**UNIVERSITY OF INFORMATION TECHNOLOGY**

\_\_Computer Science\_\_



**Computer Vision**

**PANORAMA IMAGE**

Lecturer: Nguyen Vinh Tiep

Members:

* 1. Ho Ngoc Ton – 18521516
* 2. Le Truong Hoang Phuc – 18521253
* 3. Nguyen Minh Son – 18521349
* 4. Nguyen Duy Phuc – 18521255

**January 2021, Ho Chi Minh City**

**CONTENT:**

* 1. Introduction
* 2. Keypoint detection & Local invariant descriptors SIFT
* 3. Feature matching
* 4. Homography estimation using RANSAC
* 5. Perspective warping

**1. Introduction**

Previously, when the technology and application of camera are not yet developed, we create the panorama picture by use an algorithm to stitch many similar images together. That technique is also called *Image Stitching*. And currently, the panoramic photography technology is being used popularly on smart phones. These image are quite large in size, wide view. We can be created by moving the camera slowly through the photo frame to be taken. This technique helps the panorama image have a much better quality than the image created from the algorithm. In fact, the Image Stitching algorithm is still useful in some cases. For example, we want to combine multiple photos into one photo. In this paper, we will introduce the *Image Stitching* algorithm

*Image Stitching* is basically combining two or more different images to form a single panorama view. In simple terms, for input that requires a group of images, the output is a high-resolution composite image like high-quality original. In order to relate the common points of the images, the algorithm needs to identify the mathematical model that involves comparing the pixel coordinates of the two images. Previously, people followed the method "directly minimizing pixel-to-pixel dissimilarities", which meant finding overlap by matching 2 regions of 2 images with the smallest difference. However, this approach proved ineffective. Then, Feature-based solution was born to help solve the Image Stitching problem more easily.

This paper uses the feature-based image registration method and selects scale-invariant SIFT features to implement panorama image stitching. The aim of image stitching is to transform multiple source images with areas overlapping each other to unify in the same coordinate system through transformation matrixes.

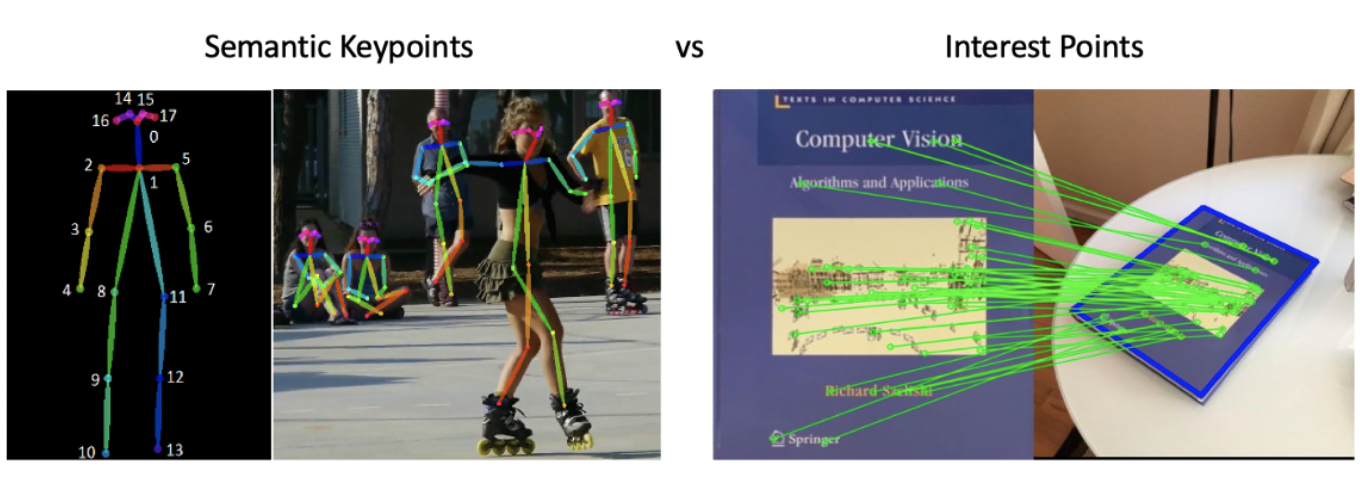
The rest of this paper is organized as follows. Section [2](https://www.hindawi.com/journals/mpe/2015/428076/#image-registration) introducted about keypoint detection and local invariant descriptors SIFT for extracting image features. Section [3](https://www.hindawi.com/journals/mpe/2015/428076/#bundle-adjustment-using-l-m-algorithm) describes the feature matching using Bruce Force Matching algorithm. Section [4](https://www.hindawi.com/journals/mpe/2015/428076/#the-traditional-stitching-method) introducted the RANSAC algorithm for purifying the matching feature points; meanwhile it obtains the transformation matrix for the matching. images describes the traditional image stitching process and Section [5](https://www.hindawi.com/journals/mpe/2015/428076/#the-improved-stitching-method)…

**2. Keypoint detection & Local invariant descriptors SIFT**

a. Keypoint detection

Keypoint or interest point detection is one important building block for many computer vision tasks, such as [SLAM](https://youtu.be/ufvPS5wJAx0?t=40) (simultaneous localization and mapping), [SfM](https://grail.cs.washington.edu/rome/) (structure from motion) and [camera calibration](https://en.wikipedia.org/wiki/Chessboard_detection#Chessboard_feature_extraction). Keypoint detection has a long history predating deep learning, and many glorious algorithms in wide industry applications (such as [FAST](https://en.wikipedia.org/wiki/Features_from_accelerated_segment_test), [SIFT](https://en.wikipedia.org/wiki/Scale-invariant_feature_transform) and [ORB](https://en.wikipedia.org/wiki/Oriented_FAST_and_rotated_BRIEF)) are based on hand-crafted features.

There are largely two types of keypoints in common use in computer vision. ***Semantic keypoints*** are points of interest with semantic meaning for objects in an image, such as the left eye corner of a face, the right shoulder of a person or the front left tire hub of a car. ***Interest points***are more low-level points that may not have clear semantic meaning, such as a corner point or ending point of a line segment.



Deep learning method has dominated state-of-the-art semantic keypoint detection. [**Mask RCNN**](https://arxiv.org/abs/1703.06870) (ICCV 2017) and [**PifPaf**](https://arxiv.org/abs/1903.06593) (CVPR 2019) are two representative methods for detecting semantic keypoints. These methods are supervised learning and require extensive and expensive human annotation. This makes them difficult to readily apply to interest point detection, as interest points are semantically ill-defined and thus a human annotator cannot reliably and repeatedly identify the same set of interest points. It is therefore impossible to formulate the task of interest point detection as a supervised learning problem.

In this paper, we will use Interest Points to find same points in group image.

b. Local invariant descriptors SIFT

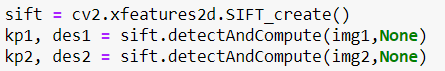
An initial and probably naive approach would be to extract key points using an algorithm such as Harris Corners. Then, we could try to match the corresponding key points based on some measure of similarity like Euclidean distance. As we know, corners have one nice property: they are invariant to rotation. It means that, once we detect a corner, if we rotate an image, that corner will still be there.

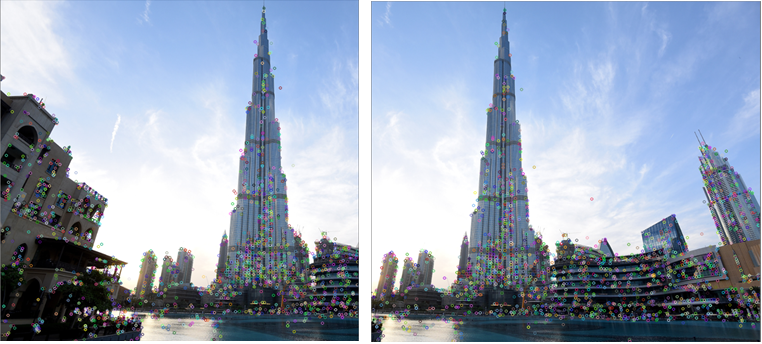
However, what if we rotate then scale an image? In this situation, we would have a hard time because corners are not invariant to scale. That is to say, if we zoom-in to an image, the previously detected corner might become a line!

In summary, we need features that are invariant to rotation and scaling. That is where more robust methods like SIFT, SURF, and ORB come in.

Methods like SIFT try to address the limitations of corner detection algorithms. Usually, corner detector algorithms use a fixed size kernel to detect regions of interest (corners) on images. It is easy to see that when we scale an image, this kernel might become too small or too big.

To address this limitation, methods like SIFT uses Difference of Gaussians (DoD). The idea is to apply DoD on differently scaled versions of the same image. It also uses the neighboring pixel information to find and refine key points and corresponding descriptors.

To start, we need to load 2 images, a query image, and a training image…



**3. Feature matching**

As we can see, we have a large number of features from both images. Now, we would like to compare the 2 sets of features and stick with the pairs that show more similarity.

With OpenCV, feature matching requires a Matcher object. Here, we explore two flavors:

• Brute Force Matcher

• KNN (k-Nearest Neighbors)

The BruteForce (BF) Matcher does exactly what its name suggests. Given 2 sets of features (from image A and image B), each feature from set A is compared against all features from set B. By default, BF Matcher computes the Euclidean distance between two points. Thus, for every feature in set A, it returns the closest feature from set B. For SIFT and SURF OpenCV recommends using Euclidean distance. For other feature extractors like ORB and BRISK, Hamming distance is suggested.

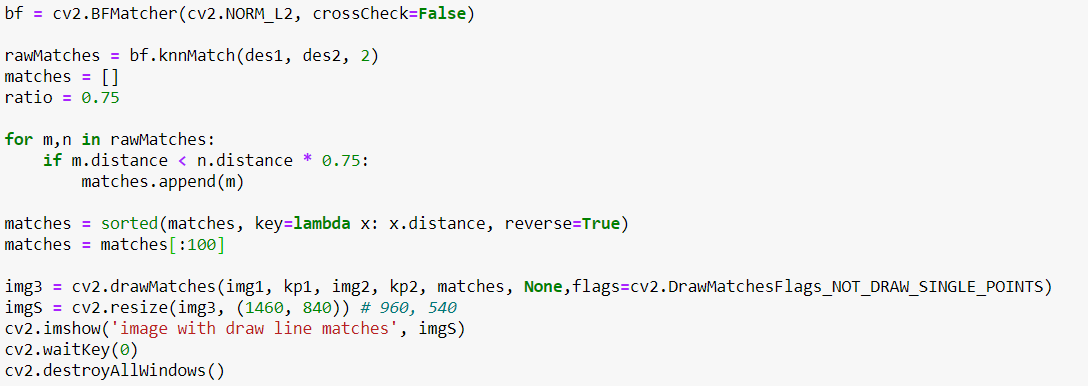
To create a BruteForce Matcher using OpenCV we only need to specify 2 parameters. The first is the distance metric. The second is the crossCheck boolean parameter.

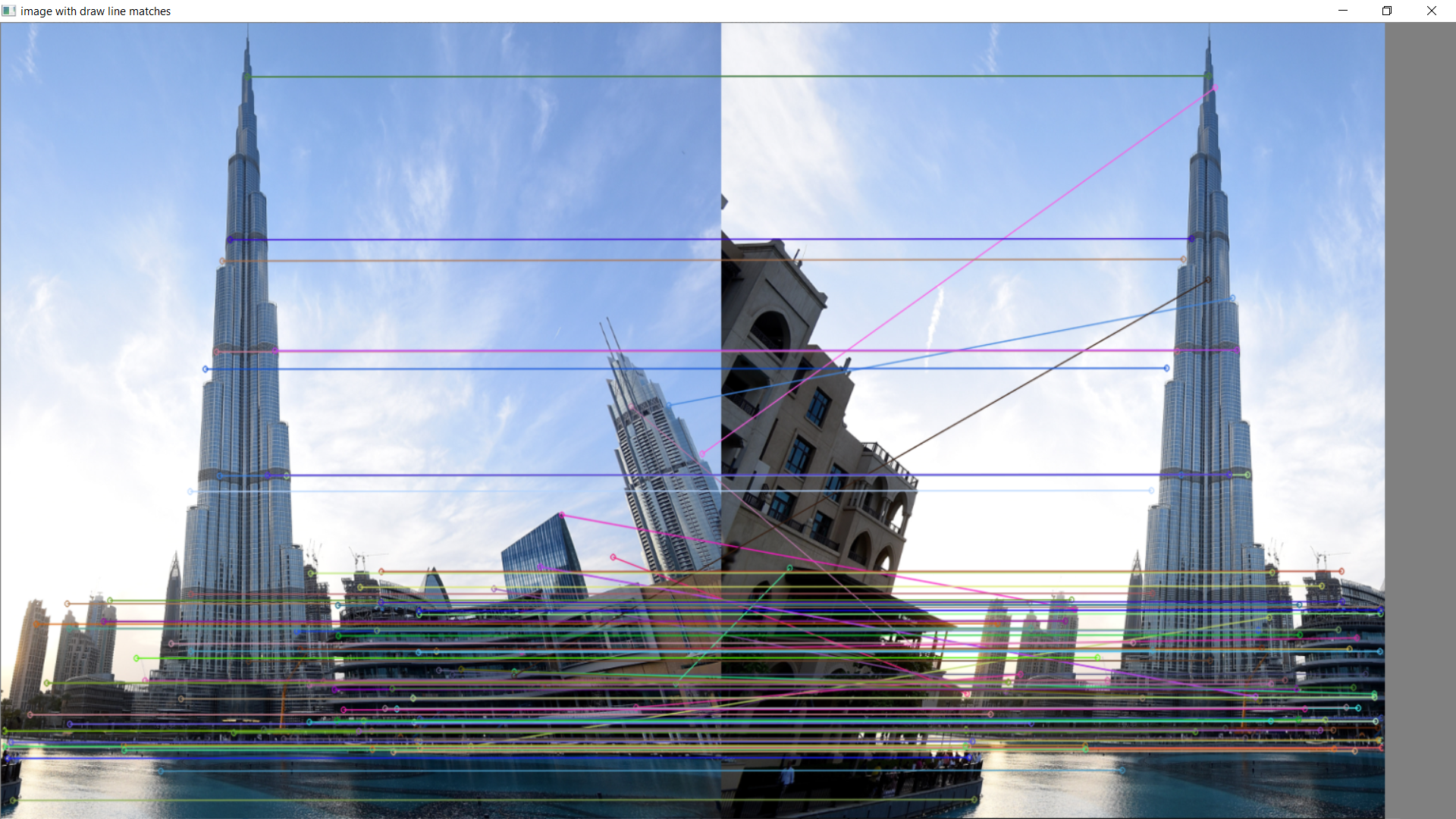
The crossCheck bool parameter indicates whether the two features have to match each other to be considered valid. In other words, for a pair of features (f1, f2) to considered valid, f1 needs to match f2 and f2 has to match f1 as the closest match as well. This procedure ensures a more robust set of matching features and is described in the original SIFT paper.

However, for cases where we want to consider more than one candidate match, we can use a KNN based matching procedure.

Instead of returning the single best match for a given feature, KNN returns the k best matches.

Note that the value of k has to be pre-defined by the user. As we expect, KNN provides a larger set of candidate features. However, we need to ensure that all these matching pairs are robust before going further.





**4. Homography estimation using RANSAC**

RANdom SAmple Consensus or RANSAC is an iterative algorithm to fit linear models. Different from other linear regressors, RANSAC is designed to be robust to outliers.

Models like Linear Regression uses least-squares estimation to fit the best model to the data. However, ordinary least squares is very sensitive to outliers. As a result, it might fail if the number of outliers is significant.

RANSAC solves this problem by estimating parameters only using a subset of inliers in the data. The figure below shows a comparison between Linear Regression and RANSAC. First, note that the dataset contains a fairly high number of outliers.

We can see that the Linear Regression model gets easily influenced by the outliers. That is because it is trying to reduce the average error. Thus, it tends to favor models that minimize the overall distance from all data points to the model itself. And that includes outliers.

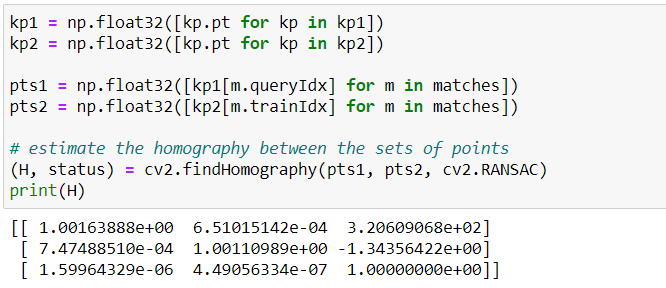
On the contrary, RANSAC only fits the model on the subset of points identified as the inliers.

This characteristic is very important to our use case. Here, we are going to use RANSAC to estimate the Homography matrix. It turns out that the Homography is very sensitive to the quality of data we pass to it. Hence, it is important to have an algorithm (RANSAC) that can filter points that clearly belong to the data distribution from the ones which do not.

Once we have the estimated Homography, we need to warp one of the images to a common plane.

Here, we are going to apply a perspective transformation to one of the images. Basically, a perspective transform may combine one or more operations like rotation, scale, translation, or shear. The idea is to transform one of the images so that both images merge as one. To do this, we can use the OpenCV warpPerspective() function. It takes an image and the homography as input. Then, it warps the source image to the destination based on the homography.

The resulting panorama image is shown below. As we see, there are a couple of artifacts in the result. More specifically, we can see some problems related to lighting conditions and edge effects at the image boundaries. Ideally, we can perform post-processing techniques to normalize the intensities like histogram matching. This would likely make the result look more realistic.



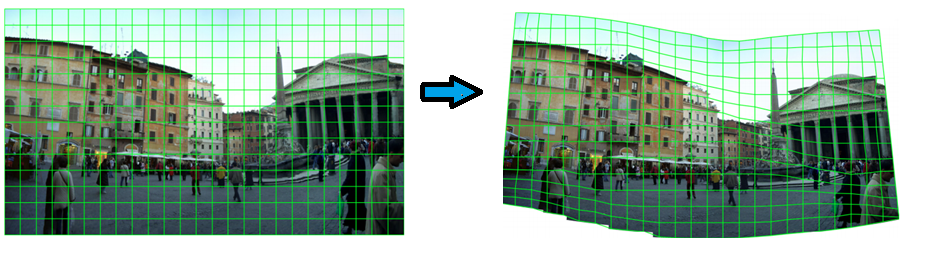
**5. Perspective warping**

The key idea is a two-step method that first locally warps the image to fit a rectangle and then globally optimizes a mesh placed on this rectangle.

In the first step, we modify the Seam Carving algorithm to expand the irregular image to a rectangle. We consider Seam Carving as a warping method that displaces all pixels on one side of each seam.



In the second step, we place a grid mesh on the rectangle image generated by Seam Carving. With the displacement field of the first step, this mesh is warped back and placed in the original irregular image.



Then we globally optimize this mesh, fitting it to a rectangle while preserving perceptual properties including shapes and straight lines. Our method is fully automatic. It is purely content-based and requires no prior knowledge about the projections.

