Digital Image Process Project Report

Image Fusion Part 1- SURF

Background

When we want to match different images, we often encounter the problem of different image scales. The distance of feature points in different images becomes different, and the object will become different sizes. If we correct the size of feature points, it will cause intensity mismatch. In order to solve this problem, a scale invariant SURF feature detection is proposed, in which the scale factor is added when calculating the feature points. SURF is similar to SIFT algorithm. SIFT algorithm is relatively stable and detects more feature points, but the complexity is high. SURF needs simple operation, high efficiency and short operation time.

Method

① Construction of Hessian matrix

Hessian matrix is defined as:

$$H(f(x,y))=\begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}.$$

Surf uses Hessian matrix to extract feature points, and a Hessian matrix can be obtained for each pixel.

The discriminant of Hessian matrix is defined as:

$$det(H)=\frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2}-\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2.$$

The value of the discriminant is the eigenvalue of the Hessian matrix. All points can be classified by using the symbol of the judgment result. The positive and negative values are taken according to the discriminant to always judge whether the point is or not the value of the pole. Because our feature points need scale independence, we need to Gaussian filter them before constructing Hessian matrix.

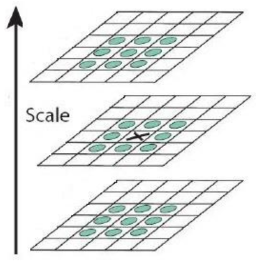
The Hessian matrix and discriminant after Gaussian filtering are

$$H(x,\sigma)=\begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$ and $$det(H_{approx})=D_{xx}D_{yy}-(0.9D_{xy})^2.$$

② Using non maximum suppression to preliminarily determine feature points and accurately locate feature points

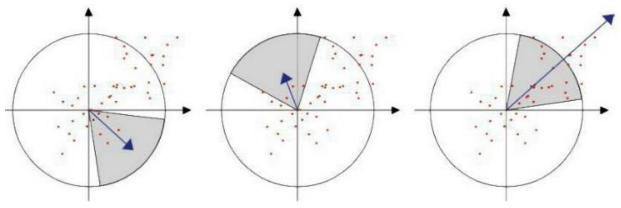
Compare the size of each pixel processed by Hessian matrix with the 26 points in its threedimensional field. If it is the maximum or minimum of the 26 points, it will be retained as a preliminary feature point. In the detection process, the filter corresponding to the image resolution of the scale layer is used for detection.

Then, the three-dimensional linear interpolation method is used to obtain sub-pixel feature points. At the same time, those points whose value is less than a certain threshold are removed, and the extreme value is increased to reduce the number of detected feature points. Finally, only a few strongest feature points will be detected. The following is based on 3×3 filter as an example:



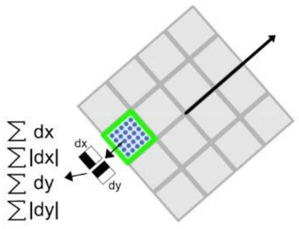
③ Determination of main direction of feature points

Calculate the sum of Haar wavelet responses of all points in the X (horizontal) and Y (vertical) directions in the 60 degrees sector of neighborhood with radius 6s (s is the scale value of the feature point). Traverse the whole circular region, and select the direction of the longest vector as the main direction of the feature point. The schematic diagram of this process is as follows:



④ Constructing surf feature point description operator

Take a square box with a side length of 20s around the feature points, and the direction is the main direction detected in the previous step. Then the frame is divided into 16 sub regions, and each sub region counts the Haar wavelet features of 25 pixels in the horizontal and vertical directions relative to the main direction. The schematic diagram of this process is as follows:



⑤ Feature point matching

According to the above steps, the description operator can be obtained. The Euclidean distance of feature point description vector is used as the similarity judgment measure of feature points in two images. Take a feature point in image 1 and find the first two feature points closest to the Euclidean distance in image 2. In these two feature points, if the nearest distance divided by the second closest distance is less than a certain scale threshold, this pair of matching points will be accepted. Adjusting this proportional threshold can deal with different matching situations.

⑥ Unifying coordinate transformation

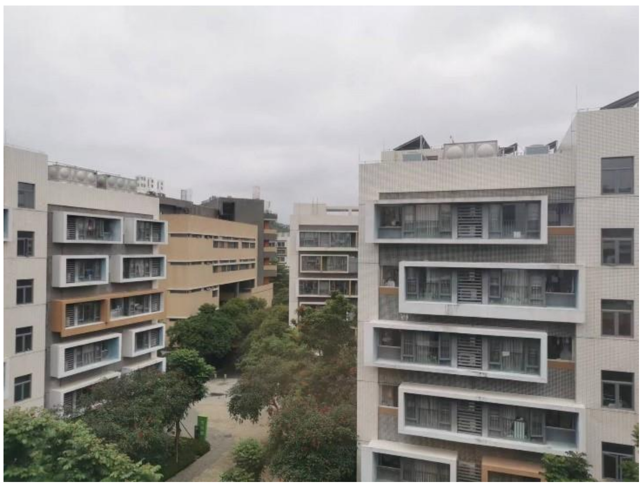
The homography matrix is calculated to map the points in the first picture to the second picture. At the same time, RANSAC algorithm is used to continue to screen the reliable matching points

⑦ Image fusion (crack removal processing)

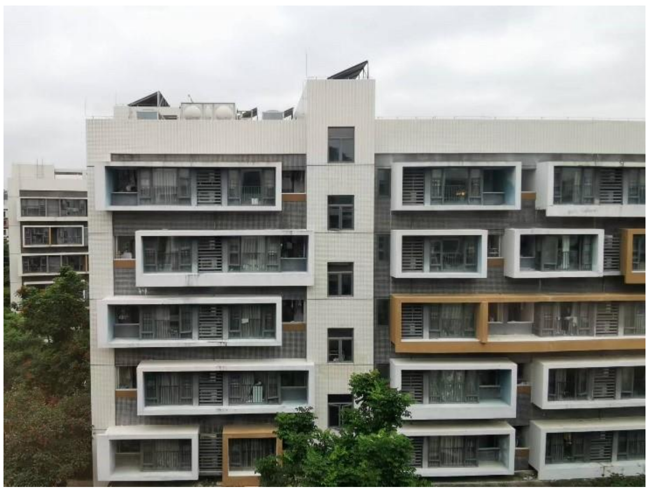
The above processed image will make the transition between the two images very bad due to the light color, so specific processing is needed to solve this unnatural problem. The processing idea here is weighted fusion, which adds the pixel values of the overlapping area of the image according to a certain weight to synthesize a new image.

Result

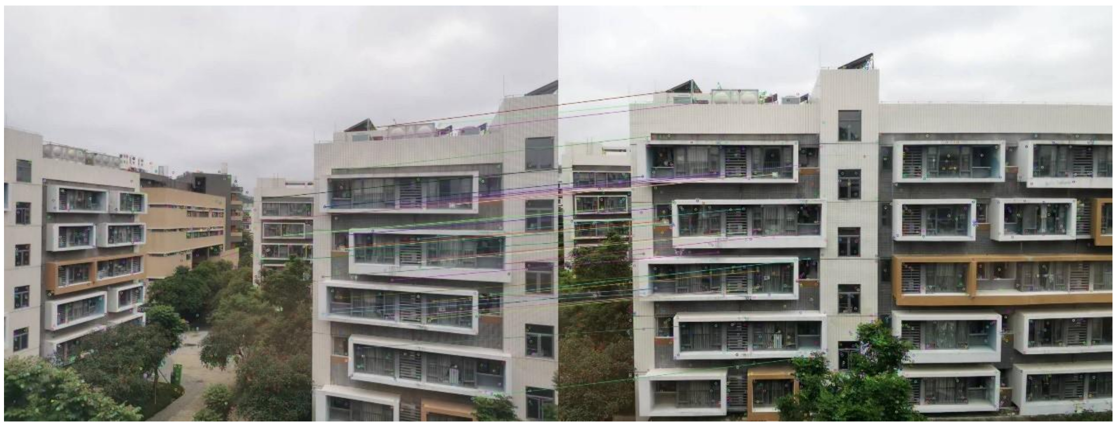
The original left image:



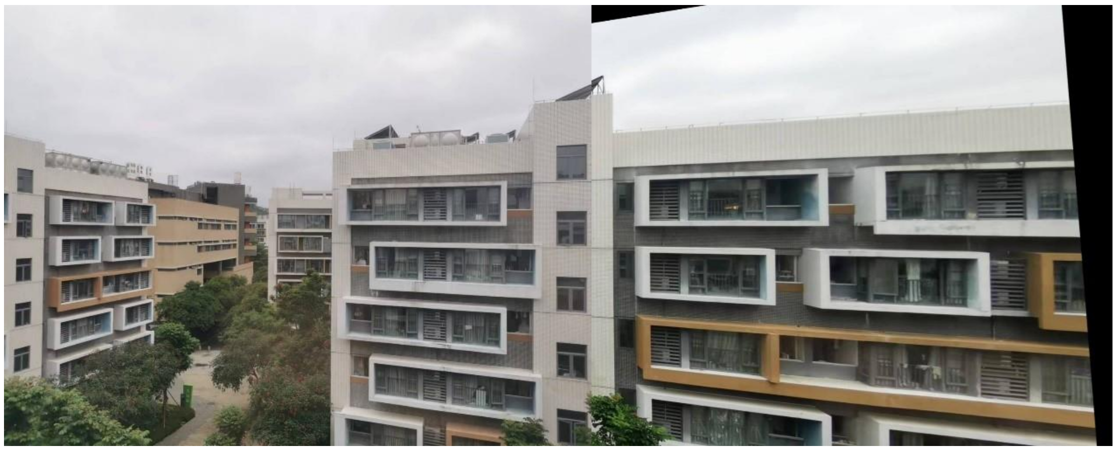
The original right image:



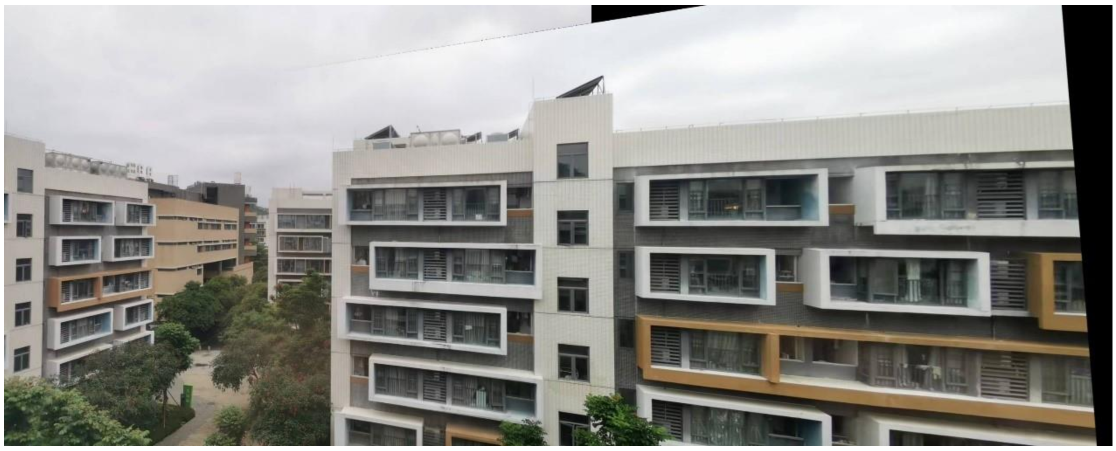
The image after the first matching:



The simple fusion:



The optimized fusion:



Discussion and summary

The above are the mathematical details and results of the algorithm. In the specific implementation process, the feature point matching uses the feature point detector and description extractor of OpenCV. Compared with the original algorithm, the effect is significantly improved after the weighted fusion of overlapping parts in the final result. However, the fused image still has a black area because the y-axis coordinates of the feature points of the two original images do not correspond. This problem was solved by my one of my teammates Sekai.

Reference

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359, 2008.

[2] HE Gengxin1, MA Jiawen2, ZHANG Xike1, PAN Dawei2, CHEN Liang2, An improved image mosaic algorithm based on SURF and RANSAC

Digital Image Process Project Report

Image Fusion using ORB

1. Introduction:

For the group project of this course, our chosen topic is **image fusion**. Each member in our group would like to use one kind of feature point extraction and matching method, such as "SURF", "ORB", and the built-in function of OpenCV, to implement our own algorithms respectively. And for all the contents in this report, I used the "ORB" method to implement image fusion. These are what I have implemented in the project:

1) Image Fusion with **C++**,
2) Rewrite the program using **Python** to implement fusion (**vertical stitching**),
3) Rewrite and optimize the **Python** program to mitigate cracks (**horizontal stitching**).

This report will describe the algorithm steps to achieve image fusion and detailly explain the results and characteristics of all three implementations.

2. Algorithms:

2.1 Feature point extraction and matching using ORB:

ORB first determines candidate corners, and then performs non-maximum suppression, and then achieves three advantages, **scale invariance**, **rotation invariance**, and insensitivity to **light changes**, through some algorithms. It is a fast and stable feature point detection and extraction algorithm. In python code, the **KNN** algorithm is used to match the feature points.

ORB uses the Brief algorithm to calculate the descriptor of a feature point. The core idea of the BRIEF algorithm is to select N point pairs in a certain pattern around the key point P and combine the comparison results of these N point pairs as a descriptor. Here are the basic steps of ORB algorithms:

1) Make a circle O with the key point P as the center and d as the radius.
2) Select N point pairs in a certain pattern in circle O. Assuming $N = 4$, the currently selected 4-point pairs are marked as:

$$P_1(A,B)、 P_2(A,B)、 P_3(A,B)、 P_4(A,B)$$

3) Define operation T:

$$T(P(A,B)) = \begin{cases} 1 & I_A > I_B \\ 0 & I_A \leq I_B \end{cases}$$

where IA is the intensity of the A

4) Perform T operations on the selected point pairs respectively, and combine the obtained results.

$$T(P_1(A,B)) = 1$$

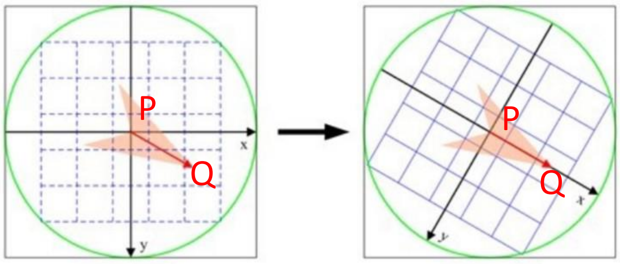
$$T(P_2(A,B)) = 0$$

$$T(P_3(A,B)) = 1$$

$$T(P_4(A,B)) = 1$$

Then the final descriptor is: 1011

5) Because the center of the circle is fixed and rotates as the object rotates. When we use PQ as the coordinate axis, under different rotation angles, the points we take out in the same point taking mode are consistent. This solves the problem of rotational consistency. The formula for calculating the centroid Qx is as follows



$$M_{00} = \sum_{X=-R}^{R} \sum_{Y=-R}^{R} I(x,y)$$

$$M_{10} = \sum_{X=-R}^{R} \sum_{Y=-R}^{R} xI(x,y)$$

$$M_{01} = \sum_{X=-R}^{R} \sum_{Y=-R}^{R} yI(x,y)$$

$$Q_X = \frac{M_{10}}{M_{00}}, Q_Y = \frac{M_{01}}{M_{00}}$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \qquad \theta = \operatorname{atan2}(m_{01}, m_{10})$$

The principle of KNN is that when predicting a new value x, it is judged which category x belongs to according to the category of the nearest K points. It is used to match the feature points pairs in the different images. Euclidean Distance is used to measure distance, which measures the absolute distance between points in space. The formula is:

$$d_2(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2.2 Image registration:

I used random sample consensus to calculate the homograph matrix, which will be used for image warping, since the angles of the original images may be different. Affine transformation does not change the parallel property, but no longer keeps the angle unchanged, which can be expressed as:

$$X' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \tilde{X}$$

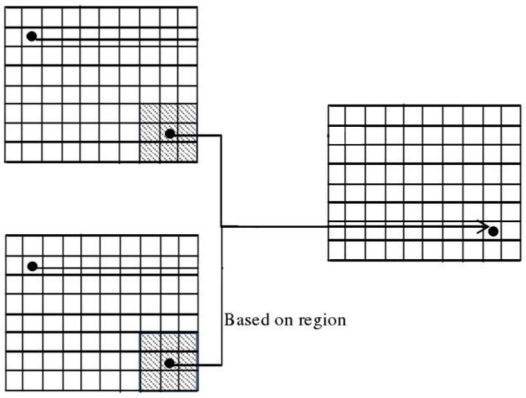
2.3 Image copy:

Copy the second image directly to the registration image. No need to change any angle or color of the second image currently.

2.4 Image fusion (implemented in 3.3):

Use weighted fusion to slowly transition from one image to the other image in the overlapping part, making the edge between the images more natural. In Python code, I apply a **weighted matrix** as a mask for image blending.

Considering the fusion of two images, let the input images be $I1$ and $I2$, the fusion image If can be calculated by the following formula:



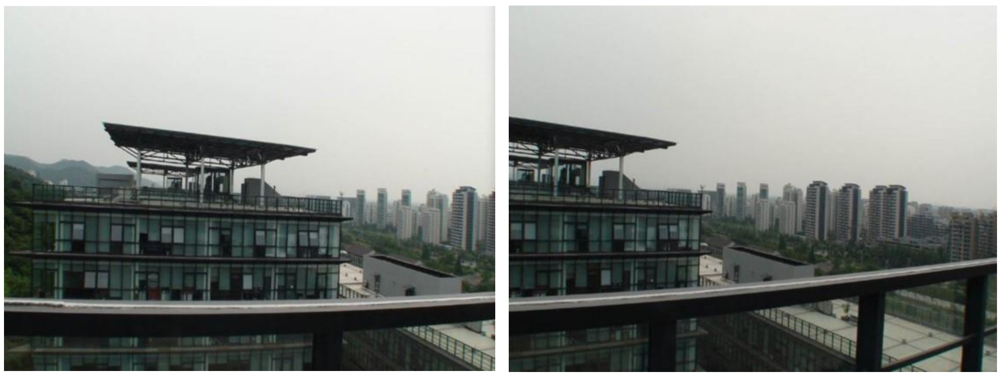
$$I_F(h, w)=w_1(h, w)I_1(h, w)+w_2(h, w)I_2(h, w)$$

where $w1$, $w2$ are the weight functions corresponding to $I1$ and $I2$, respectively, and $w1 + w2 = 1$.

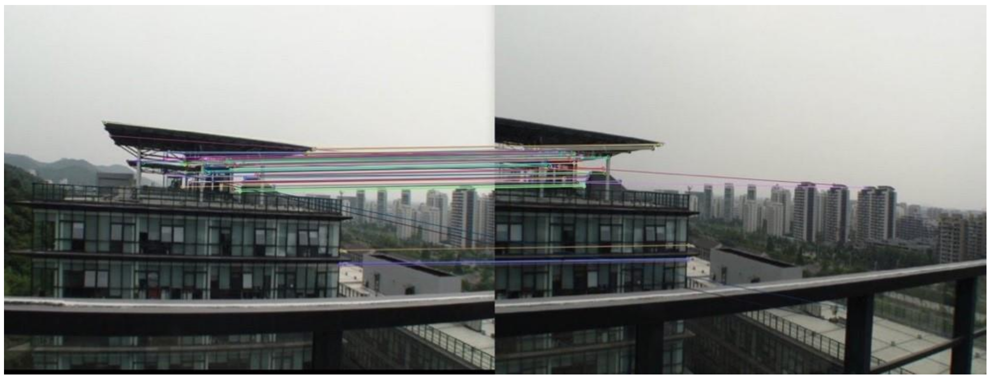
3. Implementation:

3.1 Implementation with C++:

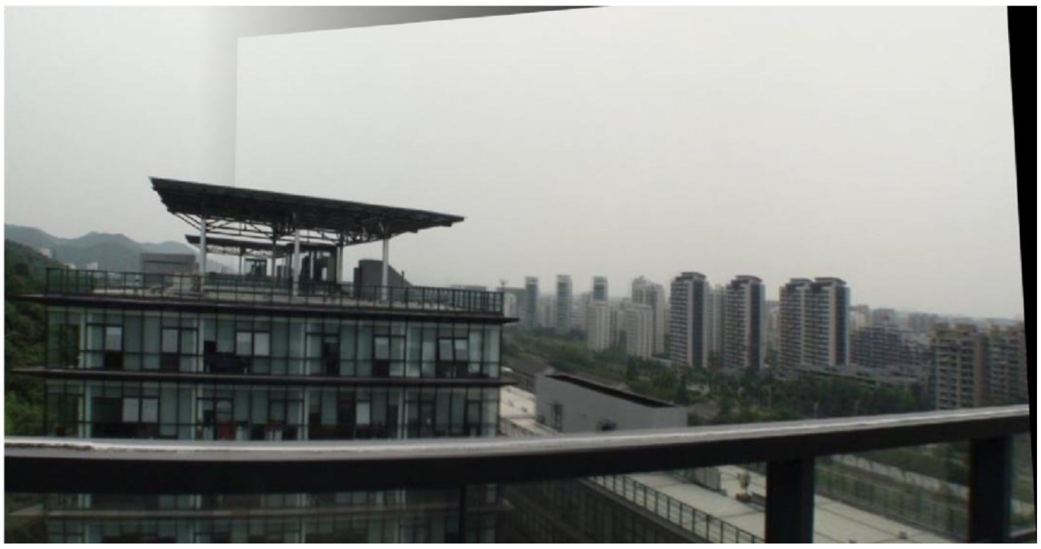
3.1.1 Results:

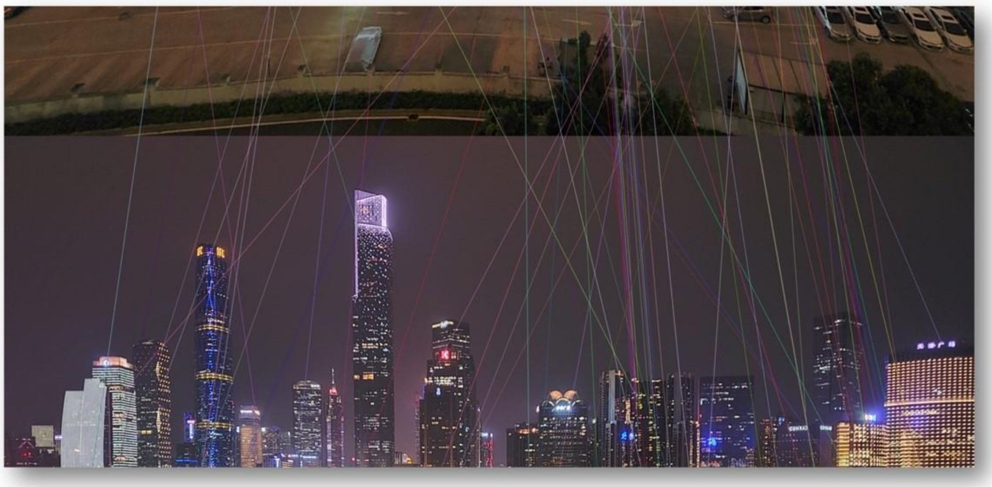


(Original images)



(Matching result)


(Fusion result)


(Example: matching errors)

3.1.2 Evaluation:

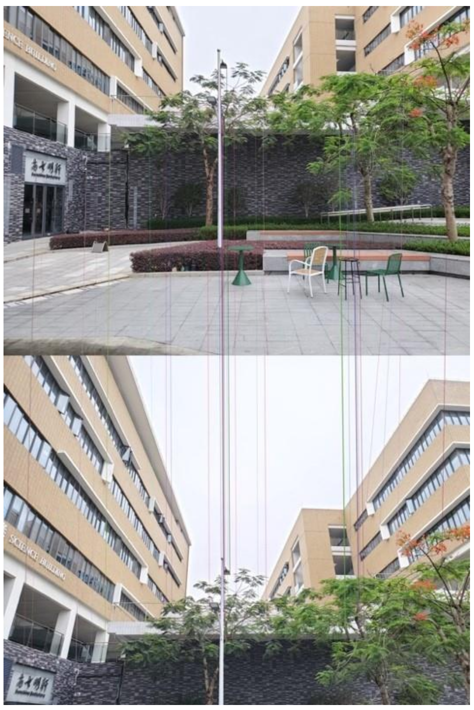
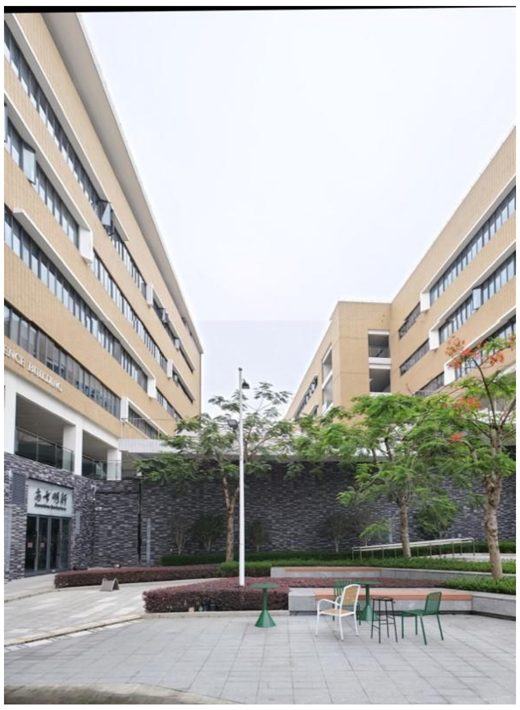
This C++ program is referenced from an online tutorial, we can see that it handled well the matching of feature points and mapping of image angles. But there is an obvious dividing line in the output image, and it cannot properly handle the images with complex and repetitive feature points, since they will confuse the program and output with a messy result. So it is necessary to optimize the program to fix these issues.

3.2 Implementation with Python (Verticalstitching):

3.2.1 Results:

Group 1

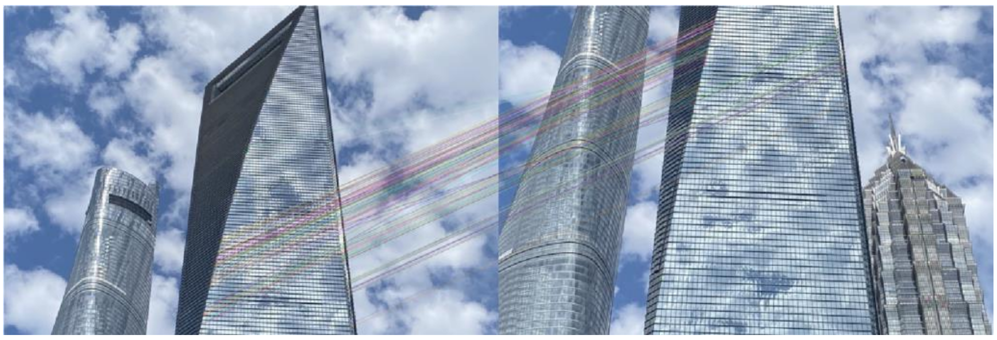
(Original images)

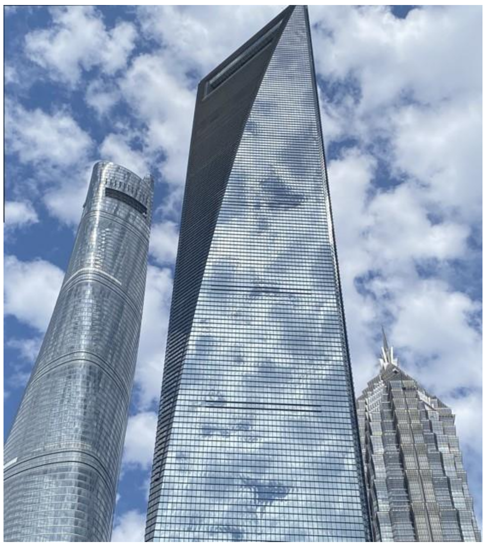



(Matching result) (Fusion result)

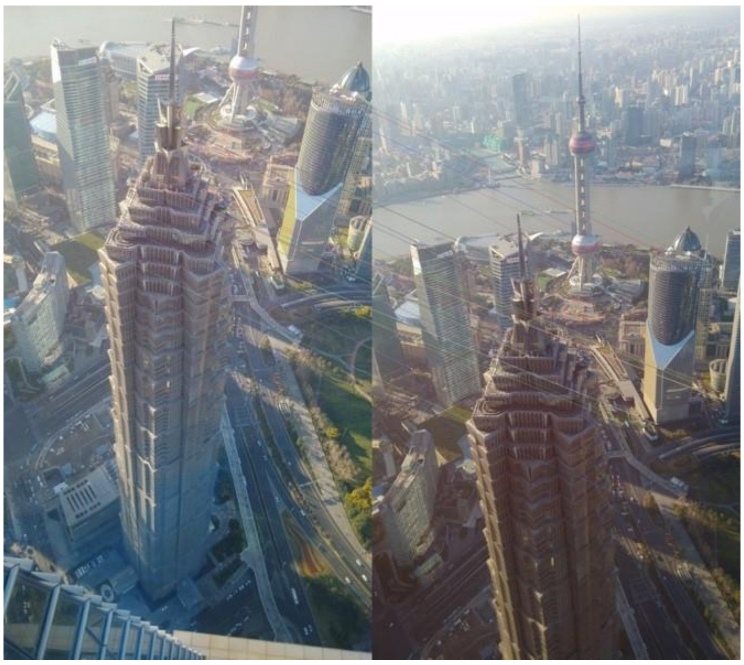
Group 2

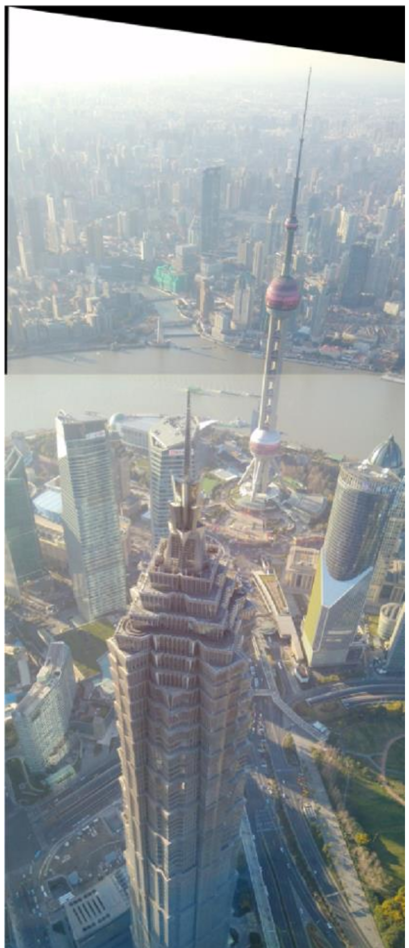
(Original images)


(Matching result)


(Fusion result)

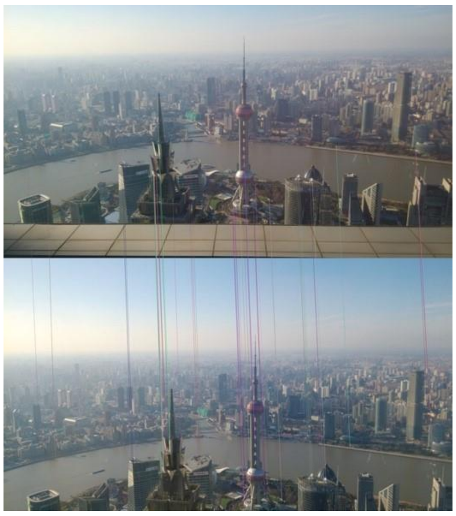
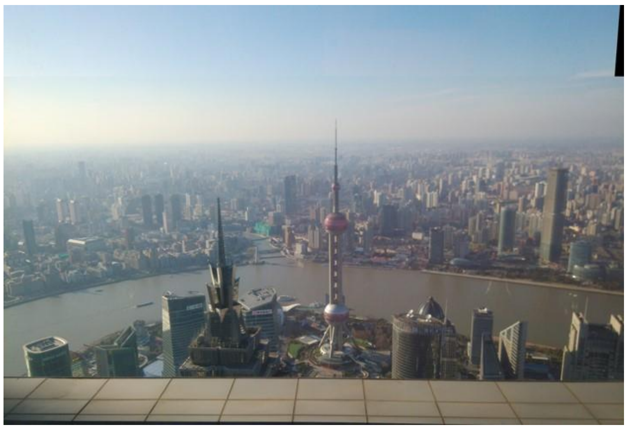
Group 3


(Matching result)


(Fusion result)

Group 4

(Original images)

(Matching result)

(Fusion result)

3.2.2 Evaluation:

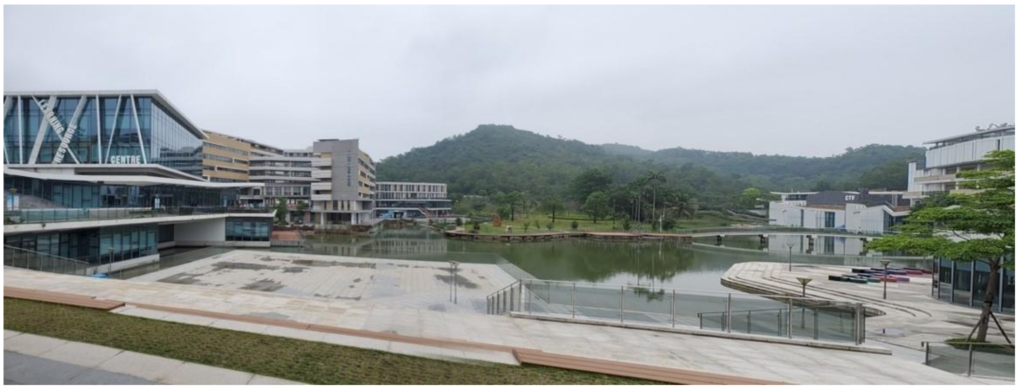
The images implemented by Python have a very promising effect on expanding the image content and the KNN feature points matching algorithm works nicely. However, there is still a little drawback in the output image, that is, If the light and dark difference of the two images is too great, we still can see a dividing line in the outputs. And the following optimized program is to solve this problem.

3.3 Optimization with Python (Horizontal stitching):

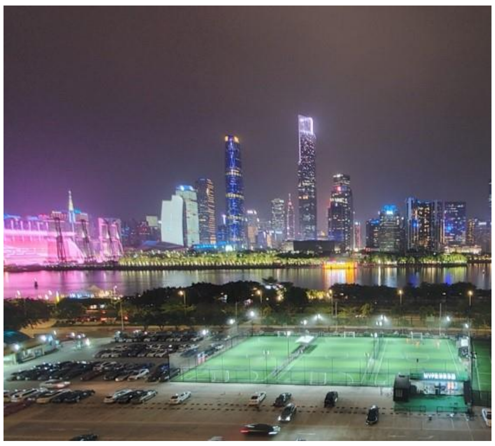
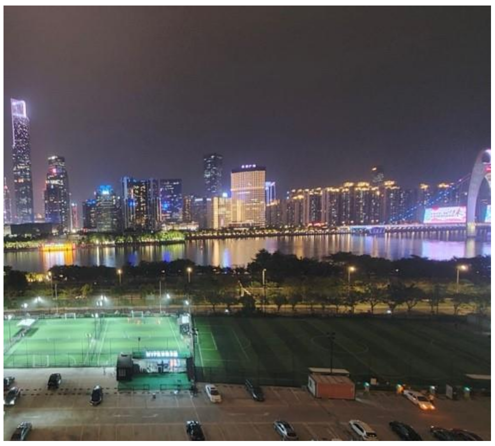
3.3.1 Results:

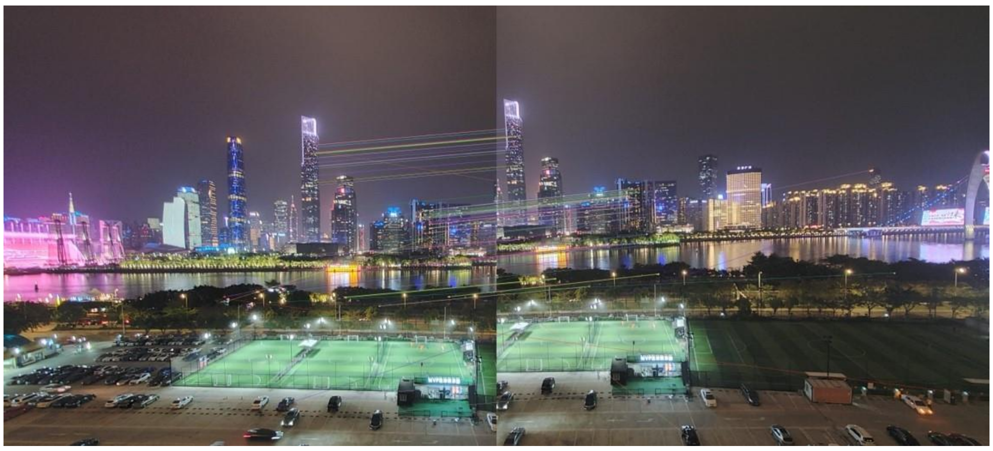
Group 1

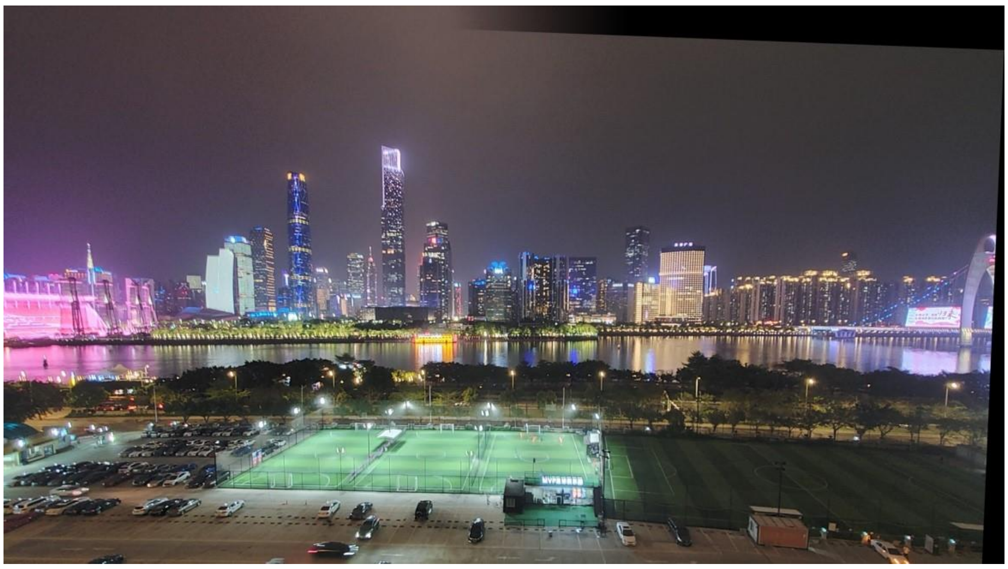
(Original images)


(Fusion result)

Group 2

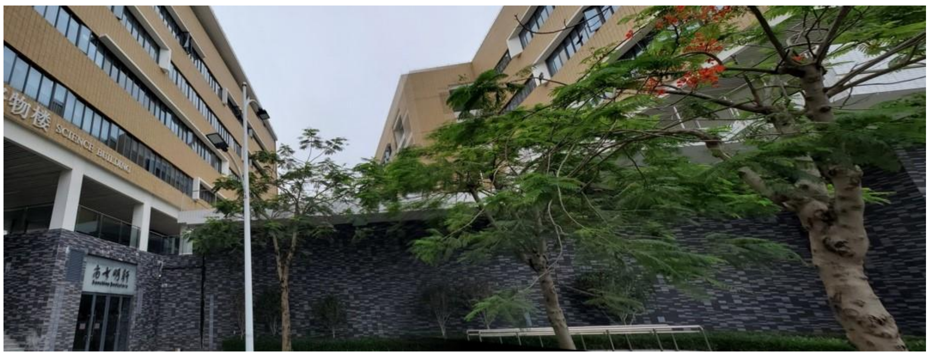


(Original images)


(Matching result)


(Fusion result)

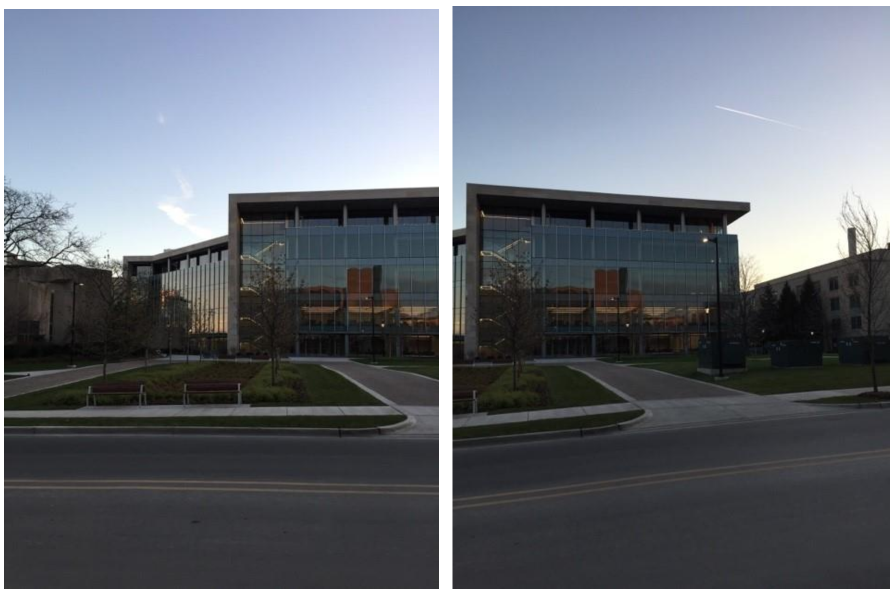
Group 3

(Original images)

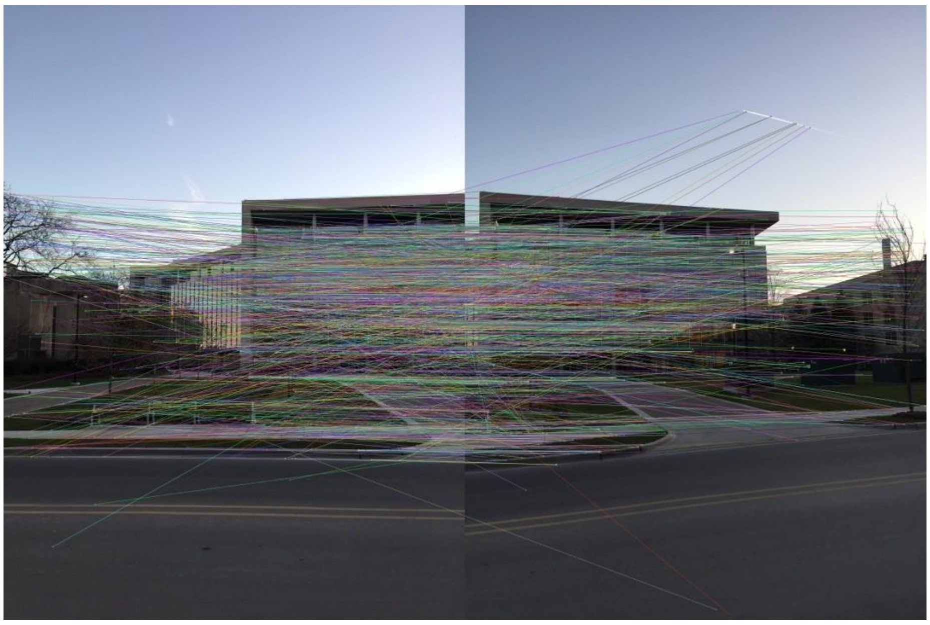


(Fusion result)

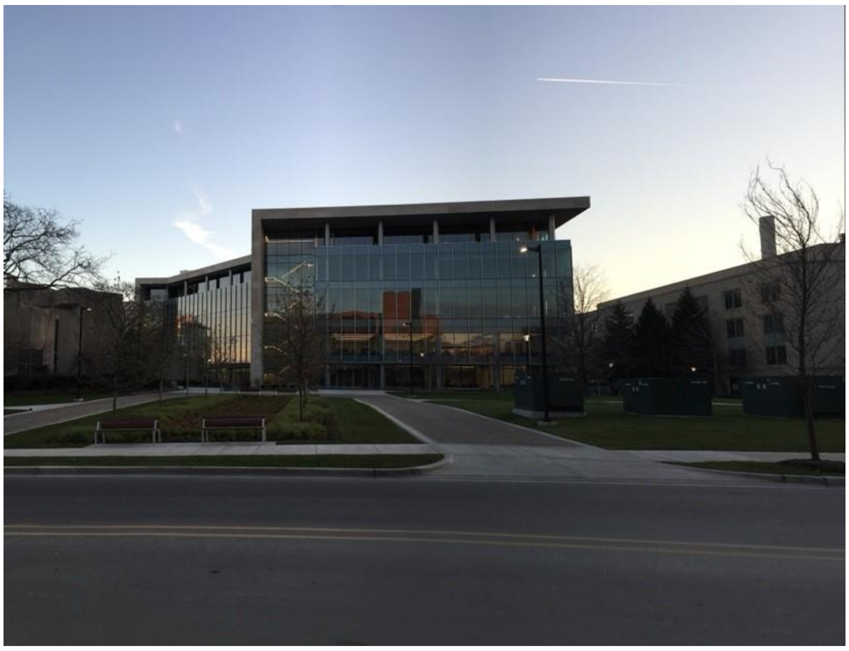
Group 4



(Original images)



(Matching result)



(Fusion result)

3.3.2 Evaluation:

This optimized program has the best fusion effect so far. It can be seen the edge between the two images is well handled now. In addition, this program can handle the images (Group 2) with complex and repetitive feature points which cannot be properly processed by the first program.

4. Summary:

Three different image fusion programs based on the ORB feature points extraction method are implemented in my project. The C++ program has a decent stitching effect but cannot handle gradient at the border between two images, and it also cannot handle the images with complex and repetitive feature points. The last two codes have the outputs with significantly better quality since they have very precise matching for each pair of feature points. Moreover, the last program handles light differences at the dividing line between the two images well, while stitching the images correctly.

References:

[1] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 25642571). IEEE.

[2] “Implementation of OpenCV Image Stitching and Image Fusion.” *Script House*, https://www.jb51.net/article/219981.htm.

[3] Kushalvyas. “Kushalvyas/Python-Multiple-Image-Stitching: Implementation of Multiple Image Stitching.” *GitHub*, https://github.com/kushalvyas/Python-Multiple-Image-Stitching.

[4] “OpenCV-Python ORB Feature Matching (Practice).” *OpenCV-Python ORB Feature Matching (Practice)_Loner...'s Blog-CSDN blog_opencv Orb Matching Python*, https://blog.csdn.net/weixin_43852752/article/details/106132738.

[5] super kid.css-1cd9gw4{margin-left:.3em;}CV. “Image Feature Algorithm (3)--A Brief Description of Orb Algorithm and Orb Feature Matching Practice in Python.” *Zhihu Column*, https://zhuanlan.zhihu.com/p/261966288.

[6] “Weighted Average Fusion Removes Seams from Image Stitching.” *Weighted Average Fusion Eliminates Patchwork of Image Stitching Xiaoxifei's Blog - CSDN Blog*, https://blog.csdn.net/xiaoxifei/article/details/103045958.

Digital Image Process Project Report

Image Fusion Part 3

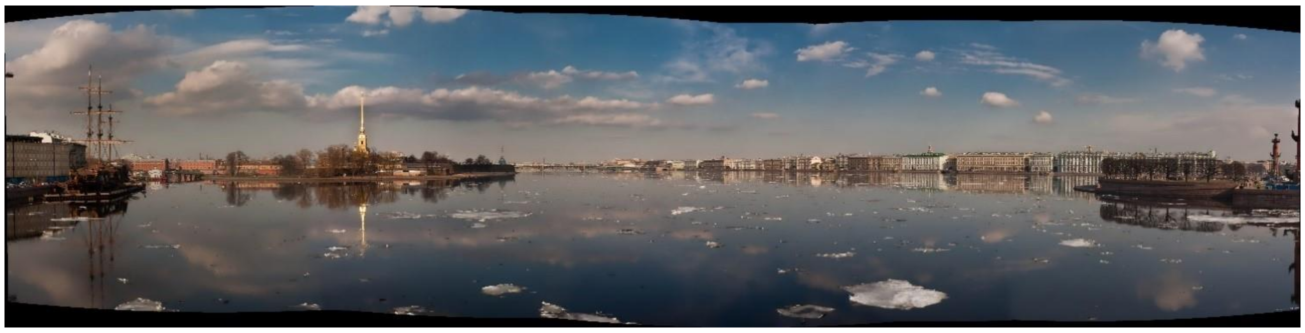
OpenCV Stitch method realizes image stitching and black edge processing

Stitch introduction

Stitch is a class in the OpenCV library. Its algorithm for detecting feature points is based on SURF or ORB. OpenCV has encapsulated this assembly algorithm, so we can call it directly. It's powerful, and it's easy to put pictures together.

We use the stitch method to stitch these six pictures together.

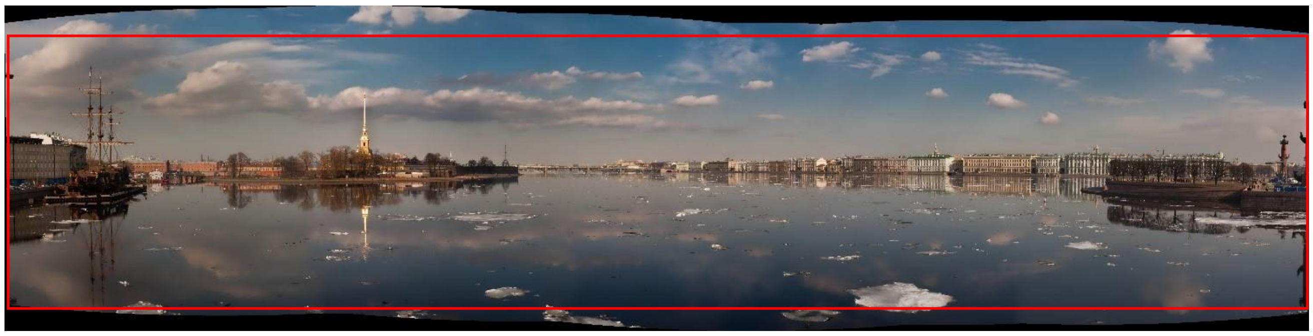
After we use the stitch algorithm, we may get an image like this:



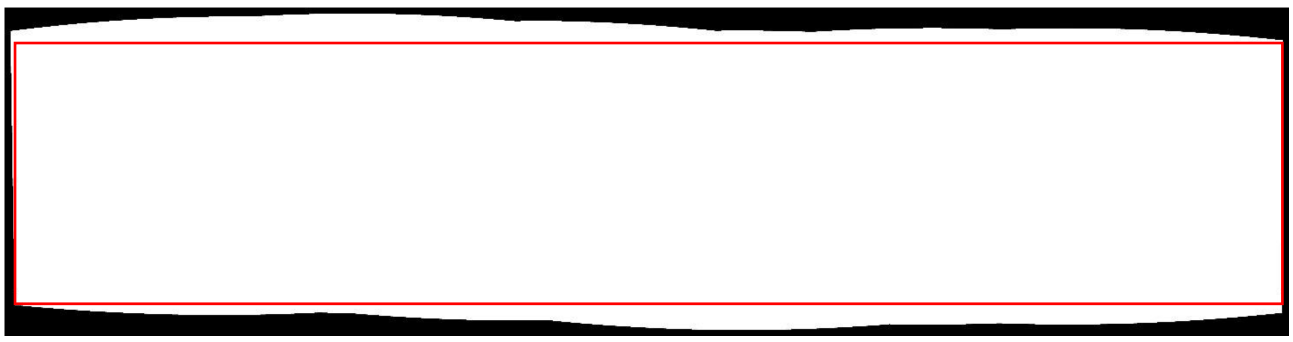
Remove the black edge from the image

Some black edges are sometimes visible in spliced images. We need to do something to get rid of the black edges.

First, we want to find the largest bounding box that is contained by the original image, like this:



The first step is to add a black border around the original image to ensure that the full outline of the image can be found, then the panorama is transformed into a grayscale image, and all pixels that are not 0 are set to 255 as the foreground, while the gray values of other pixels are 0 as the background:



Now that we have the binary image of the panorama, apply contour detection to find the boundary box of the largest contour:

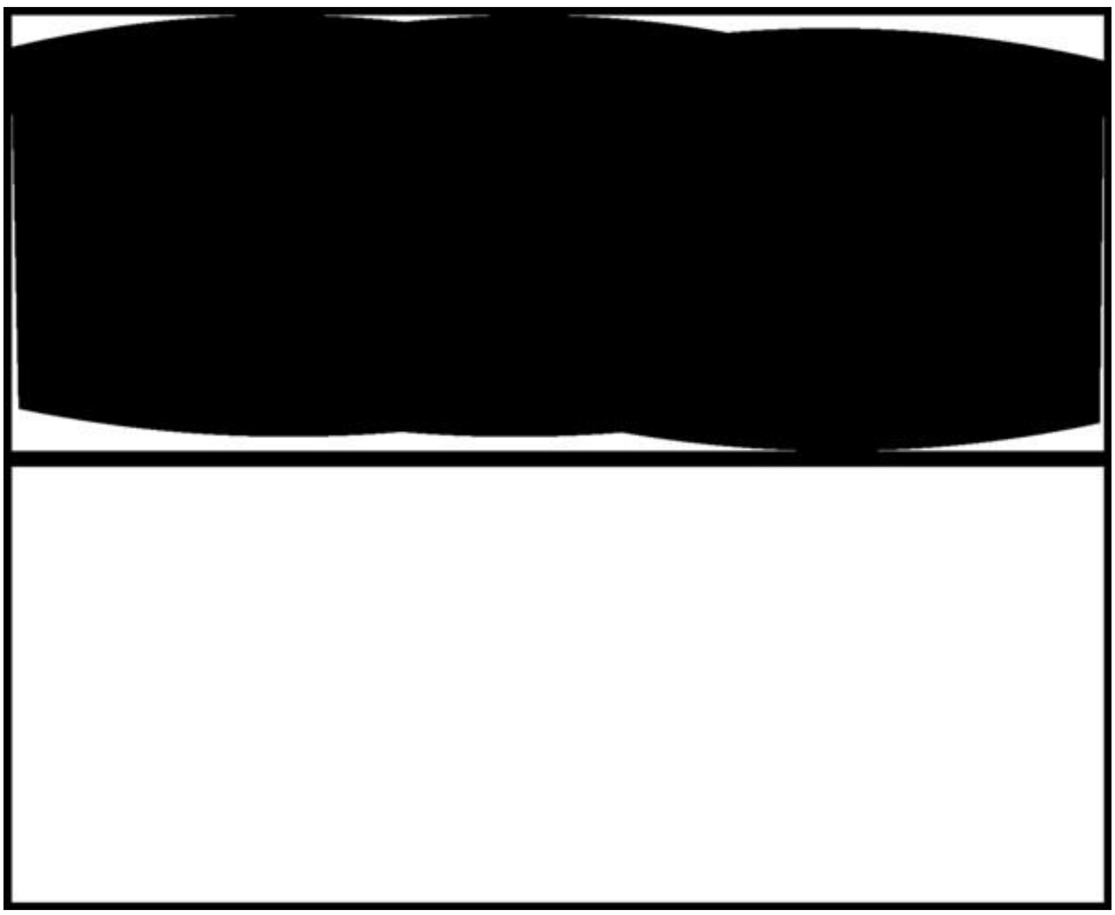
We create two copies of this bounding box, called minRect and sub:

● minRect: the white area of this mask will be gradually reduced until it is just enough to fit into the inner of the panorama. ● Sub: the mask is used to determine whether minRect needs to be further reduced to get the rectangular area that meets the requirements

Continue to erode minRect, and then subtract the threshold image obtained before with minRect to get sub, Then judge whether there are non-zero pixels in the sub. If not, then the minRect at this time is the largest rectangular area inside the panorama we finally want.

The changes of sub and minRect in the while loop are shown in the following animation:

(Sub on the top, minRect on the bottom)



We have found the largest rectangular outline that the original image can contain and reassigned it according to the coordinates.

Result:

This is a panorama obtained by stitching 6 pictures with the stitch algorithm and removing the black edges.