Homework 2: Panorama Stitching

Suiliang Mai, 22521175

1 Problem Setting

For each scene, given two (or multiple) images that captured by a fixed-center camera, our goal is to calculate the homography between these images and generate image mosaics according to these estimated homographies.

2 Method

2.1 Feature Detection, Descriptor, and Matching

Before computing homography, feature detection and description must be done by detector and descriptor. Here we utilizes SIFT as the feature detector, and utilizes both SIFT and local-window feature concatenation as the feature descriptor. In particular, the feature detector returns a bunch of keypoints and the feature descriptor returns features that describe the keypoints. We formulate it by

$$\begin{aligned} \text{kps} &= \texttt{DETECTOR}(I) \\ \text{descs} &= \texttt{DESCRIPTOR}(\text{kps}, I) \end{aligned} \tag{1}$$

where $I \in \mathbb{R}^{H \times W \times 3}$ is the input image.

After obtaining keypoints and their corresponding features of two images, we match the keypoints of the two images by the features. In order to avoid ambiguous matches, the ratio between the distance to the closest neighbor and the distance to the second closest neighbor as mentioned in the class should be adopted. We formulate it by

$$\text{ratio distance} = \frac{\|f_1 - f_2\|}{\|f_1 - f_2'\|} \tag{2}$$

where f_1 is the feature of a keypoint in one image, f_2 and f_2' are the features of the best and second best matching keypoints in the other image. We consider f_1 and f_2 are mathcing point when

$$\text{ratio distance} < \texttt{RATIO} \tag{3}$$

2.2 Homography

Suppose we have matched N keypoints of the two given images, homography is aimed at figuring out the transform matrix H that wraps one image into the other.

RANSAC and DLT algorithms are applied to obtain the largest set of inliers of the given N matching keypoints. Specifically, we iterates M times and each time we randomly choose 4 pairs of matching keypoints (x_i, y_i) and (u_i, v_i) and use them to calculate the homography by direct linear transformation (DLT). We formulate it by

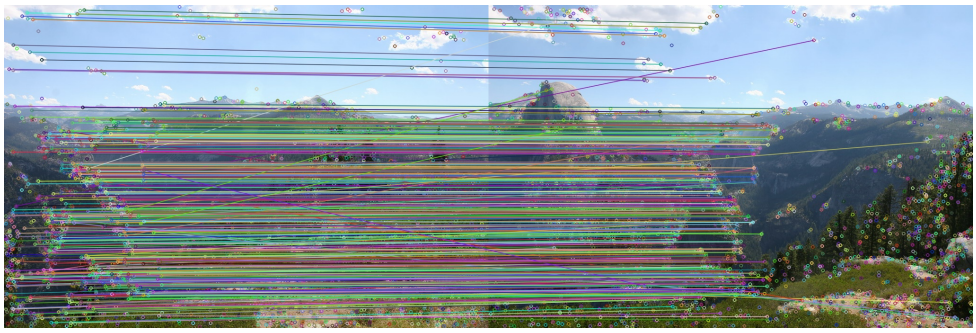


Figure 1: Visualization of matching keypoints of two images in the scene `data2`.

$$\begin{pmatrix} -x_1 & -y_1 & 1 & 0 & 0 & 0 & u_1x_1 & u_1y_1 & u_1 \\ 0 & 0 & 1 & -x_1 & -y_1 & 0 & v_1x_1 & v_1y_1 & v_1 \\ & & & & \cdots & & & & \\ -x_4 & -y_4 & 1 & 0 & 0 & 0 & u_4x_4 & u_4y_4 & u_4 \\ 0 & 0 & 1 & -x_4 & -y_4 & 0 & v_4x_4 & v_4y_4 & v_4 \end{pmatrix} \cdot \begin{pmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{pmatrix} = \mathbf{0} \tag{4}$$

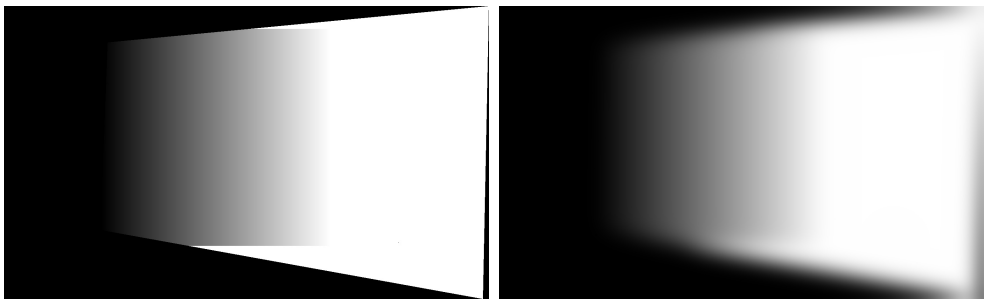
SVD decomposition is leveraged to solve this equation. After obtaining the homography matrix H, we apply this transformation to the matching keypoints and count the number of inliers which fit the model well. After iterating M times, we choose the solution of homography that provides the maximum inlier number. Moreover, M-estimator is used to rectify the homography to get better fitness.

2.3 Image Stitching

When obtaining the best estimation of the homography matrix H, wrapping can be simply computed by multiplying it. To get better quality of the image stitching, we test both linear and multi-band blending techniques to address the overlap part. Specifically, for linear blending, we simply set the blending weight to be both $w = 0.5$.

$$I = w \cdot I_1 + (1 - w) \cdot I_2, \ w = 0.5 \tag{5}$$

For multi-band blending techniques, we first compute a gradient mask M along x-axis as in **??** and we further using gaussian blur to make the edge smoother as **??**.



Gradient Mask Blurred Gradient Mask

Figure 2: Visualization of matching keypoints of two images in the scene `data2`.

With the blending weight mask M and two images I_1, I_2 to blend, we formulate multi-band blending as following steps and get the final blending result I^{BLEND}.

1. Build Gaussian pyramid

$$\begin{aligned} G_0^I &= I,\ I \in \{M, I_1, I_2\} \\ G_{i+1}^I &= \texttt{DOWNSAMPLE}(G_\sigma * G_i^I),\ i = 0, 1, \ldots, K-1 \end{aligned} \tag{6}$$

2. Build Laplacian pyramid

$$\begin{aligned} L_i^I &= G_i^I - \texttt{UPSAMPLE}(G_{i+1}^i),\ i = 0, 1, \ldots, K-1 \\ L_K^I &= G_K^I \end{aligned} \tag{7}$$

3. Blending at each band

$$L_i^{\text{BLEND}} = G_i^M \odot L_i^{I_1} + (1 - G_i^M) \odot L_i^{I_1},\ i = 0, 1, \ldots, K \tag{8}$$

4. Reconstruction

$$\begin{aligned} B_K &= L_K^{\text{BLEND}} \\ B_i &= L^{\text{BLEND}} + \texttt{UPSAMPLE}(B_{i+1}),\ i = K, K-1, \ldots, 0 \\ I^{\text{BLEND}} &= B_0 \end{aligned} \tag{9}$$

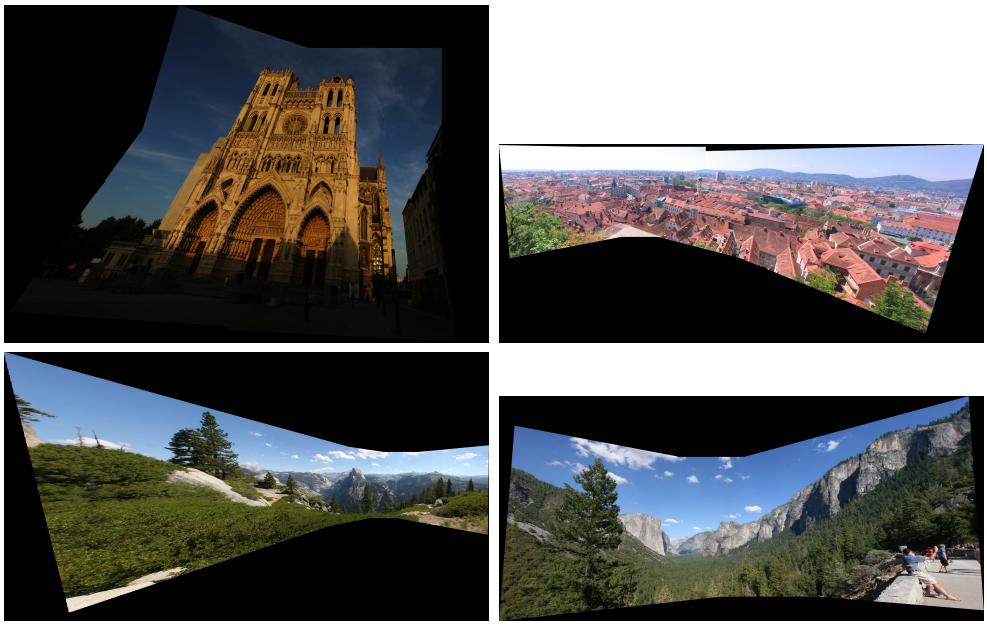


Figure 3: Qualitative results of four scenes using all provided images with SIFT descriptors and multi-band blending algorithm.

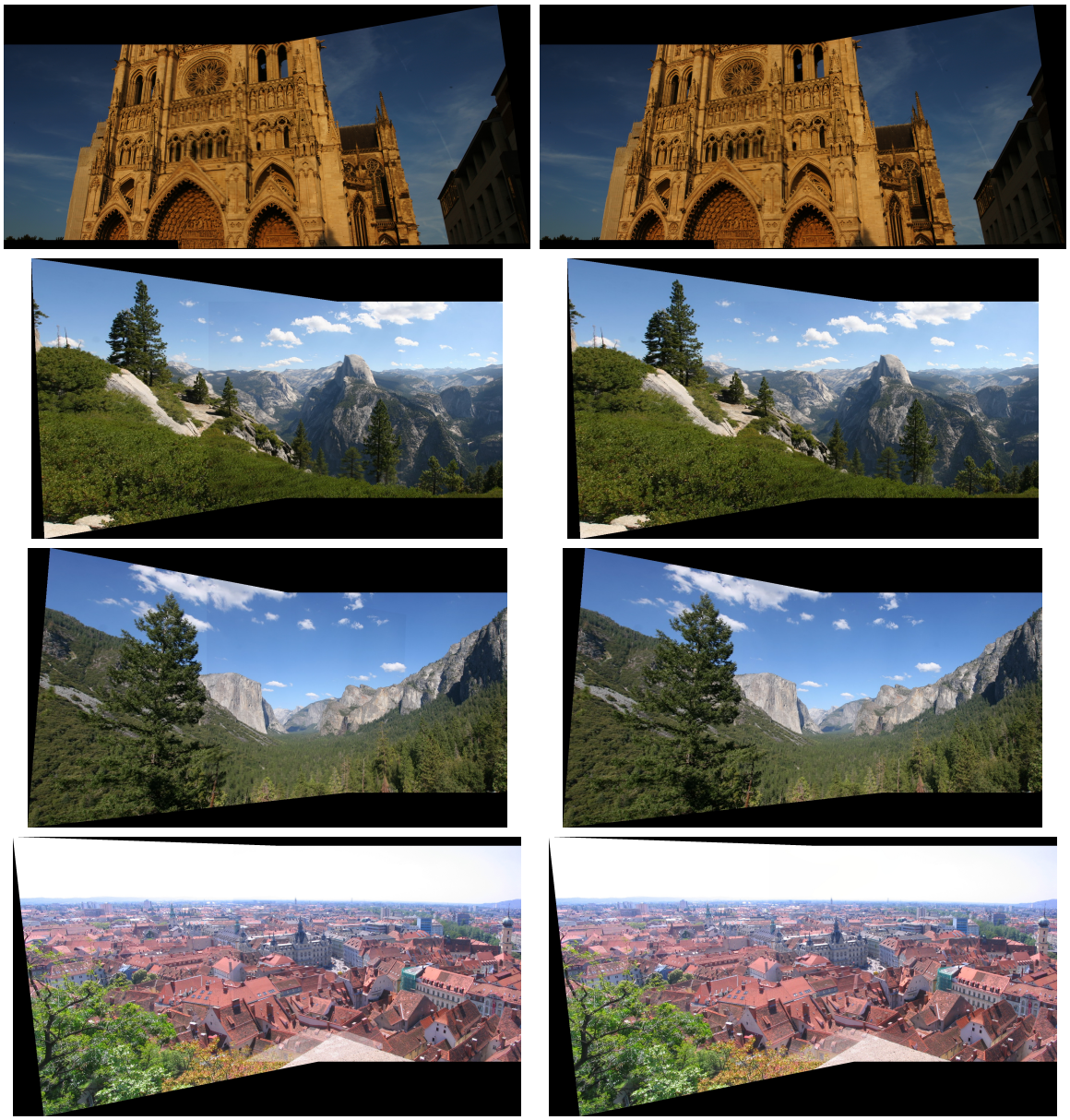
3 Experiments

Our method is tested in four different real captured scenes. The qualitative results are shown in fig. 3. We further compare the difference of blending methods of simple linear blending and multi-band blending, which are shown in fig. 4. Multi-band blending gives more smoother blending results without obvious seams.

Moreover, to prove the quality improvement provided by SIFT descriptor, we compare the qualitative results of SIFT and simple concatenation of pixels in a local window (the size of the window is $W = 11$ in practice), which are shown in fig. 5. To be clear, we use the same detector provided by the SIFT algorithm to obtain the keypoints, but use different descriptors to obtain the features. The result shows that SIFT descriptors make the matching process more accurately and thus result in better homography. Specifically, we test $N = 100$ times using RANSAC and judge the 'correct' approximation of homography by the following criterion,

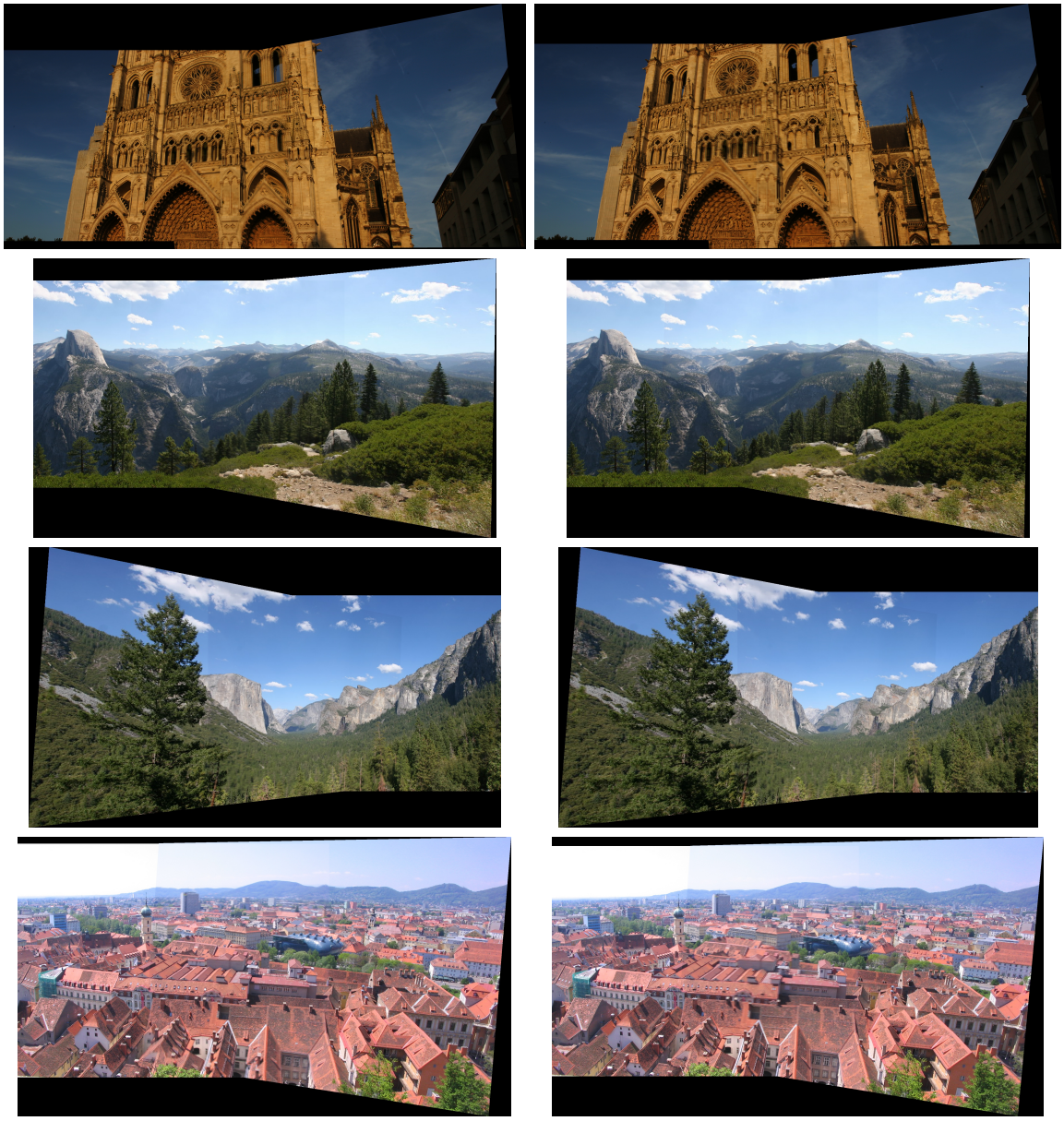
$$\#\text{inlier} > \texttt{THRESHOLD} \text{ and median error} < \texttt{ERROR\_THRESHOLD} \tag{10}$$

where we practically set $\texttt{THRESHOLD} = 20$ and $\texttt{ERROR\_THRESHOLD} = 3$. We show the ratio of correctness in **??**



Linear Blending Multi-band Blending

Figure 4: Comparisons of four scenes using two images with linear blending and multi-band blending algorithm respectively.



Concatenation Descriptor SIFT Descriptor

Figure 5: Comparisons of four scenes using two images with concatenation descriptors and SIFT descriptors respectively. Note that we use the same SIFT detector for keypoints. To make the differences more clearly, we use linear blending with constant weight $w = 0.5$ instead of multi-band blending.

	concatenation	SIFT
data1(`112_1300`, `112_1301`)	46	87
data2(`IMG_488`, `IMG_489`)	63	78
data3(`IMG_675`, `IMG_676`)	61	69
data4(`IMG_7358`, `IMG_7357`)	48	76

Table 1: Quantitative results of different descriptors in four given scenes. We test 100 times using RANSAC and count correctness. SIFT is better than the concatenation algorithm.

4 Discussion

4.1 Order of Image Stitching

When stitching more than 2 images, we need to compose the images pairwise. However, errors due to the estimation of homography and distortions due to multiple applications of transform matrices fail stitching in the order of view movement. A probably better approach is building a binary tree-like structure and let the original images be the leaf node, and then stitching pairwise from bottom to top. Therefore we can control the order of the stitching.

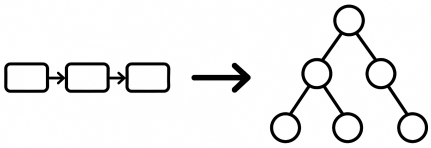
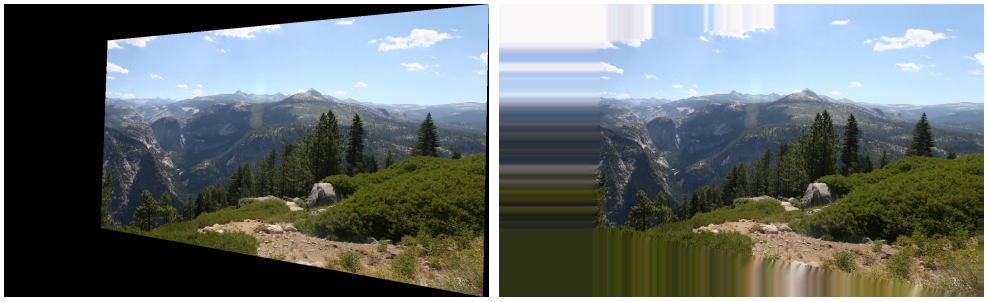


Figure 6: Sketch of the order of image stitching.

4.2 Multi-band Blending

Several troubleshooting is done in order to successfully generate smooth image blending results during the developing process. First of all, a smooth mask is needed as a guidance for blending weights. The basic linear mask is actually a mask with sharp seams, which results in bad blending. Moreover, since the wrapped images are filled with black background where there is no pixel. If we directly pass this image into the multi-band blending algorithm, it will generate more obvious seams. This issue is caused by the reason that the Gaussian blur is affected by the black background pixels. To eliminate this influence, the wrapped images need to fill the black pixels with Navier-Stokes inpainting before compute the Gaussian blur.



Wrapped Image Inpainted Image

Figure 7: Example of wrapped image and inpainted image in the scene `data2`.