm đặc biệt (keypoints) trong từng ảnh và mô tả chúng dưới dạng vector để hỗ trợ việc so khớp.
- **So khớp đặc trưng (Feature Matching):** Tìm kiếm các điểm tương đồng giữa hai hình ảnh dựa trên các vector đặc trưng đã được trích xuất.
- **Ước lượng phép biến đổi Homography (Homography Estimation):** Tính toán phép biến đổi hình học để căn chỉnh hai hình ảnh với nhau.
- **Biến đổi hình ảnh (Image Warping):** Sử dụng ma trận homography để làm biến dạng hình ảnh, sao cho chúng khớp nhau trên cùng một mặt phẳng phối cảnh.

- **Kết hợp và hòa trộn (Blending):** Loại bỏ đường biên, chồng lấp và tạo ra một bức ảnh duy nhất, mượt mà.

Trong bài toán này, chúng ta sẽ sử dụng thư viện **OpenCV** để hiện thực toàn bộ các bước trên. Với sự hỗ trợ của các thuật toán hiện đại như ORB và SIFT, quá trình ghép nối ảnh không chỉ đạt được độ chính xác cao mà còn tối ưu về thời gian xử lý. Mục tiêu cuối cùng là tạo ra một bức ảnh toàn cảnh từ hai hoặc nhiều ảnh đầu vào sao cho:

- Các điểm chồng lấp giữa các ảnh được căn chỉnh chính xác.
- Các vùng chuyển tiếp giữa ảnh liền mạch và tự nhiên.
- Ảnh kết quả mang tính thẩm mỹ cao, không xuất hiện lỗi thị giác.

## 2 Công thức toán

### 2.1 Phát hiện và Mô tả Đặc trưng

Phát hiện và mô tả đặc trưng là bước đầu tiên và quan trọng nhất trong quá trình ghép ảnh. Mục tiêu của bước này là xác định các điểm đặc trưng (keypoints) trong từng ảnh và mô tả chúng bằng các vector để hỗ trợ việc so khớp. Hai thuật toán phổ biến được sử dụng trong bài toán này là ORB (Oriented FAST and Rotated BRIEF) và SIFT (Scale-Invariant Feature Transform).

#### 2.1.1 ORB (Oriented FAST and Rotated BRIEF)

**ORB** kết hợp hai phương pháp chính là FAST để phát hiện đặc trưng và BRIEF để mô tả đặc trưng.

**1. Phát hiện đặc trưng với FAST (Features from Accelerated Segment Test)** FAST phát hiện các điểm đặc trưng (thường là các góc) trong ảnh dựa trên sự thay đổi độ sáng trong một vùng lân cận xung quanh điểm đó. Cụ thể:

- Xét một điểm ảnh  $p$  với giá trị độ sáng  $I(p)$ , điểm này được kiểm tra với một vòng tròn gồm 16 điểm lân cận.
- Điểm  $p$  được xác định là một đặc trưng nếu có ít nhất  $N$  điểm trên vòng tròn này có độ sáng khác biệt lớn hơn một ngưỡng  $T$  so với  $I(p)$ .

$$|I(p_i) - I(p)| > T, \quad \forall p_i \in \text{Vòng tròn FAST}$$

Để tăng cường tính ổn định, ORB sử dụng thang đo Harris để xếp hạng các điểm đặc trưng và giữ lại các điểm có giá trị cao nhất.

**2. Mô tả đặc trưng với BRIEF (Binary Robust Independent Elementary Features)** BRIEF tạo mô tả đặc trưng nhị phân bằng cách so sánh cường độ sáng của các cặp điểm trong một vùng lân cận của từng điểm đặc trưng:

$$f(i, j) = \begin{cases} 1, & I(p_i) < I(p_j) \\ 0, & I(p_i) \geq I(p_j) \end{cases}$$

Ở đây,  $p_i$  và  $p_j$  là hai điểm trong vùng lân cận của điểm đặc trưng. Kết quả là một vector nhị phân dài  $n$ , thường là  $n = 256$  hoặc  $n = 512$ .

**3. Bổ sung khả năng xoay (Orientation Assignment)** ORB bổ sung hướng xoay cho các đặc trưng bằng cách sử dụng trọng tâm cường độ sáng trong vùng lân cận:

$$\theta = \arctan2 \left( \sum_p y \cdot I(p), \sum_p x \cdot I(p) \right)$$

BRIEF sau đó được xoay theo góc  $\theta$  để làm cho đặc trưng bền vững với sự xoay.

### 2.1.2 SIFT (Scale-Invariant Feature Transform)

SIFT là một thuật toán mạnh mẽ, cung cấp khả năng phát hiện và mô tả đặc trưng bền vững với biến đổi tỷ lệ, xoay và ánh sáng.

**1. Phát hiện đặc trưng với DoG (Difference of Gaussian)** SIFT sử dụng DoG để xác định các điểm cực trị trong không gian đa tỷ lệ:

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

Trong đó:

- $L(x, y, \sigma)$  là ảnh được làm mờ bằng Gaussian:

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

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

- $k$  là tỉ lệ thay đổi giữa các mức trong không gian tỷ lệ.

Các điểm cực trị được xác định bằng cách so sánh từng pixel với các lân cận trong cùng mức và hai mức kề.

**2. Xác định hướng xoay (Orientation Assignment)** Hướng xoay được xác định bằng cách tính histogram của gradient cường độ trong vùng lân cận của mỗi điểm đặc trưng:

$$m(x, y) = \sqrt{\left(\frac{\partial L}{\partial x}\right)^2 + \left(\frac{\partial L}{\partial y}\right)^2}, \quad \theta(x, y) = \arctan\left(\frac{\frac{\partial L}{\partial y}}{\frac{\partial L}{\partial x}}\right)$$

Các góc gradient  $\theta(x, y)$  được chia thành các bin, và hướng có bin cao nhất được gán cho điểm đặc trưng.

**3. Mô tả đặc trưng với vector gradient** Vùng lân cận  $16 \times 16$  được chia thành  $4 \times 4$  ô, mỗi ô tạo một histogram gradient 8 hướng, tổng cộng  $4 \times 4 \times 8 = 128$  chiều cho vector đặc trưng.

$$\mathbf{f} = [h_1, h_2, \dots, h_{128}]$$

Vector này được chuẩn hóa để bền vững với thay đổi ánh sáng.

### 2.1.3 So sánh giữa ORB và SIFT

Thuật toán	Tốc độ	Độ chính xác
ORB	Nhanh, phù hợp với thời gian thực	Bền vững với xoay, nhưng độ chính xác thấp hơn
SIFT	Chậm, yêu cầu tính toán cao	Bền vững với xoay, tỷ lệ, ánh sáng, và biến dạng

Bảng 1: So sánh giữa ORB và SIFT.

## 2.2 So Khớp Đặc Trưng

Sau khi phát hiện và mô tả đặc trưng, bước tiếp theo là so khớp các đặc trưng giữa hai ảnh để tìm các điểm tương đồng. So khớp đặc trưng là bước quan trọng để xác định mối quan hệ hình học giữa hai ảnh và được thực hiện dựa trên các vector đặc trưng được trích xuất trong bước trước.

### 2.2.1 Phương pháp So Khớp

Có nhiều phương pháp để so khớp đặc trưng, tùy thuộc vào loại vector đặc trưng (nhị phân hoặc số thực). Trong bài toán này, hai phương pháp phổ biến được sử dụng là:

**1. Brute-Force Matching** Phương pháp này so sánh tất cả các vector đặc trưng từ ảnh thứ nhất với tất cả các vector đặc trưng từ ảnh thứ hai để tìm cặp có khoảng cách nhỏ nhất.

- Nếu vector đặc trưng là nhị phân (như trong ORB), khoảng cách được tính bằng **Hamming distance**:

$$d(\mathbf{d}_i, \mathbf{d}_j) = \sum_{k=1}^n |\mathbf{d}_{i,k} \oplus \mathbf{d}_{j,k}|$$

Trong đó:

- $\oplus$ : Phép XOR giữa hai bit.
- $n$ : Độ dài của vector đặc trưng.

- Nếu vector đặc trưng là số thực (như trong SIFT), khoảng cách được tính bằng **Euclidean distance**:

$$d(\mathbf{d}_i, \mathbf{d}_j) = \sqrt{\sum_{k=1}^n (\mathbf{d}_{i,k} - \mathbf{d}_{j,k})^2}$$

Các cặp có khoảng cách nhỏ nhất được coi là các khớp tốt nhất.

**2. k-Nearest Neighbors (k-NN) Matching** Trong phương pháp này, với mỗi vector đặc trưng trong ảnh thứ nhất, tìm  $k$  vector gần nhất trong ảnh thứ hai. Thông thường,  $k = 2$  được sử dụng để tăng độ tin cậy trong việc so khớp.

### 2.2.2 Lọc So Khớp với Lowe's Ratio Test

Lowe's Ratio Test được áp dụng để loại bỏ các so khớp sai bằng cách so sánh khoảng cách của hai cặp gần nhất:

$$\text{Ratio} = \frac{d_1}{d_2}$$

Trong đó:

- $d_1$ : Khoảng cách đến cặp so khớp tốt nhất.
- $d_2$ : Khoảng cách đến cặp so khớp tốt thứ hai.

Một so khớp được chấp nhận nếu:

$$\text{Ratio} < \text{Threshold}$$

Với ngưỡng thường được chọn là 0.75. Điều này đảm bảo rằng các điểm so khớp có độ tin cậy cao và không bị ảnh hưởng bởi nhiễu.

### 2.2.3 Thách Thức trong So Khớp Đặc Trưng

Mặc dù phương pháp so khớp đặc trưng rất mạnh mẽ, nó vẫn gặp một số thách thức:

- Số lượng lớn đặc trưng: Quá nhiều điểm đặc trưng có thể làm tăng thời gian tính toán.
- Nhiều và ánh sáng: Điểm đặc trưng có thể bị sai lệch do nhiều hoặc thay đổi ánh sáng.
- Phép chiếu hình học: Những khác biệt lớn về phối cảnh giữa hai ảnh có thể làm giảm độ chính xác.

Để khắc phục, các thuật toán như RANSAC (được sử dụng trong bước tiếp theo) giúp cải thiện độ chính xác bằng cách loại bỏ các điểm so khớp sai.

## 2.3 Ước Lượng Homography

Sau khi tìm được các điểm tương đồng giữa hai ảnh, bước tiếp theo là ước lượng ma trận homography  $H$ . Ma trận  $H$  mô tả phép biến đổi hình học giữa hai mặt phẳng, giúp căn chỉnh các điểm tương đồng trong hai ảnh.

### 2.3.1 Khái niệm Homography

Homography là một phép chiếu phối cảnh, mô tả mối quan hệ giữa các điểm trong hai ảnh khi chúng thuộc cùng một mặt phẳng. Một điểm  $\mathbf{x} = [x, y, 1]^T$  trong ảnh gốc sẽ được ánh xạ tới điểm  $\mathbf{x}' = [x', y', 1]^T$  trong ảnh đích thông qua:

$$\mathbf{x}' = H \cdot \mathbf{x}$$

Trong đó:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Ma trận  $H$  có 8 bậc tự do (1 phần tử được chuẩn hóa).

### 2.3.2 Phương trình Homography

Fương trình homography được mở rộng từ:

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

Phân rã thành các phương trình tuyến tính:

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

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

Nhân chéo để loại bỏ mẫu số, ta có hai phương trình tuyến tính:

$$h_{11}x + h_{12}y + h_{13} - h_{31}x \cdot x' - h_{32}y \cdot x' - h_{33}x' = 0$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}x \cdot y' - h_{32}y \cdot y' - h_{33}y' = 0$$

### 2.3.3 Ước Lượng Ma Trận Homography

Từ  $n$  cặp điểm tương đồng  $(\mathbf{x}_i, \mathbf{x}'_i)$ , ta xây dựng hệ phương trình:

$$A \cdot \mathbf{h} = 0$$

Trong đó:

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny'_n & -y_ny'_n & -y'_n \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}$$

Để giải  $\mathbf{h}$ , ta sử dụng phương pháp Phân rã Giá trị Kỳ dị (SVD). Ma trận  $\mathbf{h}$  là vector riêng tương ứng với giá trị riêng nhỏ nhất của  $A^T A$ .

### 2.3.4 RANSAC để Loại Bỏ Nhiễu

Dữ liệu thực tế thường chứa nhiễu và các điểm so khớp sai (outlier). Thuật toán RANSAC (Random Sample Consensus) được sử dụng để ước lượng  $H$  một cách bền vững:

1. Chọn ngẫu nhiên 4 cặp điểm tương đồng.
2. Tính ma trận homography  $H$  dựa trên 4 cặp điểm này.
3. Kiểm tra số lượng inlier bằng cách tính lỗi tái chiếu (reprojection error):

$$e = \|\mathbf{x}' - H \cdot \mathbf{x}\|$$

Điểm được coi là inlier nếu  $e$  nhỏ hơn một ngưỡng  $\epsilon$ .

4. Lặp lại quá trình nhiều lần và chọn  $H$  với số inlier lớn nhất.

### 2.3.5 Thách Thức và Giải Pháp

- **Thách thức:** Nếu số lượng điểm tương đồng ít hoặc chứa nhiều nhiễu, việc ước lượng homography sẽ không chính xác.
- **Giải pháp:** Sử dụng RANSAC để loại bỏ nhiễu và tăng số lượng điểm đặc trưng thông qua các thuật toán mạnh như SIFT.

## 2.4 Biến Đổi Hình Ảnh

Biến đổi hình ảnh là bước quan trọng trong quy trình ghép ảnh toàn cảnh, sử dụng ma trận homography  $H$  đã được ước lượng để ánh xạ một ảnh từ hệ tọa độ của nó sang hệ tọa độ của ảnh tham chiếu.

### 2.4.1 Khái niệm Biến Đổi Hình Ảnh

Phép biến đổi hình học là quá trình ánh xạ mỗi điểm  $\mathbf{x} = [x, y, 1]^T$  trong ảnh gốc sang điểm  $\mathbf{x}' = [x', y', 1]^T$  trong ảnh đích bằng cách sử dụng ma trận homography  $H$ :

$$\mathbf{x}' = H \cdot \mathbf{x}$$

Trong đó:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Điểm  $\mathbf{x}'$  được biểu diễn dưới dạng tọa độ thuần túy bằng cách chuẩn hóa giá trị  $w'$ :

$$x' = \frac{x'}{w'}, \quad y' = \frac{y'}{w'}$$

với:

$$w' = h_{31}x + h_{32}y + h_{33}$$

#### 2.4.2 Tính Toán Kích Thước Ảnh Kết Quả

Để đảm bảo ảnh kết quả có thể chứa toàn bộ nội dung của hai ảnh (gốc và đích), cần xác định kích thước vùng ảnh đầu ra. Điều này được thực hiện bằng cách ánh xạ các góc của ảnh gốc sang hệ tọa độ của ảnh đích:

$$\text{Top-left: } \mathbf{x}_1 = H \cdot [0, 0, 1]^T, \quad \text{Bottom-right: } \mathbf{x}_2 = H \cdot [w, h, 1]^T$$

Kích thước vùng ảnh được tính toán từ giá trị nhỏ nhất và lớn nhất của các tọa độ:

$$x_{\min}, x_{\max}, y_{\min}, y_{\max}$$

#### 2.4.3 Phép Ánh Xạ Ngược (Inverse Mapping)

Do ảnh đích thường không có sẵn nội dung ở các pixel  $\mathbf{x}'$ , cần sử dụng phương pháp ánh xạ ngược:

$$\mathbf{x} = H^{-1} \cdot \mathbf{x}'$$

Điều này đảm bảo rằng mỗi pixel trong ảnh đích có giá trị tương ứng từ ảnh gốc.

#### 2.4.4 Nội Suy Giá Trị Pixel

Vì tọa độ  $\mathbf{x}$  thường không phải số nguyên, giá trị pixel tại  $\mathbf{x}$  được nội suy từ các pixel lân cận:

- **Nội suy gần nhất (Nearest Neighbor):** Lấy giá trị pixel gần nhất với  $\mathbf{x}$ .
- **Nội suy song tuyến tính (Bilinear Interpolation):**

$$I(x, y) = (1-a)(1-b)I(x_1, y_1) + a(1-b)I(x_2, y_1) + (1-a)bI(x_1, y_2) + abI(x_2, y_2)$$

Trong đó:

- $(x_1, y_1), (x_2, y_2)$ : Các pixel lân cận.
- $a, b$ : Khoảng cách tương đối từ  $(x, y)$  đến các pixel lân cận.

#### 2.4.5 Thách Thức và Giải Pháp

- **Thách thức:**

- Sai lệch phép chiếu do homography không hoàn hảo.
- Các vùng đen (black holes) trong ảnh kết quả do thiếu thông tin đầu vào.

- **Giải pháp:**

- Sử dụng homography ước lượng chính xác với RANSAC.
- Tiền xử lý ảnh bằng cách căn chỉnh điểm đặc trưng.
- Lấp vùng đen bằng các kỹ thuật ngoại suy (extrapolation).

### 2.5 Kết Hợp Ảnh (Blending)

Kết hợp ảnh là bước cuối cùng trong quy trình ghép ảnh toàn cảnh, nơi các vùng chồng lấp giữa hai ảnh được xử lý để loại bỏ đường biên và tạo ra sự chuyển tiếp mượt mà. Quá trình này không chỉ đảm bảo tính thẩm mỹ mà còn giúp khắc phục các khác biệt về ánh sáng, màu sắc và các vấn đề gây ra bởi phép chiếu phối cảnh.

#### 2.5.1 Thách Thức trong Kết Hợp Ảnh

- **Đường biên rõ ràng:** Xuất hiện khi hai ảnh có sự khác biệt đáng kể về ánh sáng hoặc màu sắc trong vùng chồng lấp.
- **Nhiễu trong vùng chồng lấp:** Do lỗi nhỏ trong ước lượng homography hoặc parallax (góc nhìn khác nhau).
- **Sự không liên tục:** Các điểm ảnh ở vùng biên của hai ảnh không khớp hoàn toàn, dẫn đến sự thiếu mượt mà.

#### 2.5.2 Các Phương Pháp Kết Hợp Ảnh

1. **Alpha Blending** Alpha Blending là một phương pháp đơn giản nhưng hiệu quả, sử dụng trọng số để kết hợp các pixel từ hai ảnh trong vùng chồng lấp:

$$I(x, y) = \alpha \cdot I_1(x, y) + (1 - \alpha) \cdot I_2(x, y)$$

Trong đó:

- $I_1(x, y)$ : Cường độ pixel từ ảnh thứ nhất.
- $I_2(x, y)$ : Cường độ pixel từ ảnh thứ hai.
- $\alpha \in [0, 1]$ : Trọng số, thường thay đổi tuyến tính từ biên này đến biên kia của vùng chồng lấp.

**Ưu điểm:**

- Dễ thực hiện.
- Phù hợp cho các vùng chồng lấp nhỏ.

**Nhược điểm:**

- Có thể tạo ra hiện tượng mờ hoặc bóng ma (ghosting) khi hai ảnh không căn chỉnh chính xác.

**2. Feathering** Feathering là một cải tiến của Alpha Blending, trong đó trọng số  $\alpha$  được tính dựa trên khoảng cách từ mỗi pixel đến biên của vùng chồng lấp:

$$\alpha(x, y) = \frac{\text{distance to border in } I_2}{\text{total distance across overlap}}$$

Điều này giúp quá trình chuyển tiếp giữa hai ảnh trở nên tự nhiên hơn.

**3. Multi-Band Blending** Multi-Band Blending sử dụng các kỹ thuật biến đổi tần số để kết hợp ảnh:

- Phân tách mỗi ảnh thành các thành phần tần số thấp và tần số cao bằng cách sử dụng Gaussian và Laplacian pyramids.
- Kết hợp từng dải tần số từ hai ảnh bằng cách sử dụng mặt nạ trọng số.
- Tái tạo lại ảnh bằng cách gộp các thành phần tần số đã kết hợp.

**Ưu điểm:**

- Giảm thiểu sự khác biệt về ánh sáng hoặc màu sắc trong vùng chồng lấp.
- Cho kết quả mượt mà và tự nhiên hơn.

**Nhược điểm:**

- Yêu cầu nhiều tính toán hơn.
- Phức tạp hơn để triển khai.

**4. Gradient Domain Blending** Gradient Domain Blending dựa trên việc kết hợp gradient (độ dốc) của hai ảnh thay vì trực tiếp kết hợp pixel:

$$\min \sum_{x,y} \|\nabla I(x,y) - \nabla I_1(x,y)\|^2 + \|\nabla I(x,y) - \nabla I_2(x,y)\|^2$$

Kỹ thuật này đảm bảo rằng các sự chuyển tiếp về ánh sáng và màu sắc trở nên mượt mà hơn.

### 2.5.3 Thách Thức và Giải Pháp

- **Thách thức:**

- Sự khác biệt về màu sắc và ánh sáng giữa các ảnh.
- Các lỗi căn chỉnh nhỏ dẫn đến hiện tượng mờ hoặc bóng ma.

- **Giải pháp:**

- Tiền xử lý ảnh để cân bằng màu sắc và ánh sáng.
- Sử dụng các kỹ thuật nâng cao như Multi-Band Blending hoặc Gradient Domain Blending.

## 3 Hiện thực

Trong phần này, chúng ta sẽ sử dụng Python và thư viện OpenCV để hiện thực quá trình ghép ảnh toàn cảnh. Hai phương pháp được triển khai là sử dụng thuật toán ORB và SIFT. Cả hai phương pháp đều bao gồm các bước:

- Phát hiện và mô tả đặc trưng.
- So khớp đặc trưng.
- Ước lượng homography.
- Biến đổi hình ảnh.
- Kết hợp ảnh.

### 3.1 Cấu trúc Chương trình

Cấu trúc chương trình được thiết kế để nhận hai ảnh đầu vào, thực hiện ghép ảnh, và xuất ảnh toàn cảnh kết quả cùng với hình ảnh minh họa các điểm so khớp.

### 3.1.1 Phát hiện và Mô tả Đặc trưng

Để phát hiện và mô tả đặc trưng, hai phương pháp được triển khai:

- ORB: Sử dụng `cv2.ORB_create()` để phát hiện keypoints và tính toán các vector đặc trưng nhị phân.
- SIFT: Sử dụng `cv2.SIFT_create()` để phát hiện keypoints và tính toán các vector đặc trưng số thực.

Cả hai phương pháp đều chuyển ảnh đầu vào sang dạng ảnh xám để giảm nhiễu và tối ưu hóa việc phát hiện đặc trưng:

```
1 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
2 descriptor = cv2.ORB_create() # OR cv2.SIFT_create()
3 kps, features = descriptor.detectAndCompute(gray, None)
```

### 3.1.2 So Khớp Đặc Trưng

Các vector đặc trưng từ hai ảnh được so khớp để tìm các cặp điểm tương đồng:

- Với ORB, sử dụng Brute-Force Matcher và khoảng cách Hamming:

```
1 matcher = cv2.BFMatcher(cv2.NORM_HAMMING,
2     crossCheck=True)
3 matches = matcher.match(featuresA, featuresB)
```

- Với SIFT, sử dụng k-NN Matching và Lowe's Ratio Test:

```
1 matcher = cv2.BFMatcher()
2 rawMatches = matcher.knnMatch(featuresA, featuresB, 2)
3 good_matches = []
4 for m, n in rawMatches:
5     if m.distance < 0.75 * n.distance:
6         good_matches.append((m.trainIdx, m.queryIdx))
```

### 3.1.3 Ước Lượng Homography

Dựa trên các cặp điểm so khớp, ma trận homography  $H$  được ước lượng bằng thuật toán RANSAC để loại bỏ nhiễu:

```
1 ptsA = np.float32([kpsA[m.queryIdx] for m in matches])
2 ptsB = np.float32([kpsB[m.trainIdx] for m in matches])
3 H, status = cv2.findHomography(ptsA, ptsB, cv2.RANSAC, 4.0)
```

### 3.1.4 Biến Đổi Hình Ảnh

Ảnh thứ nhất được biến đổi theo homography  $H$  để căn chỉnh với ảnh thứ hai:

```
1 result = cv2.warpPerspective(imageA, H,
2                               (imageA.shape[1] +
3                                → imageB.shape[1],
3                                → imageA.shape[0]))
3 result[0:imageB.shape[0], 0:imageB.shape[1]] = imageB
```

## 4 Kết quả và Thảo luận

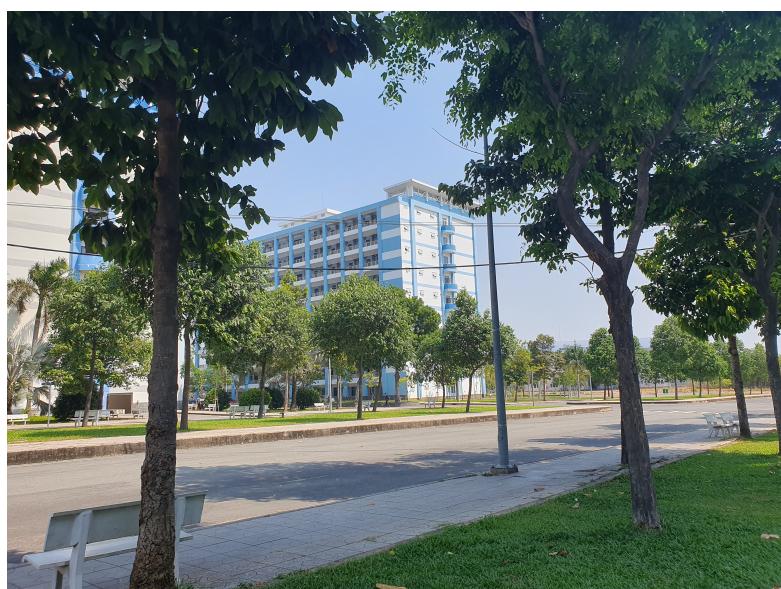
### 4.1 Ảnh nguồn



Hình 1: Ảnh nguồn mẫu 1.



Hình 2: Ảnh nguồn mẫu 2.



Hình 3: Ảnh nguồn mẫu 3.



Hình 4: Ảnh nguồn sinh viên chuẩn bị 1.



Hình 5: Ảnh nguồn sinh viên chuẩn bị 2.



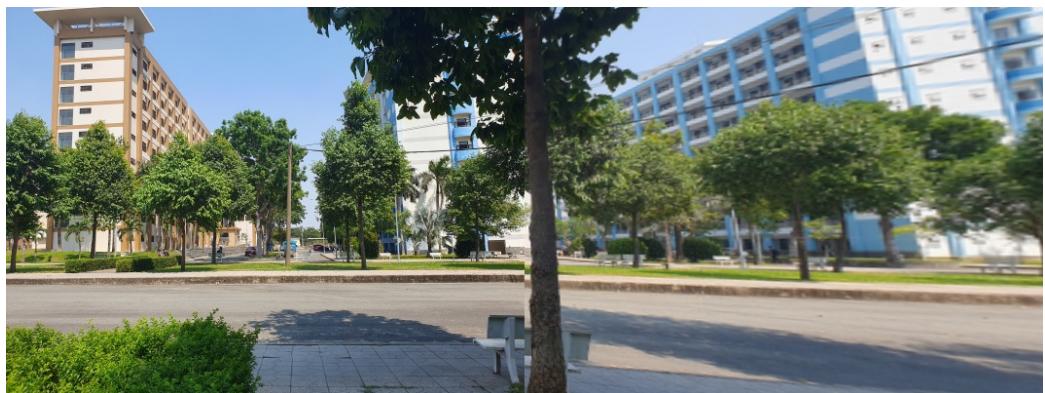
Hình 6: Ảnh nguồn sinh viên chuẩn bị 3.

## 4.2 Hình ảnh kết quả giải thuật sinh viên hiện thực

### 4.2.1 Ghép ảnh mẫu



Hình 7: Ảnh minh họa các điểm khớp 1 và 2 sử dụng giải thuật ORB.



Hình 8: Ảnh toàn cảnh 1 và 2 sử dụng giải thuật ORB.



Hình 9: Ảnh minh họa các điểm khớp 2 và 3 sử dụng giải thuật ORB.



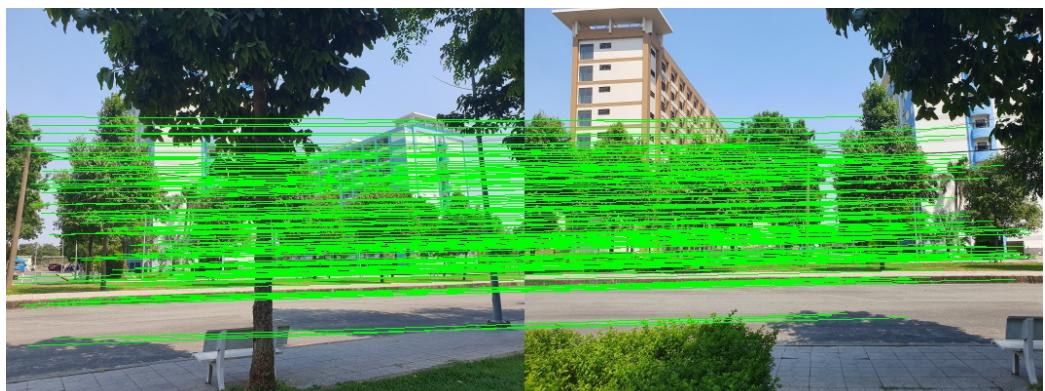
Hình 10: Ảnh toàn cảnh 2 và 3 sử dụng giải thuật ORB.



Hình 11: Ảnh minh họa các điểm khớp 1, 2 và 3 sử dụng giải thuật ORB.



Hình 12: Ảnh toàn cảnh 1, 2 và 3 sử dụng giải thuật ORB.



Hình 13: Ảnh minh họa các điểm khớp 1 và 2 sử dụng giải thuật SIFT.



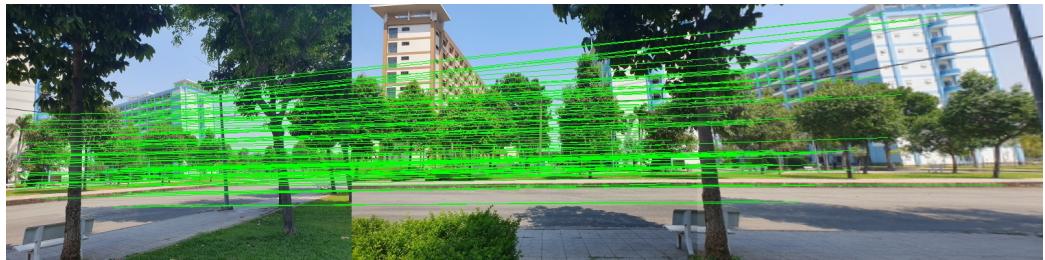
Hình 14: Ảnh toàn cảnh 1 và 2 sử dụng giải thuật SIFT.



Hình 15: Ảnh minh họa các điểm khớp 2 và 3 sử dụng giải thuật SIFT.



Hình 16: Ảnh toàn cảnh 2 và 3 sử dụng giải thuật SIFT.

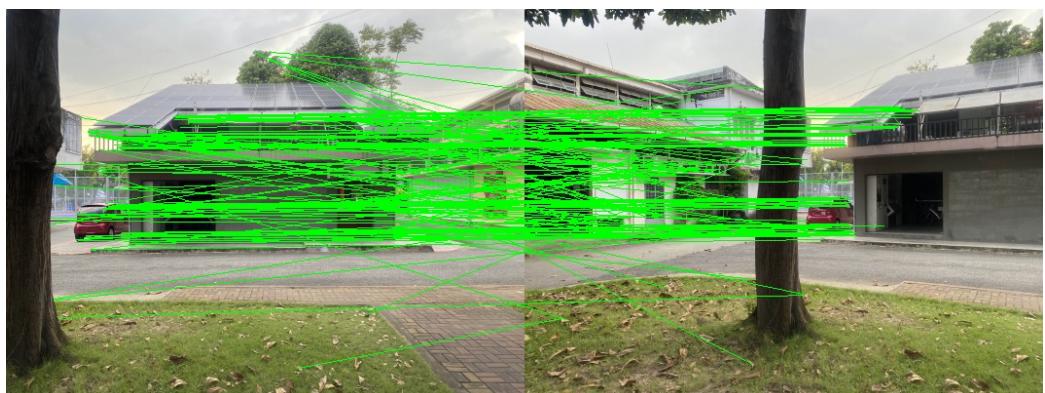


Hình 17: Ảnh minh họa các điểm khớp 1, 2 và 3 sử dụng giải thuật SIFT.



Hình 18: Ảnh toàn cảnh 1, 2 và 3 sử dụng giải thuật SIFT.

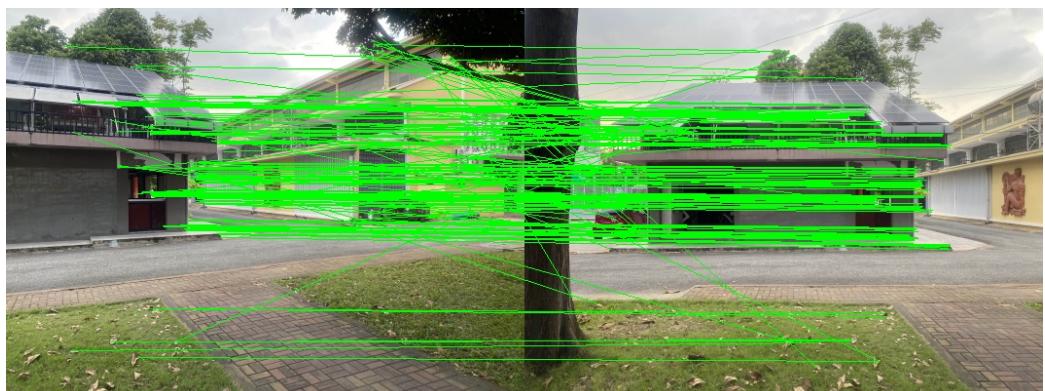
#### 4.2.2 Ghép ảnh sinh viên chuẩn bị



Hình 19: Ảnh minh họa các điểm khớp 1 và 2 sử dụng giải thuật ORB.



Hình 20: Ảnh toàn cảnh 1 và 2 sử dụng giải thuật ORB.



Hình 21: Ảnh minh họa các điểm khớp 2 và 3 sử dụng giải thuật ORB.



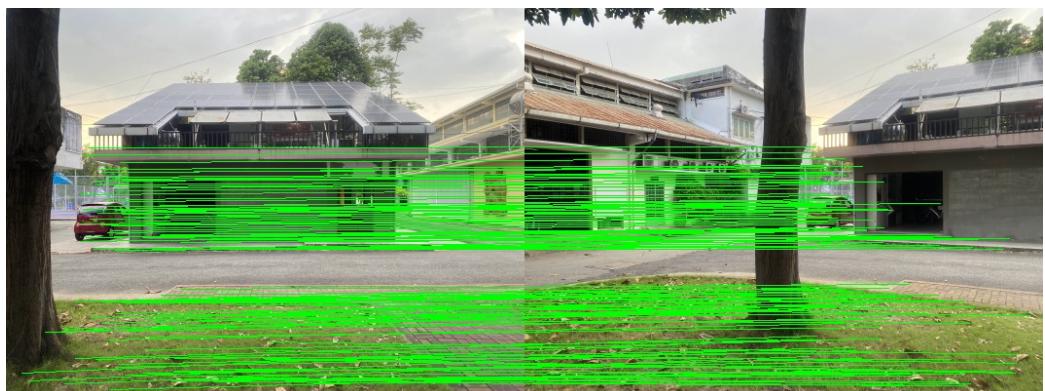
Hình 22: Ảnh toàn cảnh 2 và 3 sử dụng giải thuật ORB.



Hình 23: Ảnh minh họa các điểm khớp 1, 2 và 3 sử dụng giải thuật ORB.



Hình 24: Ảnh toàn cảnh 1, 2 và 3 sử dụng giải thuật ORB.



Hình 25: Ảnh minh họa các điểm khớp 1 và 2 sử dụng giải thuật SIFT.



Hình 26: Ảnh toàn cảnh 1 và 2 sử dụng giải thuật SIFT.



Hình 27: Ảnh minh họa các điểm khớp 2 và 3 sử dụng giải thuật SIFT.



Hình 28: Ảnh toàn cảnh 2 và 3 sử dụng giải thuật SIFT.



Hình 29: Ảnh minh họa các điểm khớp 1, 2 và 3 sử dụng giải thuật SIFT.



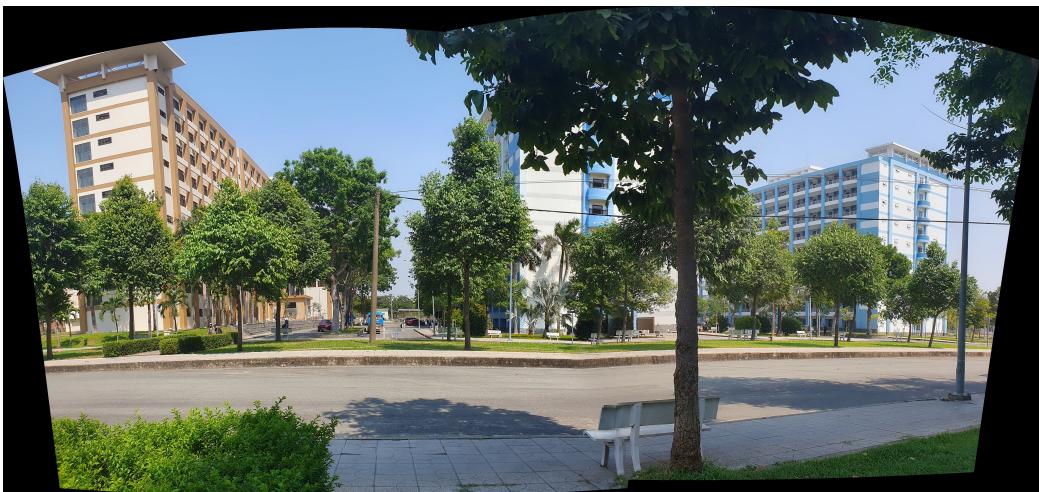
Hình 30: Ảnh toàn cảnh 1, 2 và 3 sử dụng giải thuật SIFT.

Kết quả từ ảnh mẫu cho thấy:

- **ORB:** Phát hiện được nhiều điểm khớp chính xác, đặc biệt trong các vùng có họa tiết nổi bật. Tuy nhiên, với các vùng đồng nhất, số điểm khớp giảm đáng kể, dẫn đến lỗi căn chỉnh nhỏ ở các cạnh.
- **SIFT:** Cung cấp kết quả chính xác hơn, với số lượng điểm khớp cao hơn trong các vùng chồng lấp. Các kết quả ghép ảnh toàn cảnh cho thấy sự liền mạch rõ ràng hơn so với ORB.

## 4.3 Hình ảnh kết quả gọi API của OpenCV

### 4.3.1 Ghép ảnh mău



Hình 31: Ảnh toàn cảnh 1 và 2.



Hình 32: Ảnh toàn cảnh 2 và 3.



Hình 33: Ảnh toàn cảnh 1, 2 và 3.

#### 4.3.2 Ghép ảnh sinh viên chuẩn bị



Hình 34: Ảnh toàn cảnh 1 và 2.



Hình 35: Ảnh toàn cảnh 2 và 3.



Hình 36: Ảnh toàn cảnh 1, 2 và 3.

API Stitcher của OpenCV tự động xử lý các bước phát hiện đặc trưng, so khớp và căn chỉnh ảnh. Kết quả ghép ảnh toàn cảnh rất liền mạch, ngay cả trong các trường hợp ảnh có sự khác biệt về ánh sáng hoặc chi tiết.

## 5 Code đầy đủ

Mã nguồn đầy đủ có thể truy cập tại GitHub: ComputerVisionAssignment.