ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Môn học: Xử lí ảnh số, video số

Đồ án: Khảo sát Image Stitching và ứng dụng để ghép nối các ảnh dưới các view khác nhau

Nhóm VIP

Giáo viên hướng dẫn:
Thầy PGS. TS Lý Quốc Ngọc

11/2022

Thành viên nhóm

- *20120395 – Nguyễn Anh Tuấn*
- *20120399 – Đặng Võ Hoàng Kim Tuyền*
- *20120427 – Lê Nhựt Anh*
- *20120465 – Hà Thị Hương Giang*

Nội dung

1. Giới thiệu

- 1.1 Động lực nghiên cứu
- 1.2 Phát biểu bài toán

2. Khảo sát tổng quan

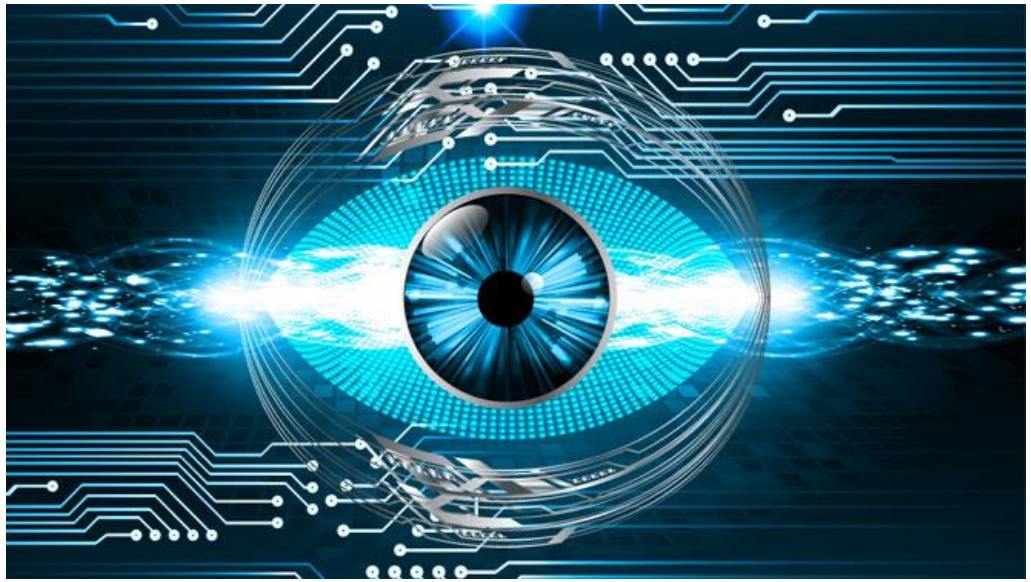
- 2.1 Giải pháp truyền thống
- 2.2 Giải pháp Deep Learning
- 2.3 Định hướng giải pháp sẽ tìm hiểu

3. Giải pháp

1. Giới thiệu

1.1. Động lực nghiên cứu

Ngày nay, cùng với sự phát triển của máy ảnh và công nghệ hiện đại, đặc biệt là lĩnh vực thị giác máy tính, việc sử dụng các hình ảnh và video với dữ liệu lớn rất phổ biến.



1. Giới thiệu

1.1. Động lực nghiên cứu

Về mặt ý nghĩa khoa học, Image Stitching giúp việc lưu trữ, truy xuất bộ sưu tập với dữ liệu lớn đạt hiệu quả cao hơn.

.

1. Giới thiệu

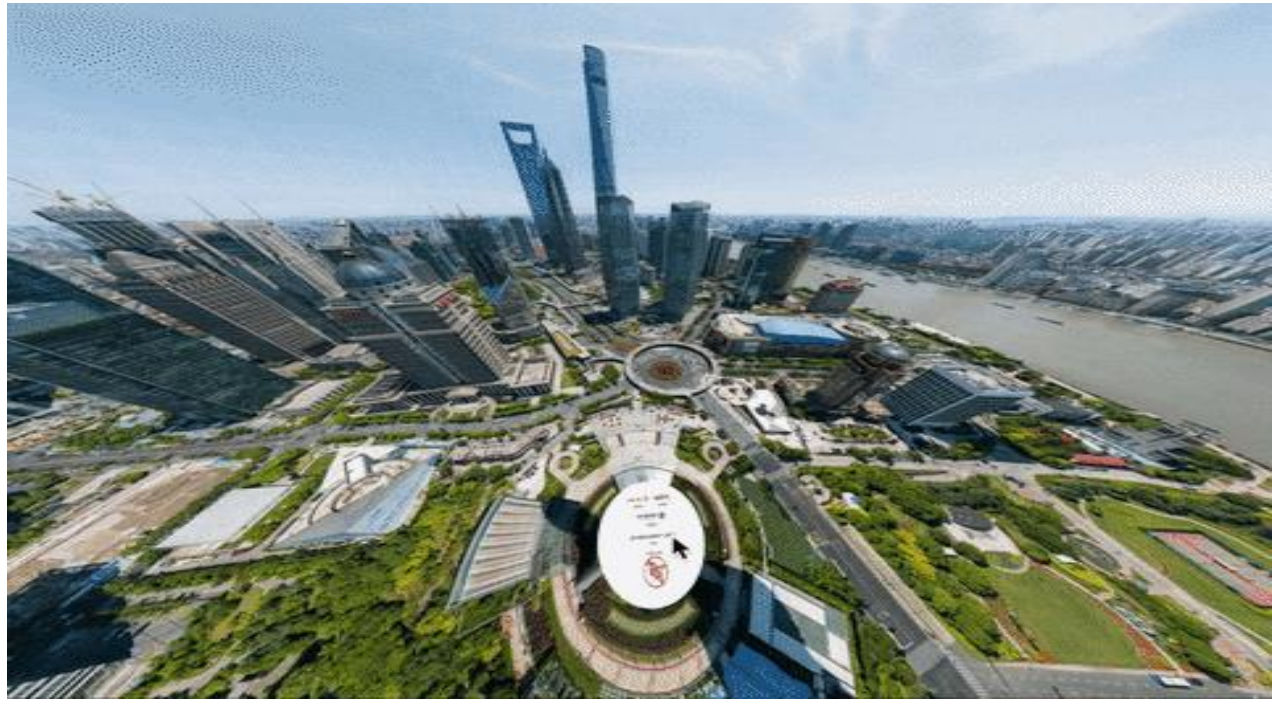
1.1. Động lực nghiên cứu

Về ý nghĩa thực tế, nó ứng dụng rộng rãi trong việc trình chiếu video trong các hội nghị, hình ảnh 3D tái tạo, nén video, chụp ảnh vệ tinh

1. Giới thiệu

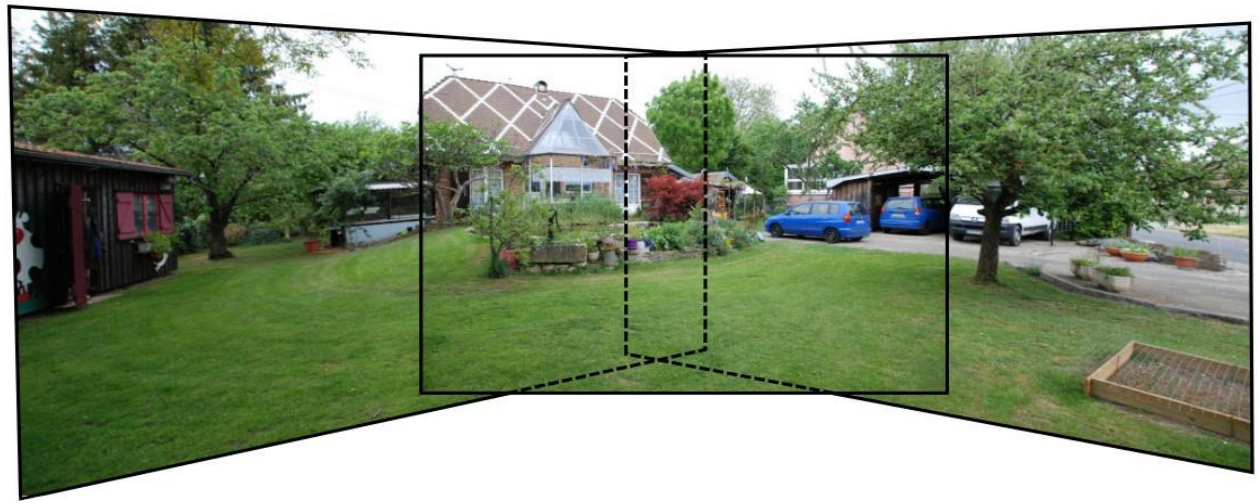
1.1. Động lực nghiên cứu

Chính những đóng góp to lớn, thiết thực ấy đã thôi thúc nhóm nghiên cứu để hiểu sâu hơn về bài toán Image Stitching này.



1. Giới thiệu

1.2. Phát biểu bài toán



1. Giới thiệu

1.2. Phát biểu bài toán

Input: Tập các ảnh có các vùng đè (overlap) lên nhau.

1. Giới thiệu

1.2. Phát biểu bài toán

Input: Tập các ảnh có các vùng đè (overlap) lên nhau.

1. Giới thiệu

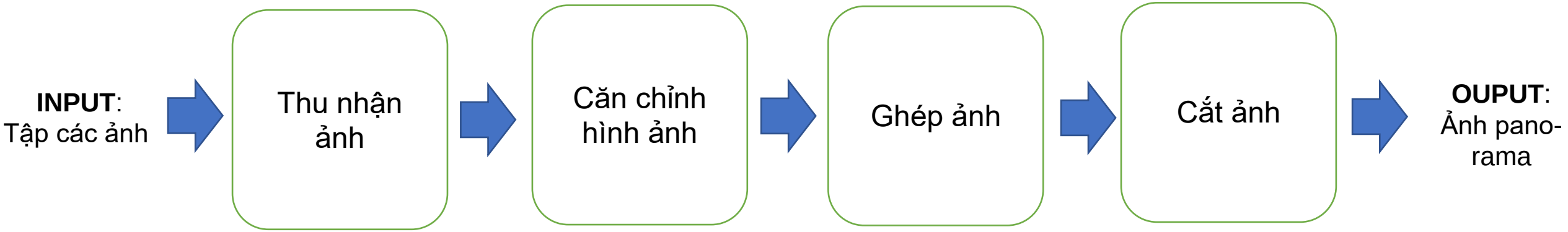
1.2. Phát biểu bài toán

Output: :Tạo ra 1 ảnh lớn (toàn cảnh) được phân đoạn hoặc hình ảnh có độ phân giải cao hơn.

1. Giới thiệu

1.2. Phát biểu bài toán

Các thao tác cần thực hiện



Sơ đồ các bước

2. Khảo sát tổng quan

2.1 Giải pháp truyền thống

- 2.1.1 Phương pháp trực tiếp (Pixel based)
- 2.1.2 Phương pháp dựa trên đặc trưng (Feature based)

2.2 Giải pháp Deep Learning

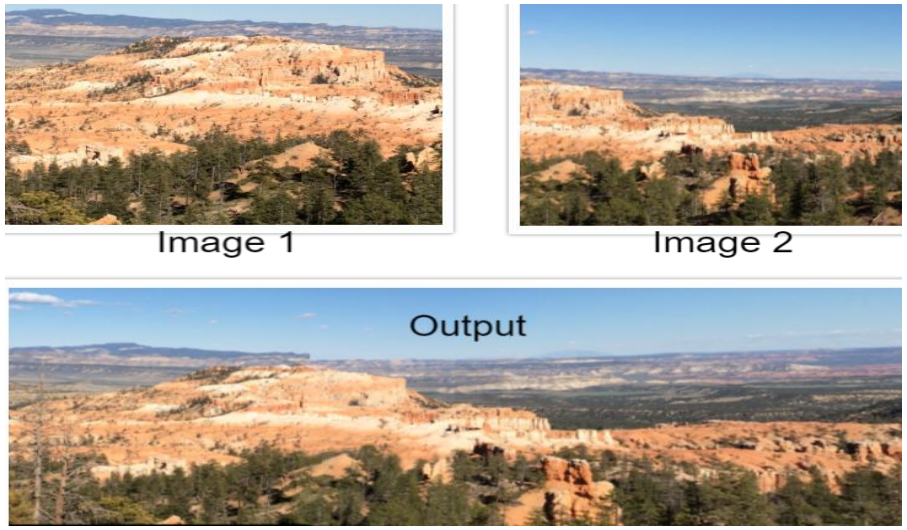
2.3 Định hướng giải pháp sẽ tìm hiểu

2. Khảo sát tổng quan

2.1. Giải pháp truyền thống

2.1.1. Phương pháp trực tiếp (Pixel based):

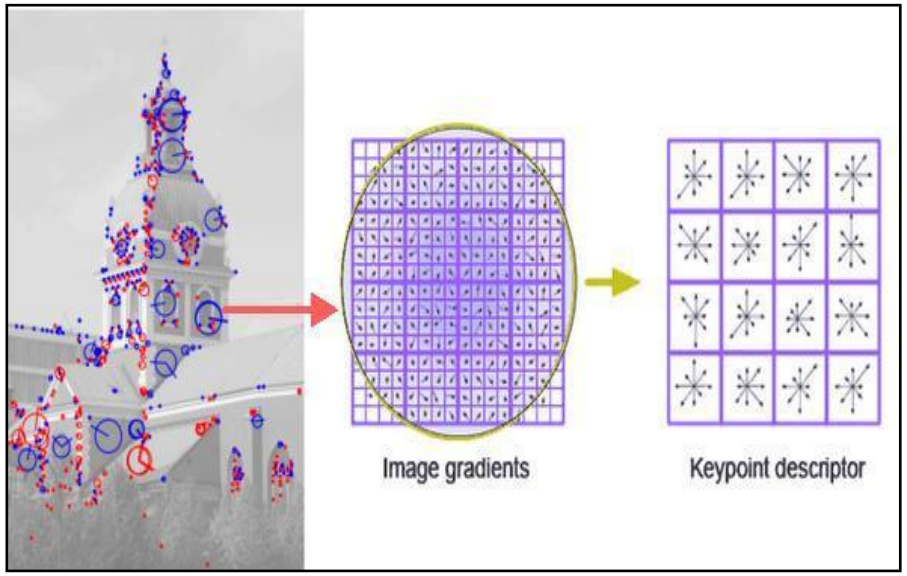
Phương pháp Pixel-based (hay còn được gọi là phương pháp trực tiếp) thực hiện dò trực tiếp từng pixel trong từng cặp ảnh từ tập ảnh đầu vào để xác định vùng chồng chéo giữa hai ảnh rồi thực hiện ghép lại tại đây tạo nên ảnh ghép.

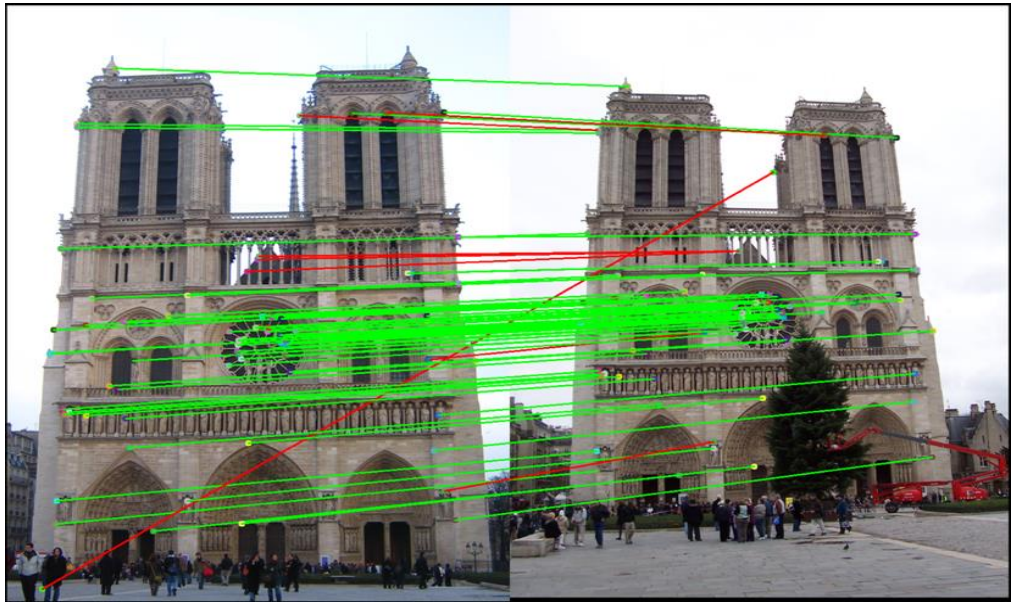


2. Khảo sát tổng quan

2.1. Giải pháp truyền thống

2.1.2. Phương pháp dựa trên đặc trưng (Feature based)





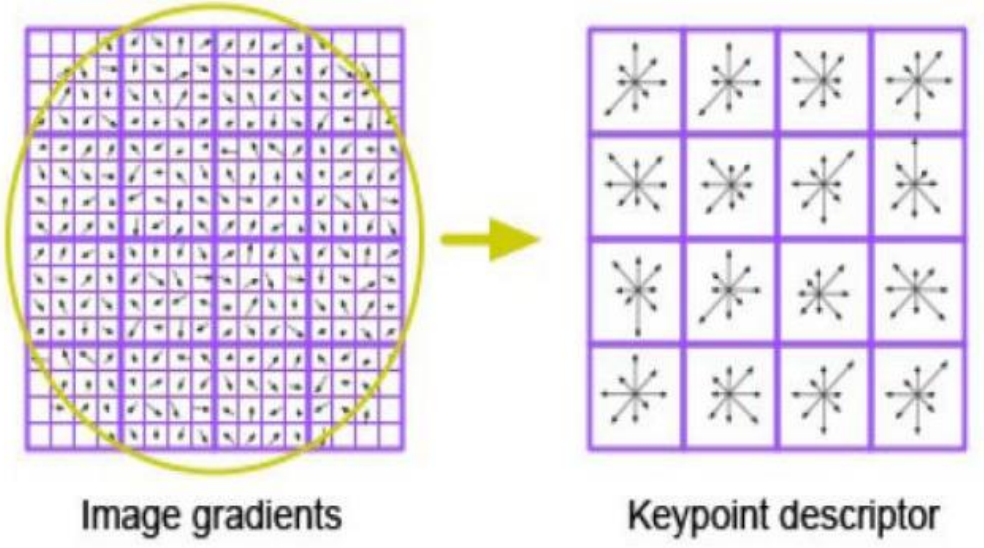
Sơ đồ các bước

2. Khảo sát tổng quan

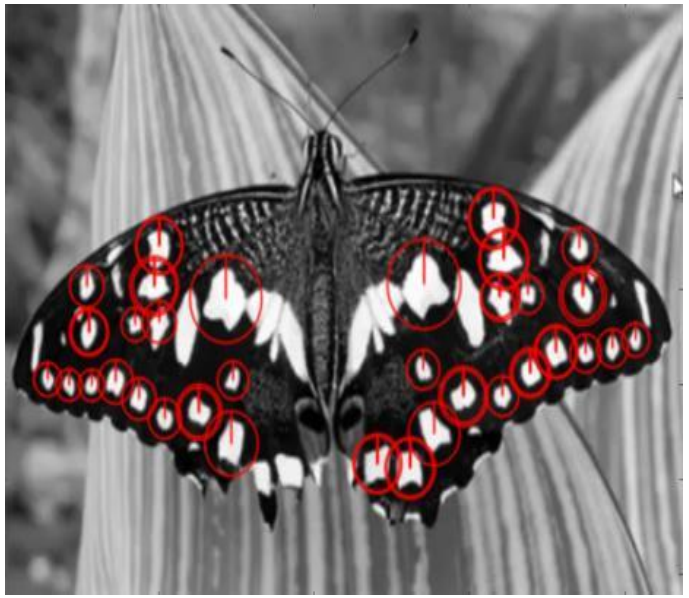
2.1. Giải pháp truyền thống

2.1.2. Phương pháp dựa trên đặc trưng (Feature based)

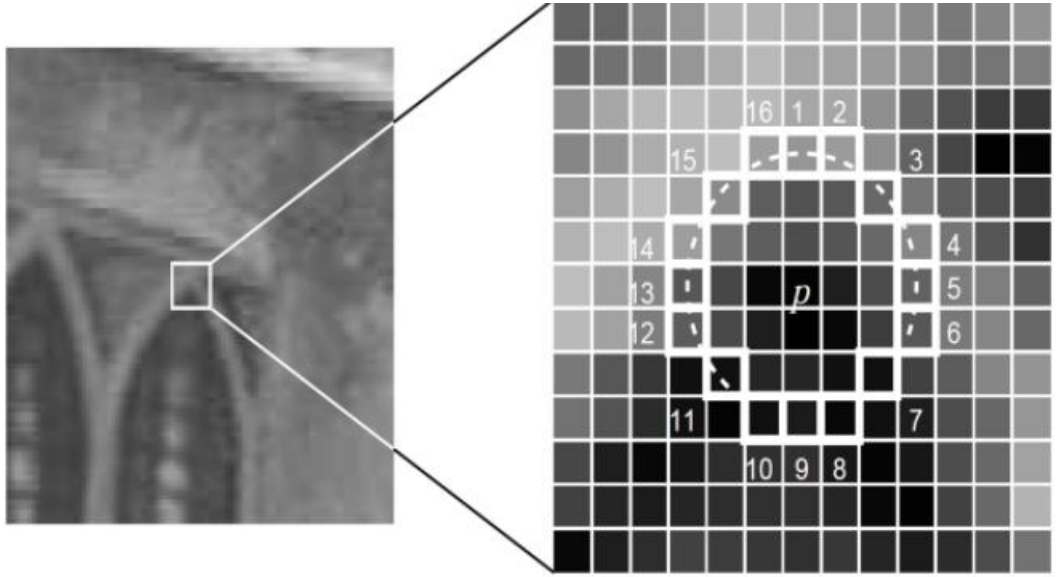
Minh họa một số
kĩ thuật trích xuất đặc trưng



Scale Invariant Feature Transform (SIFT)



Speedup Robust Feature detector (SURF)



Features from Accelerated Segment Test (FAST)

2. Khảo sát tổng quan

2.1. Giải pháp truyền thống

2.1.2. Phương pháp dựa trên đặc trưng (Feature based)

Minh họa phương pháp

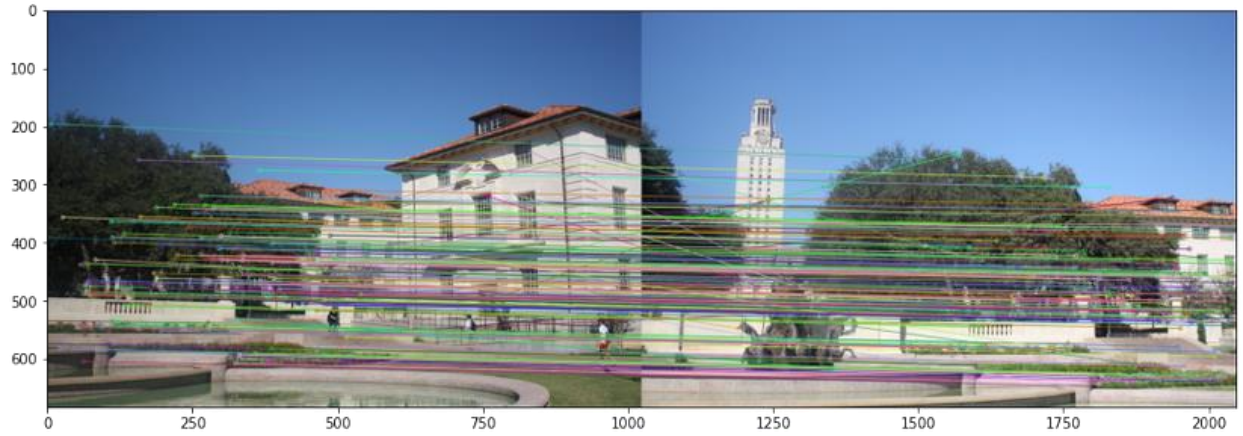
Ảnh đầu vào

2. Khảo sát tổng quan

2.1. Giải pháp truyền thống

2.1.2. Phương pháp dựa trên đặc trưng (Feature based)

Minh họa phương pháp



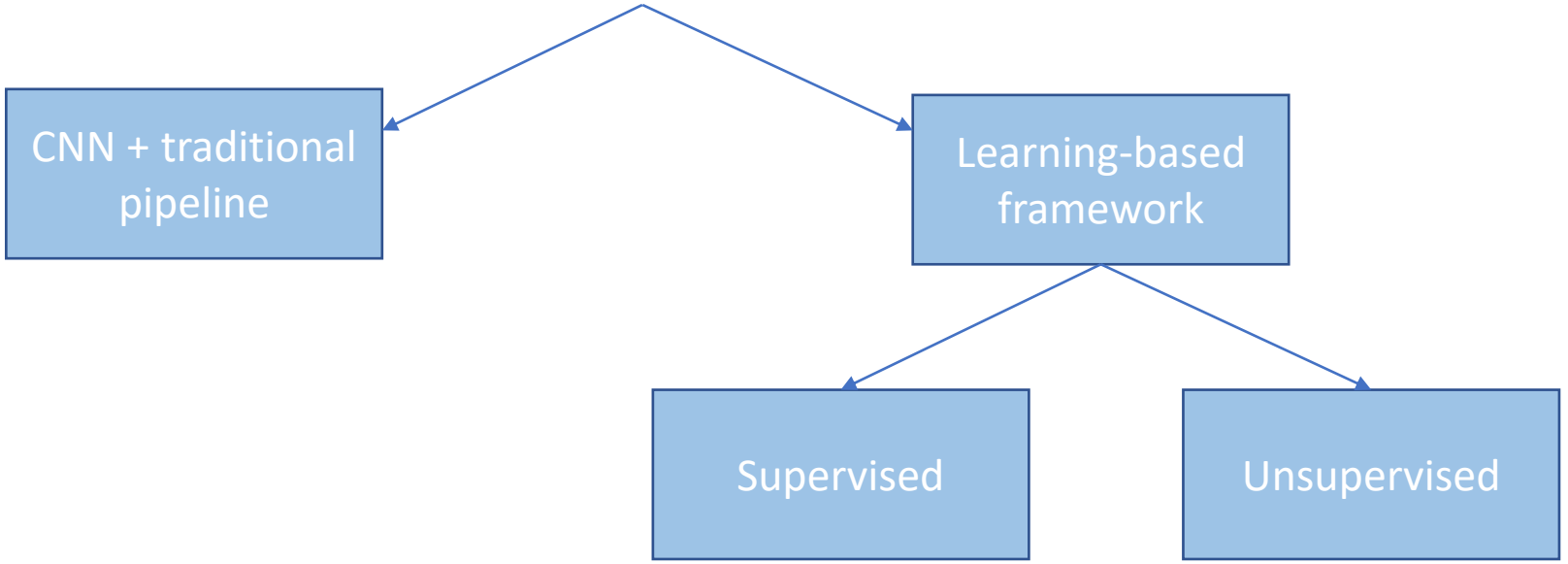
Trích xuất và so khớp các điểm đặc trưng

Ảnh ghép thu được

2. Khảo sát tổng quan

2.2. Giải pháp Deep Learning

- Dựa trên sự hiệu năng vượt trội của mô hình học sâu, sử dụng mạng nơron CNN để rút trích đặc trưng ảnh.
- Những hướng tiếp cận của CNN trong Image Stitching:



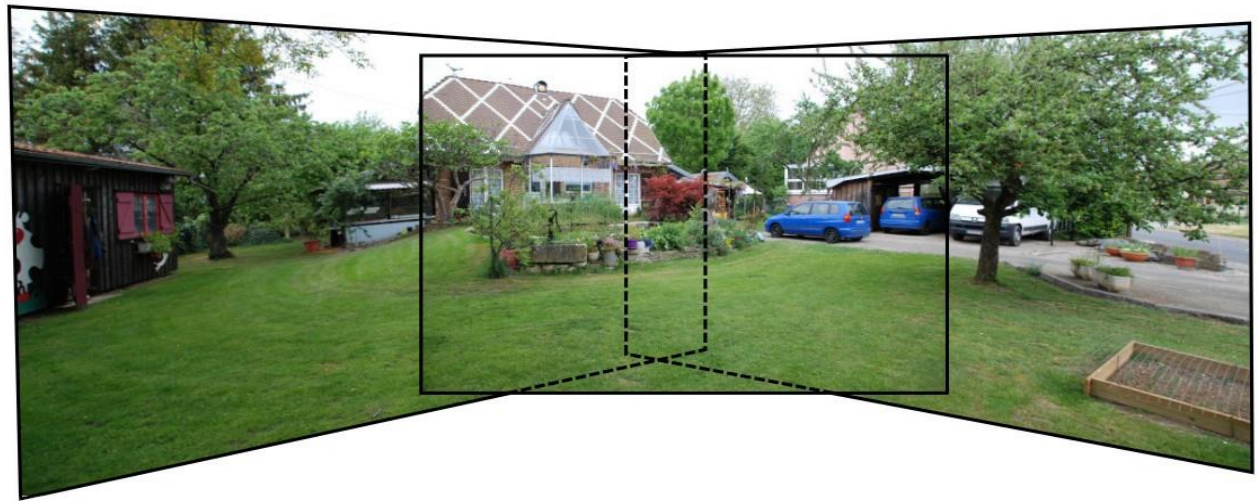
2. Khảo sát tổng quan

2.3. Định hướng giải pháp sẽ tìm hiểu

Sau khi nghiên cứu các giải pháp, nhóm nhận thấy phương pháp trực tiếp sẽ bị ảnh hưởng bởi các yếu tố xoay, thu phóng,.. nên nhóm chọn phương pháp dựa trên đặc trưng tương đồng (Feature based method) để tập trung nghiên cứu ở phần sau.

3. Giải pháp Feature based

NGUYÊN LÍ: *Từ hai ảnh* trong cùng một cảnh nhưng *ở góc nhìn khác nhau*, giải pháp sẽ thực hiện *đưa chúng về cùng một góc nhìn* bằng *cách rút trích, so khớp các đặc trưng* trong mỗi ảnh rồi tìm phép *biến đổi* phù hợp để *căn chỉnh* tạo ảnh ghép. Ta *chọn một ảnh làm ảnh cơ sở* và *tuần tự* tạo ảnh panorama theo từng cặp ảnh.



3. Giải pháp Feature based

PHƯƠNG PHÁP:

Bước 1: Tìm và so khớp các điểm đặc trưng (thuật toán SIFT)

Bước 2: Tính ma trận biến đổi đưa hai ảnh về cùng một góc nhìn (thuật toán DLT)

Bước 3: Tìm ma trận biến đổi tốt nhất (thuật toán RANSAC)

Bước 4: Hợp nhất hai ảnh về cùng một góc nhìn

CÀI ĐẶT GIẢI PHÁP

Ngôn ngữ: Python

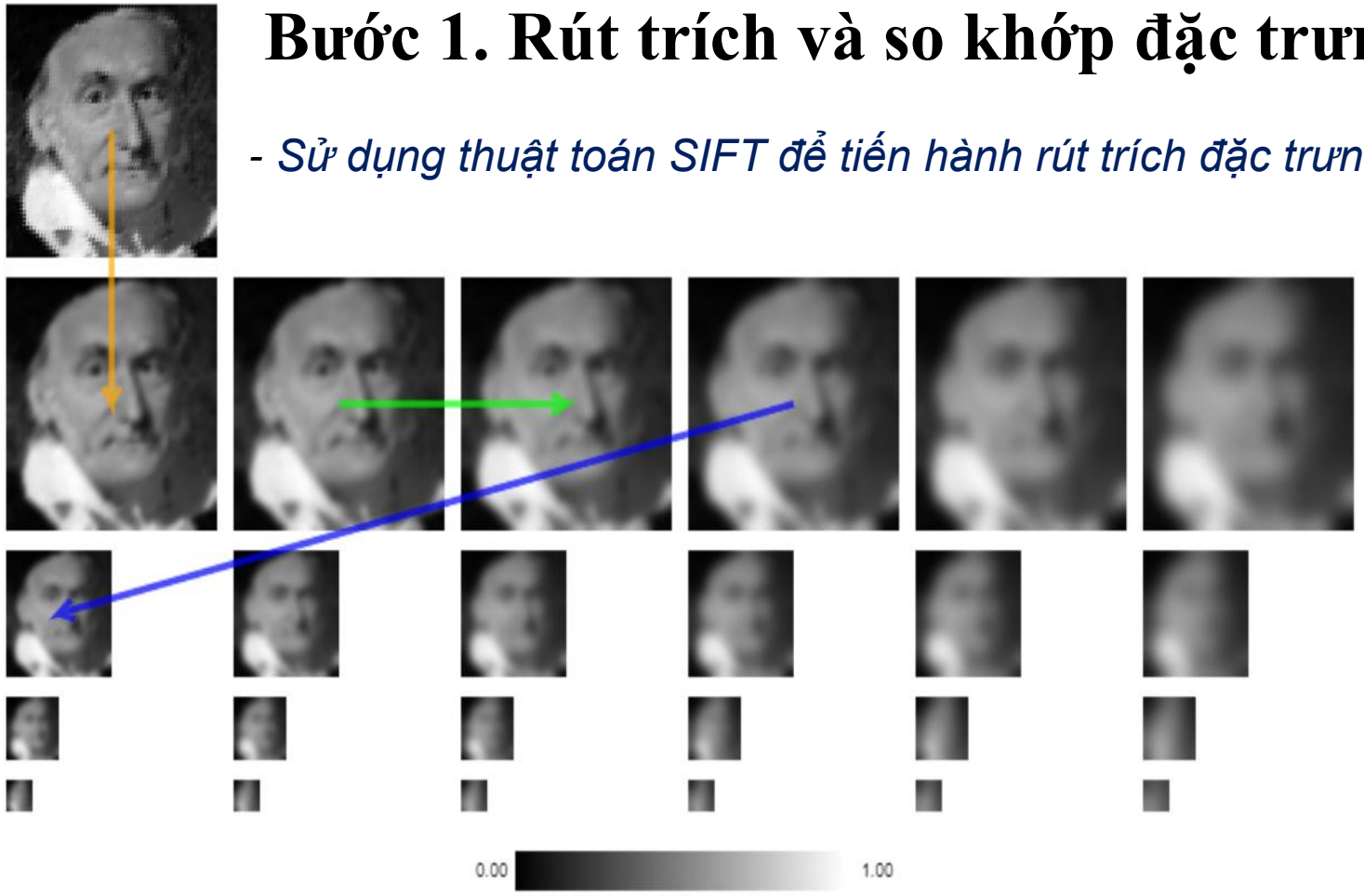
Thư viện: cv2, numpy, matplotlib.pyplot, math

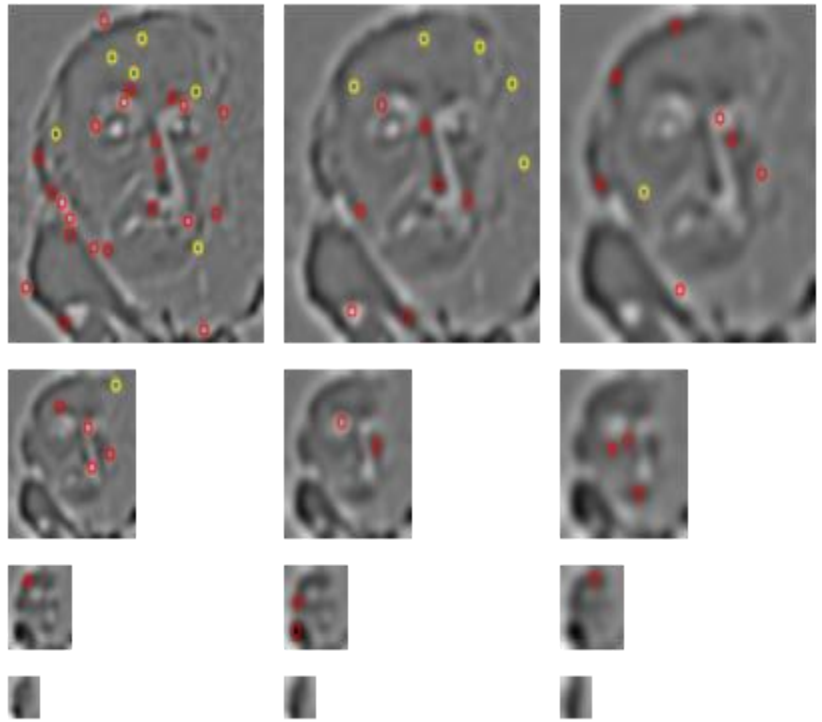
```
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from math import floor, ceil
```

3. Giải pháp Feature based

Bước 1. Rút trích và so khớp đặc trưng:

- Sử dụng thuật toán SIFT để tiến hành rút trích đặc trưng:



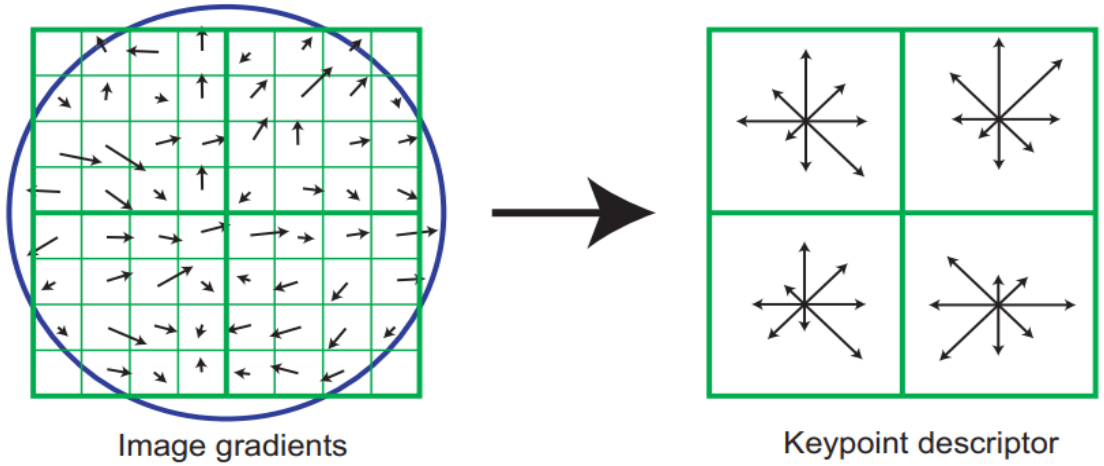


- Làm trơn ảnh với bộ lọc Gauss sử dụng các tham số σ khác nhau.

- Sử dụng hàm sai khác Difference of Gauss lọc lấy các điểm cực trị cục bộ.
- Sử dụng chuỗi Taylor mở rộng, loại bỏ các điểm có cường độ gradient nhỏ hơn giá trị ngưỡng (0.03).

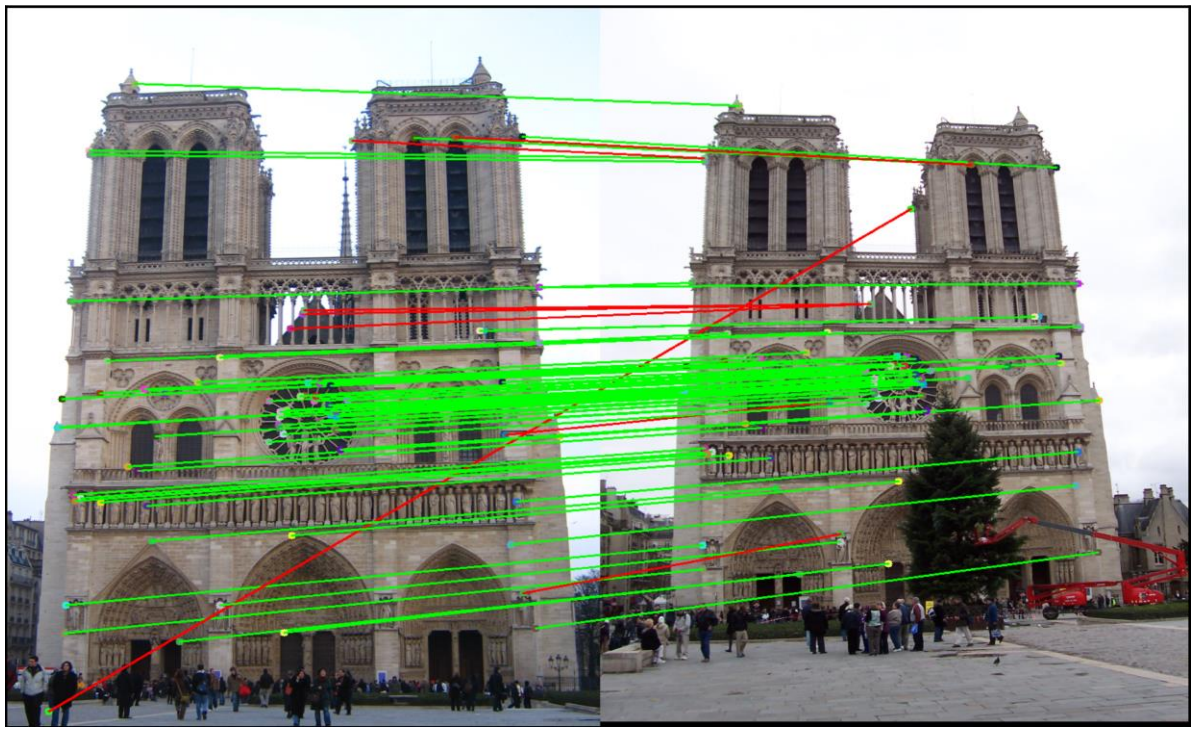
3. Giải pháp Feature based

Bước 1. Rút trích và so khớp đặc trưng:



- Gắn hướng và tạo mô tả cho các bộ điểm đặc trưng.

- So khớp các đặc trưng được rút trích ở từng cặp ảnh:



- Sử dụng lại các bộ mô tả đặc trưng ở trên và tiến hành so khớp theo phương pháp vét cạn, và hàm khoảng cách Euclid.

http://weitz.de/sift/#:~:text=The%20scale%2Dinvariant%20feature%20transform,be%20used%20for%20object%20recognition.

3. Giải pháp Feature based

Bước 1. Rút trích và so khớp đặc trưng:

- *Sử dụng các hàm được hỗ trợ bởi OpenCV để tiến hành rút trích đặc trưng:*

```python
def create_point_map(left_img, right_img):
    # Phát hiện các điểm đặc trưng bằng thuật toán SIFT
    sift = cv2.SIFT_create()
    kp1, des1 = sift.detectAndCompute(left_img, None)
    kp2, des2 = sift.detectAndCompute(right_img, None)

    # So khớp các điểm đặc trưng
    matches = cv2.BFMatcher(cv2.NORM_L2, True).match(des1, des2)
    point_map = np.array([
        [
            kp1[m.queryIdx].pt[0],
            kp1[m.queryIdx].pt[1],
            kp2[m.trainIdx].pt[0],
            kp2[m.trainIdx].pt[1]
        ]
        for m in matches])

    # Trả về cặp 2 điểm so khớp nhau trên 2 ảnh
    return point_map
```

3. Giải pháp Feature based

Bước 1. Rút trích và so khớp đặc trưng:

- *Sử dụng các hàm được hỗ trợ bởi OpenCV để tiến hành rút trích đặc trưng:*

```python
def create_point_map(left_img, right_img):
    # Phát hiện các điểm đặc trưng bằng thuật toán SIFT
    sift = cv2.SIFT_create()
    kp1, des1 = sift.detectAndCompute(left_img, None)
    kp2, des2 = sift.detectAndCompute(right_img, None)

    # So khớp các điểm đặc trưng
    matches = cv2.BFMatcher(cv2.NORM_L2, True).match(des1, des2)
    point_map = np.array([
        [
            kp1[m.queryIdx].pt[0],
            kp1[m.queryIdx].pt[1],
            kp2[m.trainIdx].pt[0],
            kp2[m.trainIdx].pt[1]
        ]
        for m in matches])

    # Trả về cặp 2 điểm so khớp nhau trên 2 ảnh
    return point_map
```

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***)

- **Input**: n cặp điểm so khớp, (n >= 4)
- **Output**: ma trận biến đổi có kích thước 3x3

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***):

B1: với mỗi cặp điểm $a_i = (x_i, y_i, w_i)^T$, $a'_i = (x'_i, y'_i, w'_i)^T$ (1<=i<=n)

tính ma trận $A_i = \begin{bmatrix} 0 & 0 & 0 & -x_i w'_i & y_i w'_i & -w_i w'_i & x_i y'_i & y_i y'_i \\ x_i w'_i & y_i w'_i & w_i w'_i & 0 & 0 & 0 & -x_i x'_i & -y_i x'_i \end{bmatrix}$

B2: Kết hợp n ma trận Ai thành một ma trân A duy nhất

$$A = \begin{bmatrix} 0 & 0 & 0 & x_1 w'_1 & y_1 w'_1 & w_1 w'_1 & -x_1 y'_1 & -y_1 y'_1 & -y'_1 \\ x_1 w'_1 & y_1 w'_1 & w_1 w'_1 & 0 & 0 & 0 & -x_1 x'_1 & -y_1 x'_1 & -x'_1 \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & x_i w'_i & y_i w'_i & w_i w'_i & -x_i y'_i & -y_i y'_i & -y'_i \\ x_i w'_i & y_i w'_i & w_i w'_i & 0 & 0 & 0 & -x_i x'_i & -y_i x'_i & -x'_i \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & x_n w'_n & y_n w'_n & w_n w'_n & -x_n y'_n & -y_n y'_n & -y'_n \\ x_n w'_n & y_n w'_n & w_n w'_n & 0 & 0 & 0 & -x_n x'_n & -y_n x'_n & -x'_n \end{bmatrix}$$

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***):

Mã nguồn B1 và B2

```
A = []
for x1, y1, x2, y2 in pairs:
    A.append([x1, y1, 1, 0, 0, 0, -x2*x1, -x2*y1, -x2])
    A.append([0, 0, 0, x1, y1, 1, -y2*x1, -y2*y1, -y2])
A = np.array(A)
```

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (**Direct Linear Transformation**):

B3: Sử dụng phương pháp SVD (Singular Value Decomposition) phân rã ma trận A thành các ma trận thành phần U, D, V

$$A = UDV^T$$

với D là ma trận chéo với giá trị được sắp giảm dần.

```
# Phân rã ma trận A bằng phương pháp SVD
U, D, V = np.linalg.svd(A)
```

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***):

B4: Lấy cột cuối cùng của ma trận V thay đổi kích thước thành ma trận H 3x3

```
H = np.reshape(V[-1], (3, 3))
```

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***):

B5: Chuẩn hóa ma trận H bằng cách chia H cho giá trị giá trị cuối.

```
# Chuẩn hóa ma trận H
H = (1 / H[2, 2]) * H
```

3. Giải pháp Feature based

Bước 2. Tính ma trận biến đổi xạ ảnh phẳng:

Thuật toán **DLT** (***Direct Linear Transformation***):

```python
def computeHomography(pairs):
    A = []
    for x1, y1, x2, y2 in pairs:
        A.append([x1, y1, 1, 0, 0, 0, -x2*x1, -x2*y1, -x2])
        A.append([0, 0, 0, x1, y1, 1, -y2*x1, -y2*y1, -y2])
    A = np.array(A)

    # Phân rã ma trận A bằng phương pháp SVD
    U, D, V = np.linalg.svd(A)

    H = np.reshape(V[-1], (3, 3))

    # Chuẩn hóa ma trận H
    H = (1 / H[2, 2]) * H

    return H
```

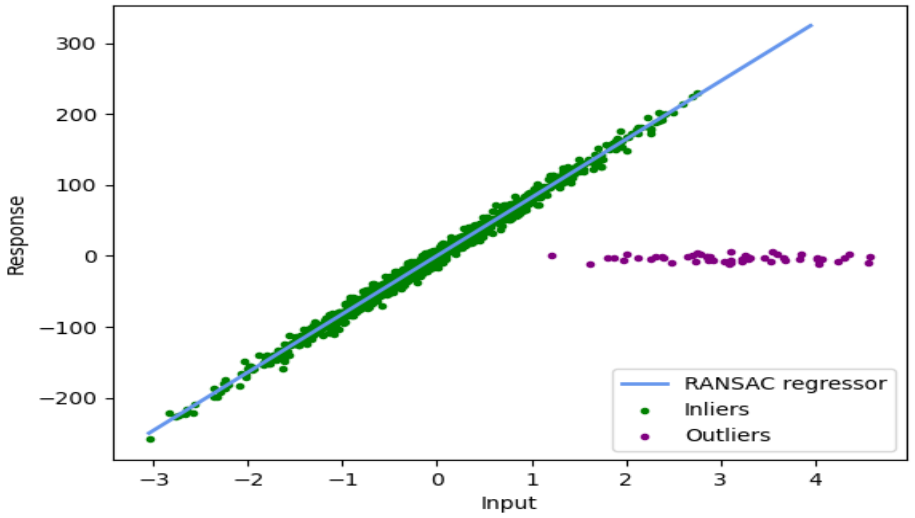
3. Giải pháp Feature based

Bước 3: Tìm ma trận biến đổi tốt nhất - Thuật toán RANSAC

- ***RANSAC*** - Random Sample Consesus

  Inlier: các điểm biến đổi tốt

  Outlier: các điểm nhiễu




```python
# Hàm loss (độ lệch giữa ước lượng biến đổi và thực tế)
def dist(pair, H):
    p1 = np.array([pair[0], pair[1], 1])
    p2 = np.array([pair[2], pair[3], 1])

    p2_estimate = np.dot(H, np.transpose(p1))
    p2_estimate = (1 / p2_estimate[2]) * p2_estimate

    return np.linalg.norm(np.transpose(p2) - p2_estimate)
```

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

3. Giải pháp Feature based

Bước 3: Tìm ma trận biến đổi tốt nhất - Thuật toán RANSAC

Lặp lại **k** lần (với giá trị k được chọn đủ lớn để đảm bảo rằng xác xuất p (thường là 0.99) của tập dữ liệu mẫu ngẫu nhiên không chứa outliers).

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

*w là tỉ lệ giữa số **inliers** trên tổng số điểm (thông thường chọn 50%).*

Các bước:

- **B1**: Chọn 4 cặp điểm tương đồng ngẫu nhiên.
- **B2**: Tính ma trận Homography $\mathbf{H_{imp}}$ từ 4 điểm.
- **B3**: Tính độ lệch **d** khi biến đổi các cặp điểm tương đồng.
- **B4**:Tính số lượng **m** các cặp điểm **inlier** thỏa mãn điều kiện $\mathbf{d_i}$**<ngưỡng.**
- **B5**: Nếu **inlier >max_inlier** thì **max_inlier = inlier** và ma trận Homography **H =** $\mathbf{H_{imp}}$.

3. Giải pháp Feature based

Bước 3: Tìm ma trận biến đổi tốt nhất - Thuật toán RANSAC

```python
def RANSAC(point_map, k, n, d):
    best_H = None
    best_inliers = []
    for i in range(k):
        # Lấy ngẫu nhiên 4 cặp điểm
        random_pairs = point_map[np.random.choice(point_map.shape[0], 4, replace=False)]

        # Tính ma trận H
        H = computeHomography(random_pairs)

        # Tính số inliers
        inliers = []
        for pair in point_map:
            if dist(pair, H) < d:
                inliers.append(pair)

        # Lưu lại ma trận H có số inliers lớn nhất
        if len(inliers) > len(best_inliers):
            best_H = H
            best_inliers = inliers

        # Dừng vòng lặp nếu số inliers lớn hơn n
        if len(best_inliers) > n:
            break

    return best_H, best_inliers
```

Hàm RANSAC:

- Input: tập điểm tương đồng, k, n, d
- Output: ma trận biến đổi, các điểm inlier

```python
final_H, inliers = RANSAC(point_map, 1200, 100, 0.5)
```

3. Giải pháp Feature based:

Bước 4: Đưa hai ảnh về chung góc nhìn và hợp nhất ảnh

Đưa hai ảnh về chung góc nhìn

- Bước cuối cùng trong việc tạo ảnh là hòa trộn hai bức ảnh lại với nhau. Ý tưởng cơ bản để thực hiện này là sử dụng một ảnh làm trung tâm, sau đó sử dụng ma trận Homography để chiếu ảnh còn lại tới mặt phẳng ảnh trung tâm.

3. Giải pháp Feature based:

Bước 4: Đưa hai ảnh về chung góc nhìn và hợp nhất ảnh

- Ảnh bên trái được sử dụng là mặt phẳng chiếu, bức ảnh bên phải là ảnh được chiếu lên mặt phẳng ảnh thứ nhất sử dụng ma trận Homography. Từ ma trận Homography đã tính tạo nên ảnh mới.

Các bước:

- Tính kích thước ảnh mới (h, w)
- Đưa ảnh về chung góc nhìn bằng ma trận biến đổi
- Tạo ảnh mặt nạ
- Hợp nhất hai ảnh bằng ảnh mặt nạ

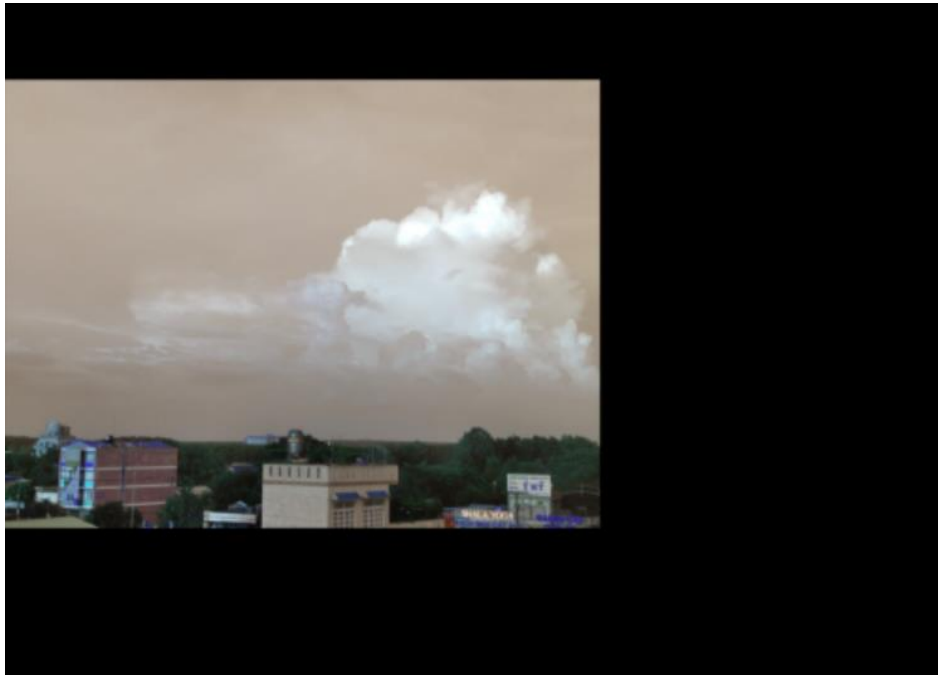
3. Giải pháp Feature based:

Bước 4: Đưa hai ảnh về chung góc nhìn và hợp nhất ảnh

Mã nguồn biến đổi ảnh về chung góc nhìn

```
warped_right_img = cv2.warpPerspective(img, np.matmul(offset, final_H),(h, w))

warped_left_img = np.zeros_like(warped_right_img)
warped_left_img[y_offset:y_offset+left_img.shape[0], x_offset:x_offset+left_img.shape[1]] = img_
```



3. Giải pháp Feature based:

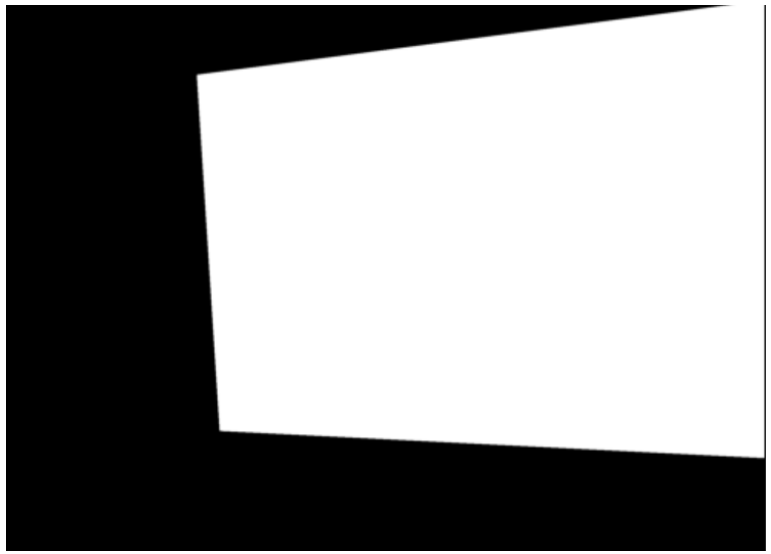
Bước 4: Đưa hai ảnh về chung góc nhìn và hợp nhất ảnh

Mã nguồn tạo ảnh mặt nạ

```
mask = np.zeros((w, h))
mask_warped_right = mask.copy()
mask_warped_left = mask.copy()

warped_right_img_gray = cv2.cvtColor(warped_right_img, cv2.COLOR_RGB2GRAY)
warped_left_img_gray = cv2.cvtColor(warped_left_img, cv2.COLOR_RGB2GRAY)

mask_warped_right = warped_right_img_gray > 0
mask_warped_left = warped_left_img_gray > 0
```



3. Giải pháp Feature based:

Bước 4: Đưa hai ảnh về chung góc nhìn và hợp nhất ảnh

Mã nguồn hợp nhất 2 ảnh bằng ảnh mặt nạ

```python
def blend(source, target, src_mask):
    final = np.zeros_like(source)

    final[src_mask] = source[src_mask]
    final[~src_mask] = target[~src_mask]

    return final

panorama = blend(warped_left_img, warped_right_img, mask_warped_left)
```

Kết quả

Ảnh đầu vào:

Kết quả

Ảnh panorama

Tài liệu tham khảo

- Image Stitching based on Feature Extraction Techniques: A Survey (8/2014)
- https://viblo.asia/p/image-stitching-thuat-toan-dang-sau-cong-nghe-anh-panorama-LzD5dee4KjY
- https://www.cc.gatech.edu
- V Megha, K K Rajkumar, "A Comparative Study on Different Image Stitching Techniques," International Journal of Engineering Trends and Technology, vol. 70, no. 4, pp. 44-58, 2022
- https://www.dropbox.com/s/amv0bsk5f90i6ai/IntroVC-CV-8-FeatureDescriptors.pptx?dl=0
- http://weitz.de/sift/#:~:text=The%20scale%2Dinvariant%20feature%20transform,be%20used%20for%20object%20recognition.


Thank
You