CV HW3 -
Automatic Panoramic Image Stitching

Group 18

Introduction

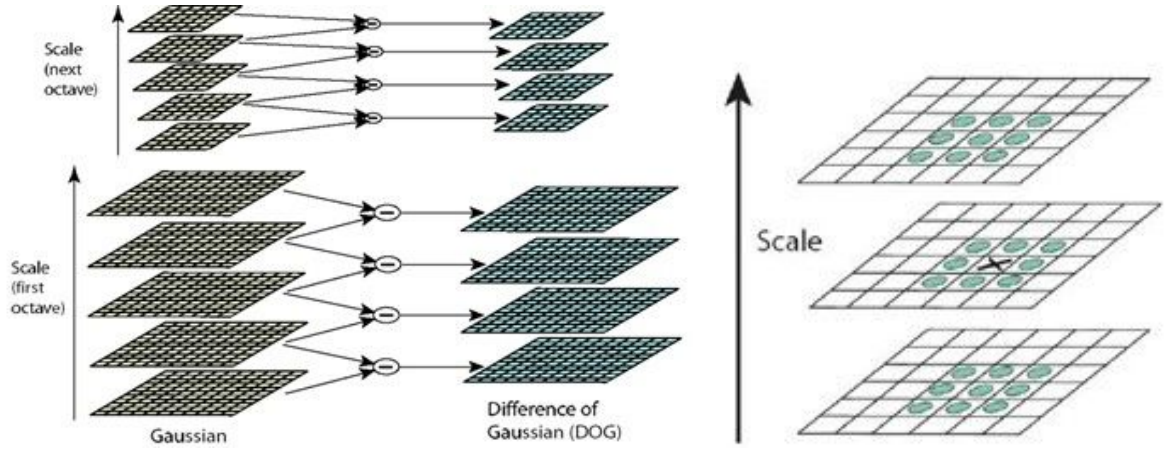
Image stitching is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image.
For image stitching, the process can be divided into four steps. First step is key points detection and feature description by SIFT. Next, we match the features between two images according to the description. After the matching, we use RANSAC algorithm to estimate a homography matrix. At last, we apply a warping transformation to stitch two images.

Step 1. Interest points detection & feature description by SIFT

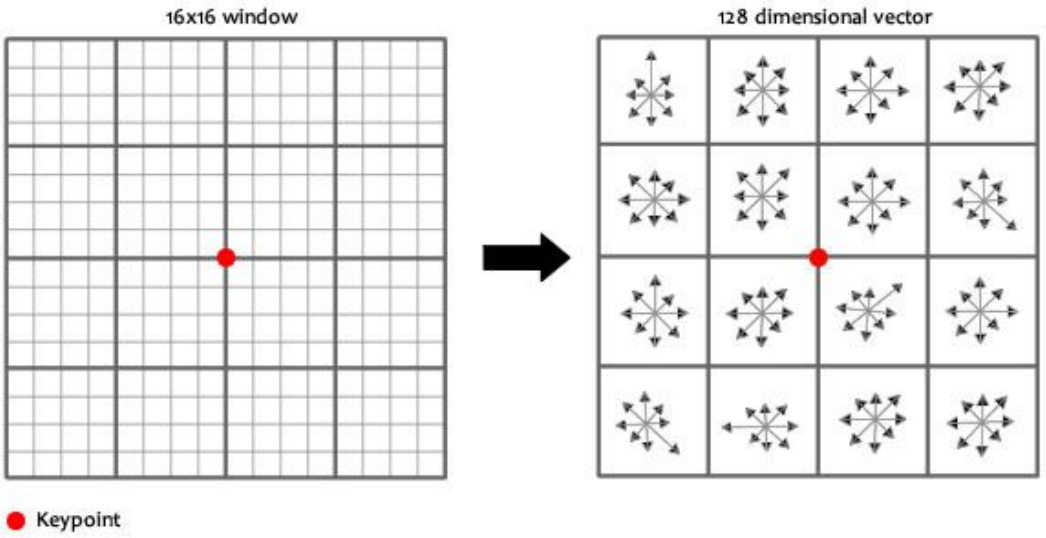
We simply use the OpenCV function:

1. Initiate the SIFT detector
   sift = cv2.xfeatures2d.SIFT_create()
2. Get the keypoints and descriptors with SIFT
   kp1, des1 = sift.detectAndCompute(g1,None)

The concept of SIFT is using DoG(Difference of Gaussians) to find the key points with different scales. The key points are local maximum not only compared with the neighbour in same scale but also the points of near scales(+σ and -σ).



For each key points, pick 16x16 near points to get their orientation with 8 bin and then divide them to 4x4 window. So we can get 4x4x8=128 dimensions vector for each key point descriptor.

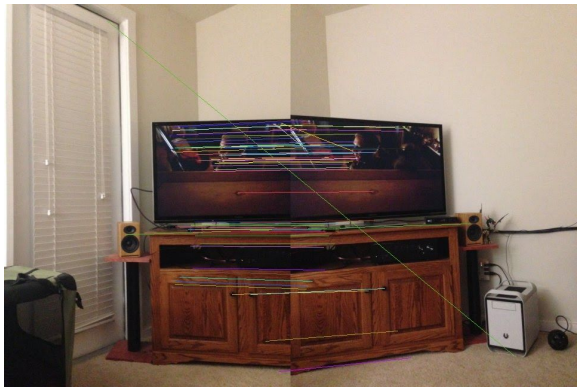


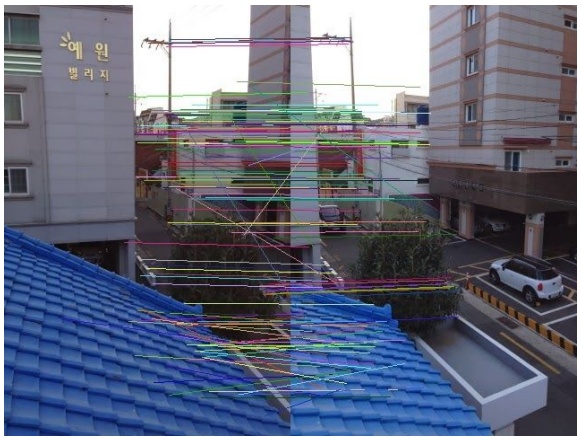
Step2: Feature matching by SIFT features

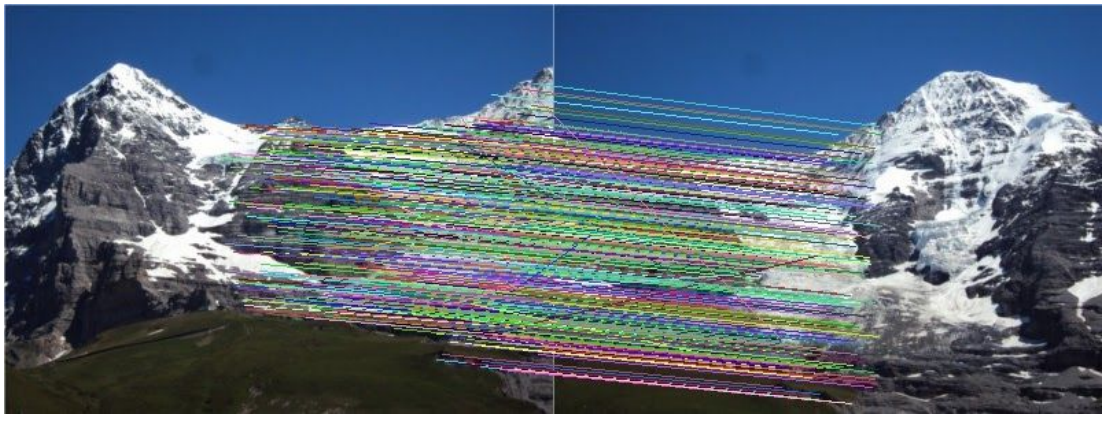
In this step, we need to find the good match pairs in two images. So we compare the two descriptors from OpenCV function. For 128 dimensions, we estimate the distance of each point to others. L2 norm distance is used for SIF in this part., if we want to use binary string based descriptors like ORB, BRIEF, BRISK, Hamming distance is applicable.
(https://www.docs.opencv.org/3.4.2/dc/dc3/tutorial_py_matcher.html)
To avoid the error pairs of features, we use ratio distance to extract the good pairs.

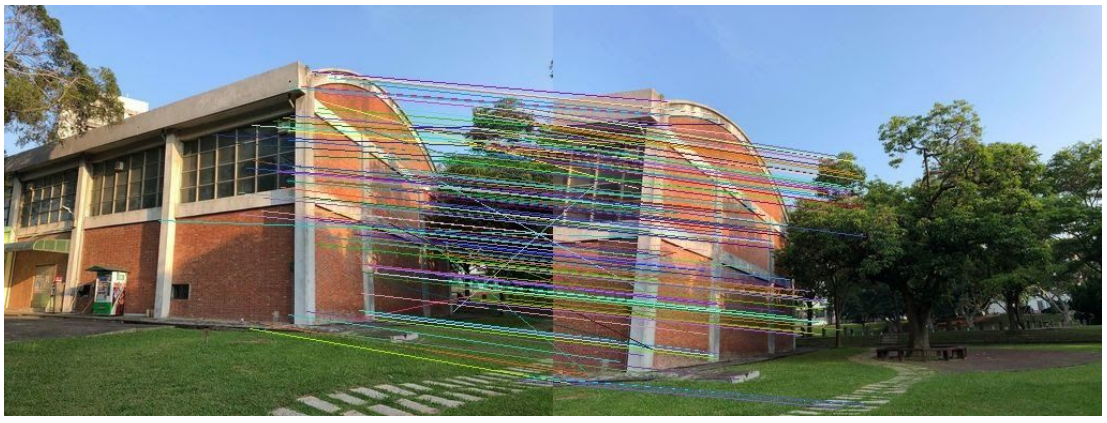
$$\frac{\|f_1 - f_2\|}{\|f_1 - f_2'\|}$$

If the ratio distance less than specific threshold, we define the match is a good match.
Our matching result(with feature points and feature corresponding line):









Step 3 Find Homography Matrix

The work in this part is very similar to the homework 1 camera calibration.

$$s_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$p_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i x_i' & y_i x_i' & x_i' \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y_i' & y_i y_i' & y_i' \end{bmatrix}$$

$$PH = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x_1' & y_1 x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 y_1' & y_1 y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 x_2' & y_2 x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 y_2' & y_2 y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 x_3' & y_3 x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 y_3' & y_3 y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 x_4' & y_4 x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 y_4' & y_4 y_4' & y_4' \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = 0$$

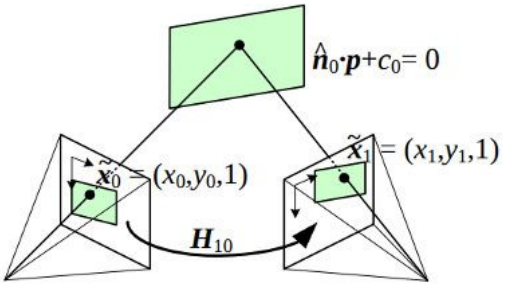
Minimize $||PH||$,subject to $||PH||^2 = 1$ to avoid trivial solution

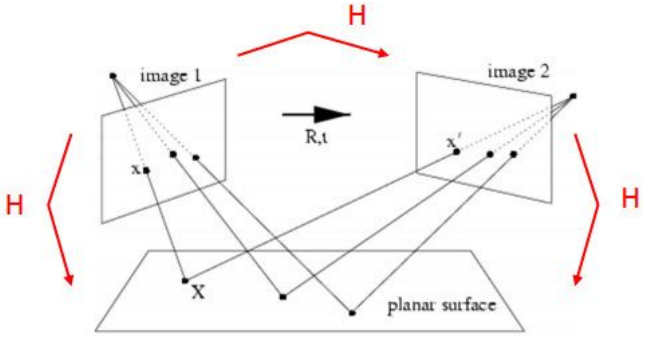
We use singular value decomposition(SVD) method to solve it.

$$PH = UDV^T$$

We set h equal to the last column of V , in other words, the last row of V^T , then reshape the vector h(1,9) to H(3,3).

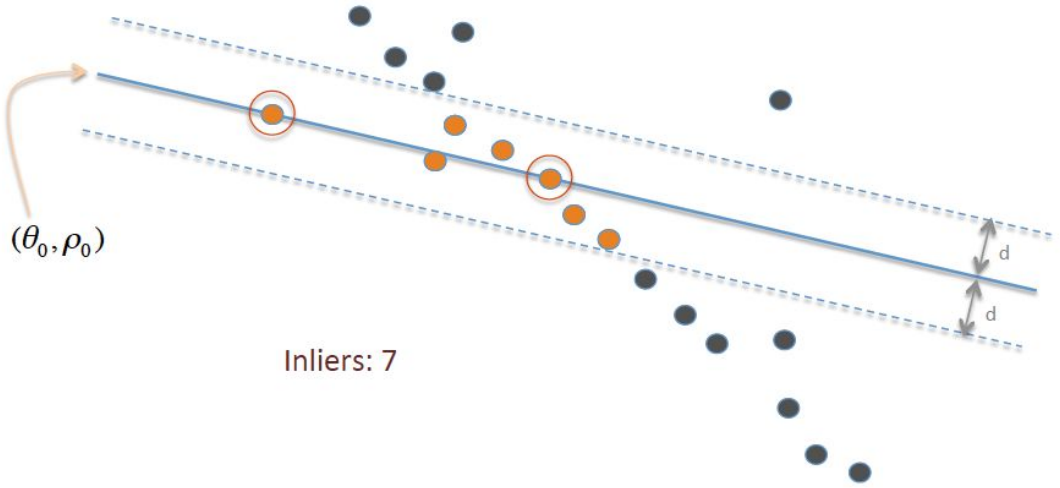
With the paired key points we got in the previous part, they should be the same point in the 3D world referring to the same feature. Therefore, we use the function we implement in homework 1 to find the relative homography matrix between 2 images.





Step 4 RANSAC (RANdom SAmple Consensus)

RANSAC is a method to estimate the parameters of model from a set of data. The data may contain some of outliers. However, with iterative sampling from the data, the outliers can be ignored in some probabilities. And the final convergent result may be the best solution parameters for the model.



In this part, we randomly choose some paired key points and calculate the best homography matrix to fit the line. If the sample points are outliers, the loss between the line distance should be large enough (Absolute sum). As a result, we can use some threshold with the limited iteration to find the best solution.

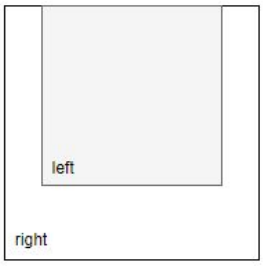
1. Randomly select num_sample paired key points
2. Calculate selected key points' homography matrix
3. Evaluate the loss between prediction and the ground true
4. Save the best score homography matrix

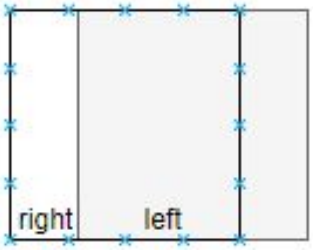
Warp image to create panoramic image

The three situations of warping image is considered for one dimention.
For example, in x axis, we get the situations:

1. max x > original left image boundary

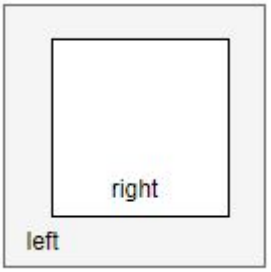
2. min x < 0



or


(not sure if this will happen)

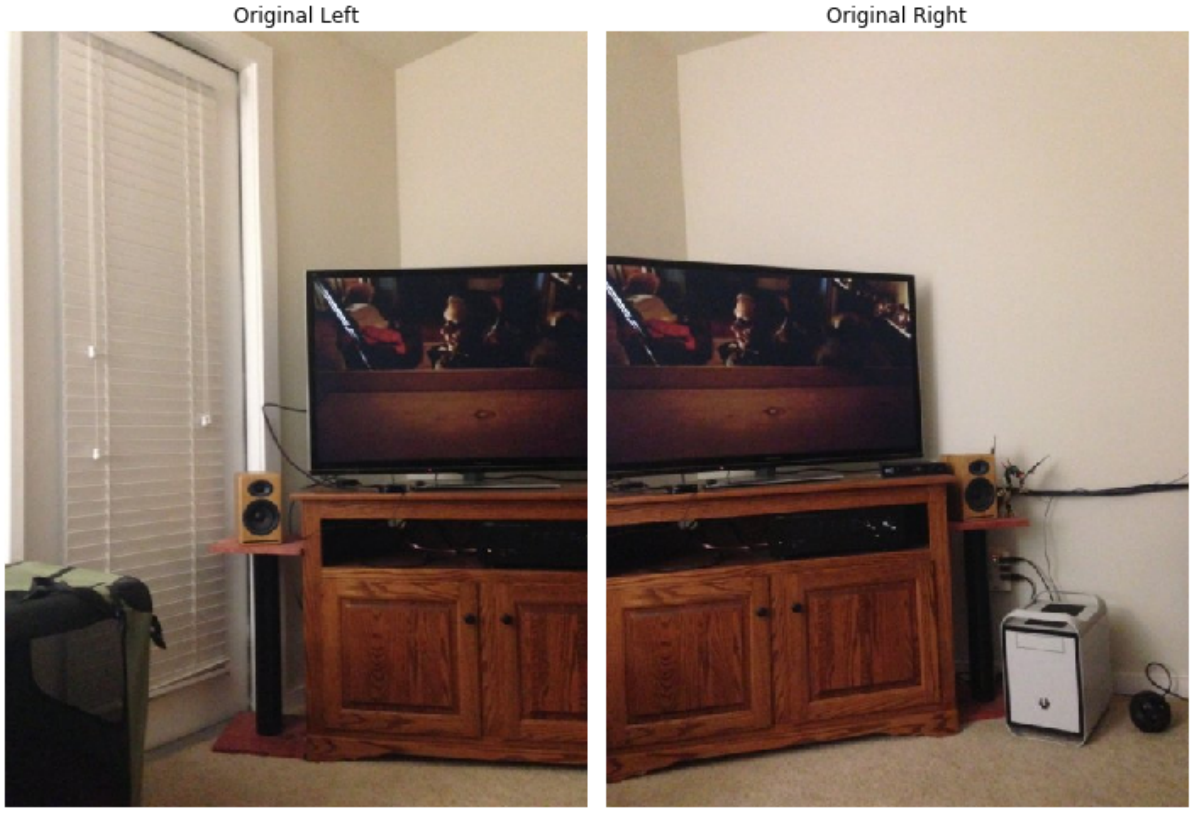
3. 0 < min & max x < original left image boundary



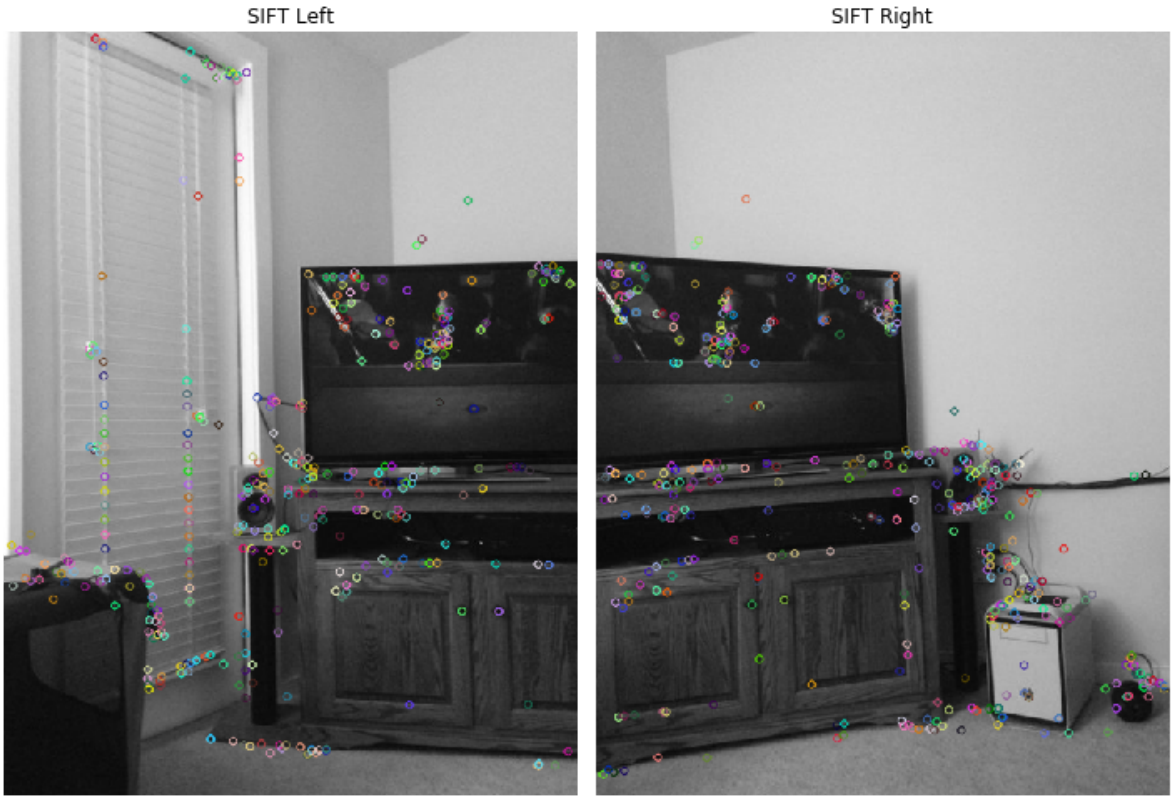
So the warping image size in x direction can be found by comparing the total length we need in three situations. If the x coordinate appears negative value, we need to consider the image shifting.
These situations is also considered in y dimension when we implemented the image stitching.
So, the warping procedure can be done in three steps:

1. Get the boundary of source image
2. Transfer back to original coordination
3. Use the bilinear interpolation to get the pixel intensity in the coordination

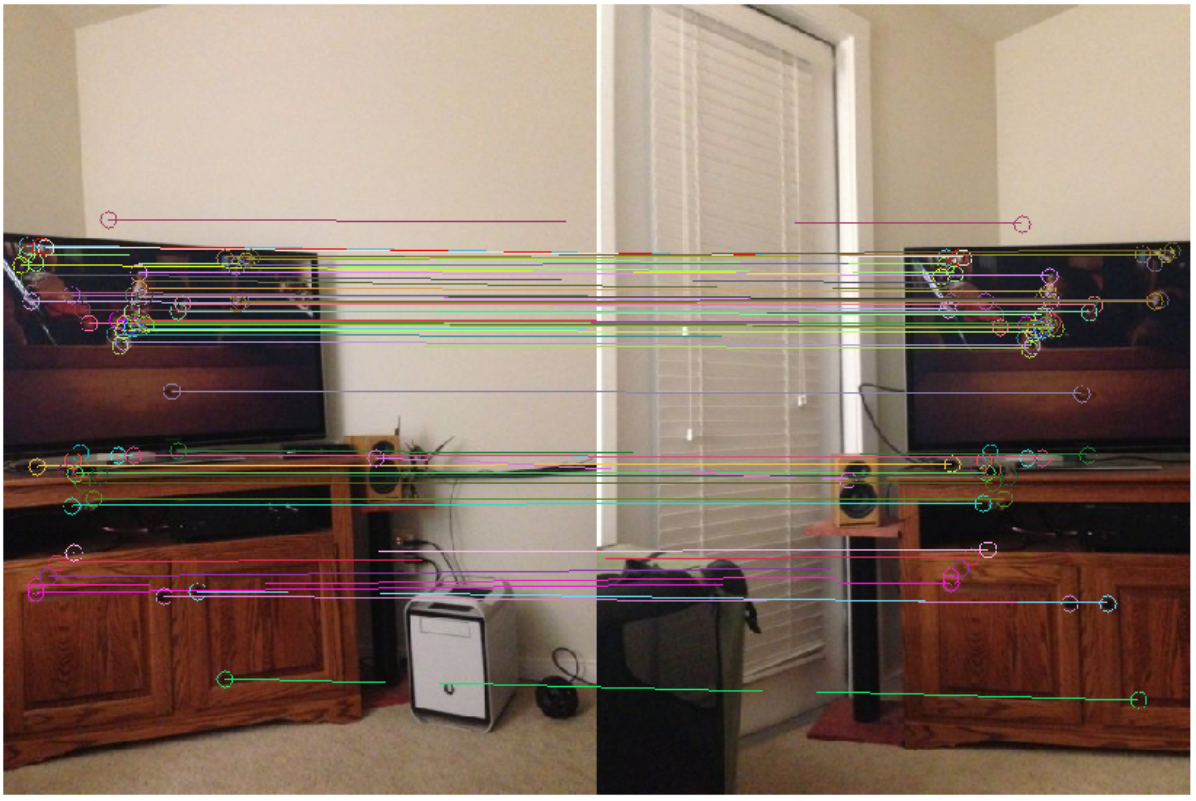
Experiment Result



(a) Original Image (Living Room)

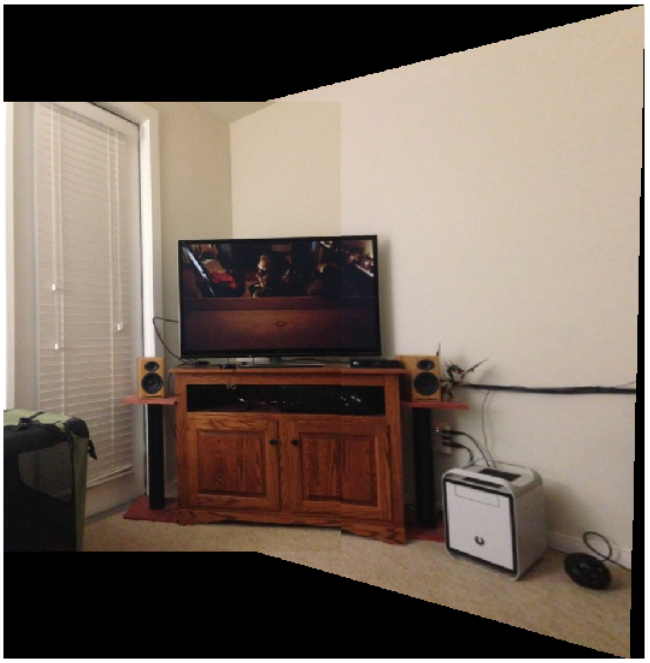


(b) SIFT Features from OpenCV



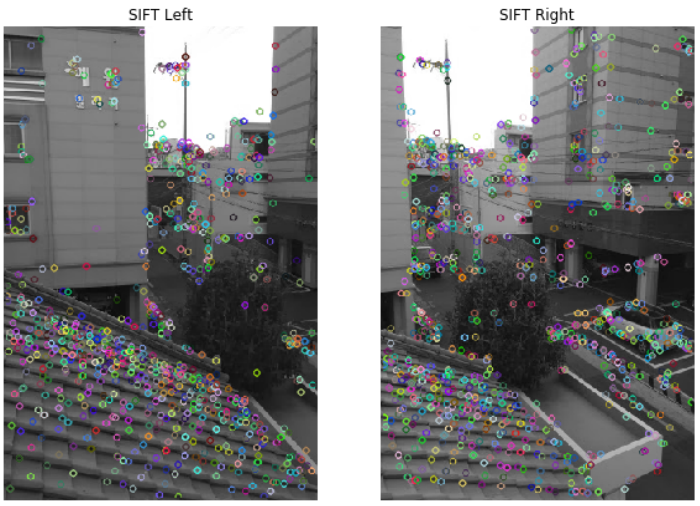
(c) Feature KnnMatch Implementation (k=2, threshold=0.5)

(d) Warpping Image

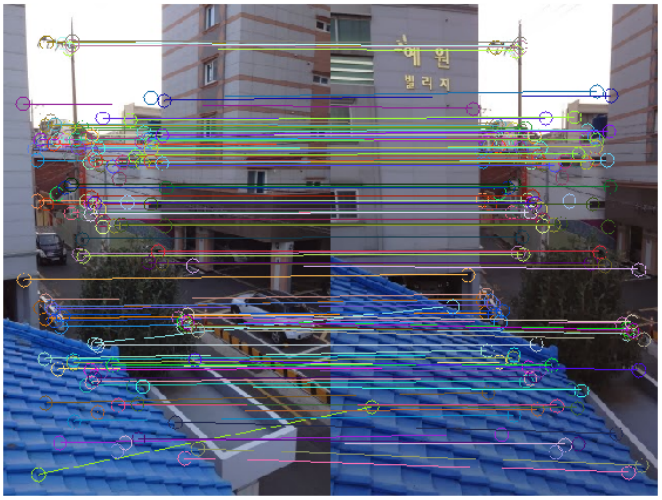


(e) Result after stitching and cropping

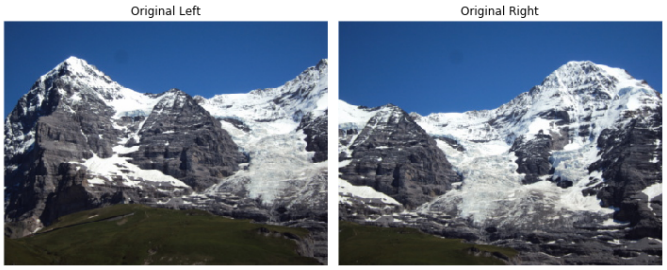
(a) Original Image (Roof)



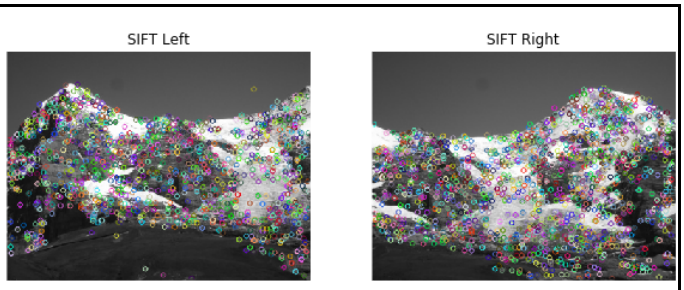
(b) SIFT Features from OpenCV

<table>
<tr>
<td>  </td>
<td> 
</td>
</tr>
<tr>
<td>(c) Feature KnnMatch Implementation
(k=2, threshold=0.5)</td>
<td>(d) Warpping Image</td>
</tr>
</table>

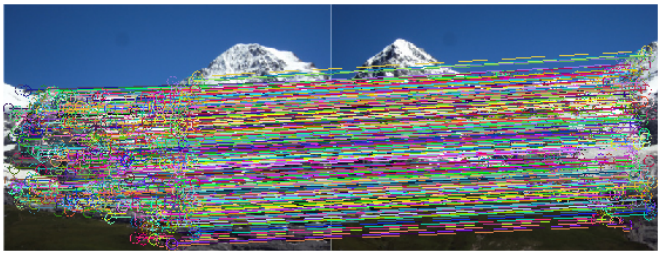
(e) Result after stitching and cropping



(a) Original Image (Hill)

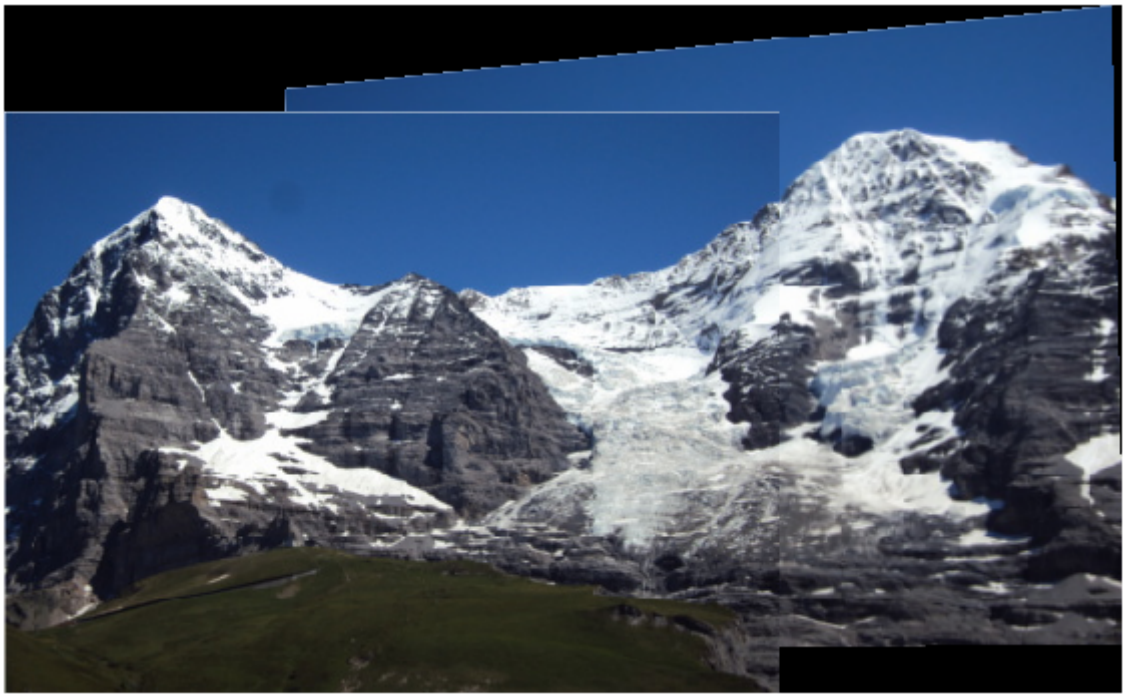


(b) SIFT Features from OpenCV

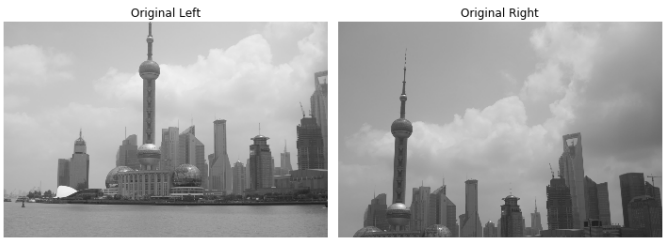


(c) Feature KnnMatch Implementation (k=2, threshold=0.5)

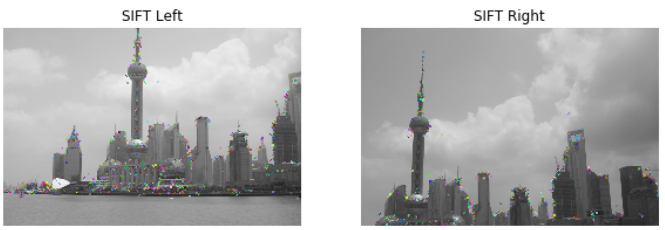
(d) Warpping Image



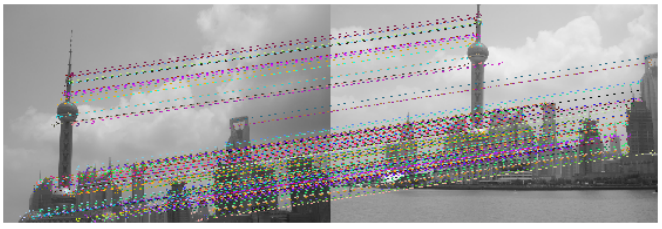
(e) Result after stitching and cropping



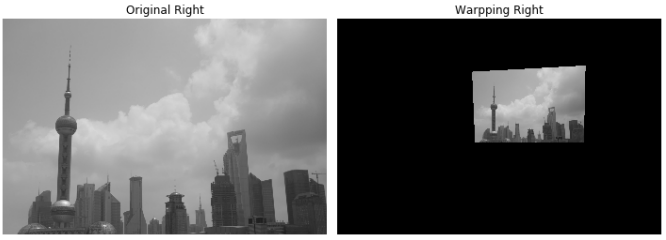
(a) Shanghai (Adobe Panoramas Dataset)



(b) SIFT Features from OpenCV



(c) Feature KnnMatch Implementation (k=2, threshold=0.5)



(d) Warpping Image

(e) Result after stitching and cropping

Discussion

Feature Matching

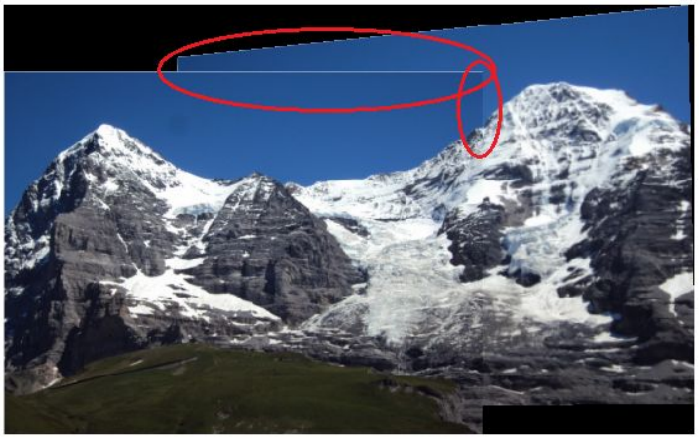
Table 1 Processing Time Ours v.s. OpenCV
(Including line drawing and image print)

(#kp1, #kp2)	(429, 422) Living Rom	(1020, 958) Roof	(1491, 1257) Hill
Our (Colab)	1.69 sec (Best of 10 loops)	8.9 sec (Best of 3 loops)	17.3 sec (Best of 3 loops)
OpenCV (Colab)	0.026 sec (Best of 10 loops)	0.084 sec (Best of 3 loops)	0.158 sec (Best of 3 loops)
Faster	65x	106x	109x

The table above shows that our feature matching time is quite slow. The main issue is on the knnMatch function. In our implementation, for each source key points, we search all the destination points to calculate every pair points' distance. Afterward, we have to sort the distance to find the k shortest (closest) distance, which is the best k solution. The whole process is quite intuitive but time consuming, which require time complexity O(n^3)

Warping & Stitching

In this step, we only simply used stitching technique, so there's some artifact edge on the boundary of two images. We believe that the blending may improve this issue slightly. Yet, due to the deadline, we couldn't finish all the blending implementation on time.



OpenCV Version Issue

When we execute on OpenCV version above 3.4.x.x, there's a patent issue on SIFT algorithm.

```
OpenCV(4.1.0) This algorithm is patented and is excluded in this configuration; Set OPENCV_ENABLE_NONFREE CMake option and rebuild the library in function 'cv::xfeatures2d::SIFT::create'
```

The solution is to downgrade opencv-python and opencv-contrib-python versions.

For pip:

```
pip install opencv-python==3.4.2.16 pip install opencv-contrib-python==3.4.2.16
```

For Anaconda:

```
conda install -c menpo opencv
```

Conclusion

In homework 3, we used SIFT functions in OpenCV, and learned the method behind those functions. Then we use K-nn matching(k = 2) to decide which 2 key points are the same point in 2 images. Then the finding homography matrix is exactly the same as what we implemented in homework 1. After that, we use RANSAC to get the best fit homography matrix. Then we shift the image border and do the coordinate transformation on the right image to match the left image with the homography matrix in the RANSAC step and do the warping with bilinear interpretation. Last we stitch the warped image and the left image together and remove the border.

Work assignment plan between team members

We finish this homework together.

Additional references

1. sift = cv2.xfeatures2d.SIFT_create() not working even though have contrib installed
2. Introduction to SIFT (Scale-Invariant Feature Transform)
3. Feature Matching
4. How to compute homography matrix H from corresponding points

5. Homography in computer vision explained
6. Random sample consensus (Wikipedia)
7. Adobe Panoramas Dataset