Project 1: Myautopano!
Photo Mosaics Implementation

Wenyan Li
Email: wenyanli@umd.edu

***Abstract*—This project mainly focuses on implementing an end-to-end pipeline for image panorama stitching. In order to do so, we first make use of Adaptive Non-maximal Suppression (ANMS) for finding a manageable amount of best feature points(corners) on the given image dataset, then descirbe each feature point with a 64*1 feature vector. By further matching the correspounding features and applying robust homographic perspective projection to project photos from various perspectives onto the same plane, we can blend the projected images and stitch them together.**

I. Introduction

In this project, feature extraction, description and matching are fully implemented for image blending and panorama formation. Works have been done in the following aspects:

1) Adaptive Non-Maximal Suppression(ANMS) method is used for extracting a manageable number of feature points(corners).
2) A 64*1 descriptor vector is extracted for describing each of the best features in the given images. Correspounding feature descriptors are also matched as inlinears by computing SSD.
3) A robust method Random Sampling Concensus (RANSAC) is used for computing the homography used in projective transformation.
4) The computed transformations are applied to images to map them into the reference image and stitch them together as panorama.

II. Feature points detection and Adaptive Non-maximal Suppression (ANMS)

In this part, first, feature points(corners) are detected in the images using harris corner detector. There are plenty of feature points produced in the process. And since we want a manageable number of corners to work with, we find all the local maxima and then for every pixel with a score over the corner threshold, the squared distances between each pair of points are checked and those points with highest score within the radius are selected as the best feature points.
In this way, only the strongest feature points are extracted for further use. The ouput of ANMS is as shown in Fig. 1.

III. Extract Feature Descriptors

This part aims to extract a 64*1 descriptor vector for each of the best features detected in the first section. The algorithm for extracting the feature descriptors is as follows: first a 40*40 window is formed around the corner(feature point), then by bluring and down sampling, a 8*8 patch can be created for describing each corner. One problem that I noticed is that when using Gaussian kernel for blurring the image, the kernel size is important for precisly extract the features. The size of the kernel should not be too large, otherwise, features points that are unrelated will be determined as matched in the following step.The patches are then normalized with mean zero and standeard derivation of one. Now, each of the best feature points is described with a vector, and by later matching the descriptors, we can calculate the homography between images taken from different perspectives.

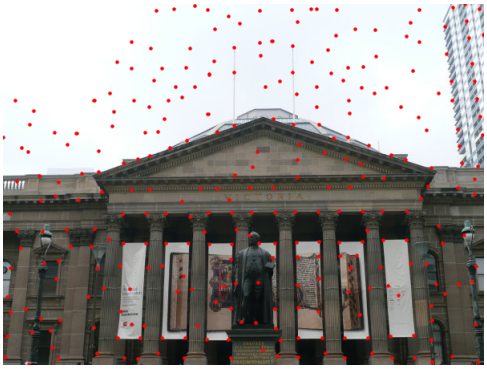

Fig. 1: Detected strongest feature points

IV. Feature Matching

With our interest features all well described, feature matching is needed to perform homography calculating for stitching the images with shared areas. In this step, the SSD between all pairs of 8*8 descriptors in one image to the descriptors in the second image is calculated. Assuming the images given are ordered in sequence, the image in the middle of the sequence is chosen as the reference image, and the feature matching is conducted pairwisely from both ends of the sequence into the middle. For each corner, two nearest neighbors are found in the second image, and we keep the nearest feature as a good match is the ratio(SSD of the nearest and sencond nearest match) under the threshold. In this process, repeat observations are needed for determining a good threshold(which is quite crucial for a good homography). Matched features from the two images are shown in Fig. 2.

V. Computing Homography using RANSAC algorithm

Images that are taken from different perspectives are related to each other by homography. With described features, we

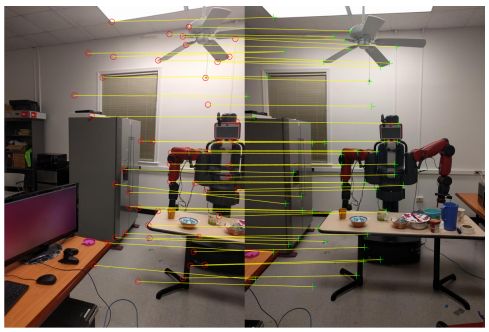

Fig. 2: Feature matching after RANSAC

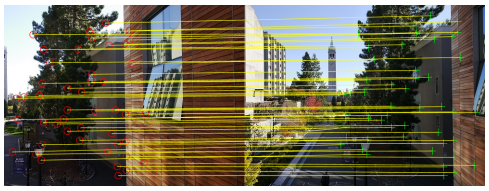

Fig. 3: Matched features with matching vectors

can calculate the correspounding projective transformation. However, some of the feature pairs shown above can skew the transform. RANSAC algorithm is applied in this step to have all the feature points vote for the transform they agree, so the we can trust the transfrom with most votes. In this way, the tranform can be refined by calculating homography using all the trusted matches(inliers). The transformation matrix needed in a panoramic image is a Homography matrix, which contains 8 degrees of freedom. So each time, four pairs of features are used for estimating homography. The final estimated homography can be calculated using all of the inliers once a certain percentage can be obtained. It can be noticed that after applying RANSAC algorithm, the features can be precisely matched, as shown in Fig. 3

VI. Cylindrical Projection

In order to avoid distortion problems at edges after applying homography, cylindrical projection is needed to be used on the images before conducting any transform. According to the following equations, transform can be made from the normal image co-ordinates to cylindrical co-ordinates.

$$x' = ftan(\frac{x - x_c}{f}) + x_c \tag{1}$$

$$y' = \frac{y - y_c}{cos(\frac{x - x_c}{f})} + y_c \tag{2}$$

However, by directly using these equation on the normal image some of the coordinate values of (x', y') map into negative values, which can be left as black pixels. In order to fix this, I calculated the cylindrical projection in a reverse way, i.e. calculating the correspounding pixel corrdinates in the normal image for each pixel in the cylinder projection

(a) Initial

(b) Cylindrical

Fig. 4: Cylindrical projection using reverse calculation

by assuming the images are of same size. In this case, the equations applied for calculating are as follows,

$$x = arctan(\frac{x' - x_c}{f}) + x_c \tag{3}$$

$$y = (y' - y_c)cos(\frac{x - x_c}{f}) + y_c \tag{4}$$

The cylindrical projections are shown in Fig.4.

VII. Stitching images together

The main idea of stitching the images is: by warping images using the homography matrix, we can map them into the reference image and allign the feature points. Then we can create a large image that includes all of the transformed points from all of the images and figure out how to stitch them together. The large panorama image can be initilized by taking limits of the transfromed images and project them into a same projective plane. There are several ways to blend and stitch the images, and I've tried three methods and try to compare which one is better.

A. Stitching by taking the maximum

The simplest way is to take the maximum out of the two images at any point of the resulting canvas. This works fine under the need to stitch only two or three images(take maximum in a chained way), as shown in Fig. 5. However,

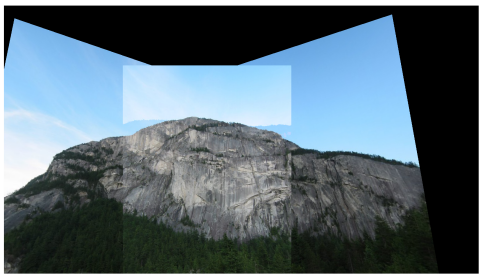

Fig. 5: Stitching by taking maximum

when the number of the images for stitching increases, by using a chained method, i.e. take the maximum pairwisely, the output image got blurred badly. So this method does not work for the panorama which is combined by more than three images.

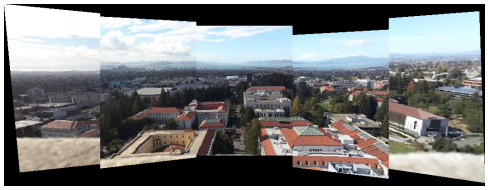

Fig. 6: Stitching by taking corresponding slices of the images(costom set)

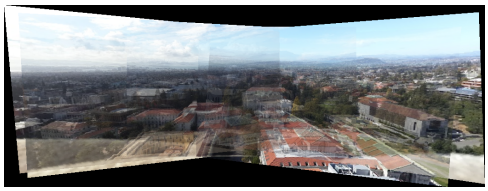

Fig. 7: Stitching by using binary masks(costom set)

B. Stitching by taking relevant image slices

If the series of the images are taken with angles stably changed only in the horizontal or vertical direction. A clear panorama can be created by combining the corresponding columns or rows in the warped images. However, if the images are taken with unstable change or in other directions, the created image may be left with black pixels in some of the areas, as shown in Fig.9.

C. Blending images with masks

The third method is to create a mask which represent the regions of the transformed images in the overall output panorama, by simple assigning weights(taking average) to the overlapping areas, a panorama can be created with multiple input images. In this case, it's still kind of blurred in the overlapping regions and the output is not as clear as the previous one. But the ouput image has a smoother boundary. The panorama produced with this method of the given set3 is shown in Fig. 8. There remains more work that can be done such as using a linear mask or using Laplacian blending with Laplacian stacks, whose output is of better performance with smoothier and seamless transitions.

VIII. Results

The generated panoramas for the two custom sets and three given training sets are shown in below(except for those that are already displayed above).

Compared to images that are taken with only changes in the horizontal direction, to blend the images in the test set is more difficult. The output images can often be skewed. Moreover, it can be noticed that some images in the test sets are mixed with other irrelavant images which require panorama recognition if we want a successful output. For these sets, my panorama creating method fails in a similar way as shown in Fig. 14 unforturnately(Some of the images become so large so I don't put the images here). It might produce better output if image

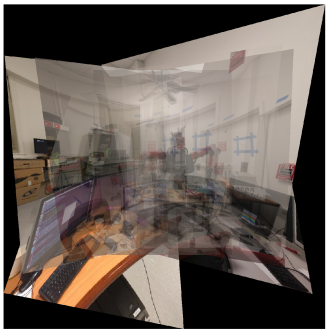

Fig. 8: Stitching by using mask and averaging the overlapped region

Fig. 9: Stitching by using image slices

rectification is processed before estimating homography, and Laplacian blending may help for a smooth output. It can also be noticed that without doing rectification first, the calculation of homography is not precise and often not stable, which results in extremely skewed output image(as shown in Fig.14).

IX. Conclusion

In this project, an end-to-end pipeline for image panorama stitching is implemented. However, while panoramas can be generated with well-captured images, more problems are exposed during the process. It is quite difficult to blend images to get a seamless panorama. A successful blending not only requires precise feature matching and homography calculation, but also proper methods for dealing with the transition between images.

References

[1] Harrison Chau, Robert Karol, *Robust Panoramic Image Stitching*

[2] *Feature Based Panoramic Image Stitching*. Retrieved Febuary 10, 2017, from https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html

Fig. 10: Result of training set 1

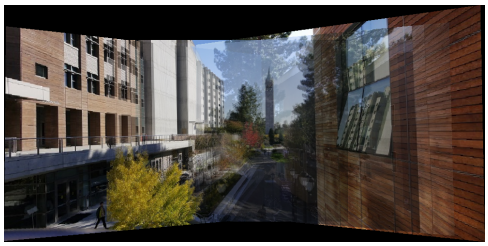

Fig. 11: Result of custom training set 1(mask used)

Fig. 12: Result of custom training set 1(stitching with slices)

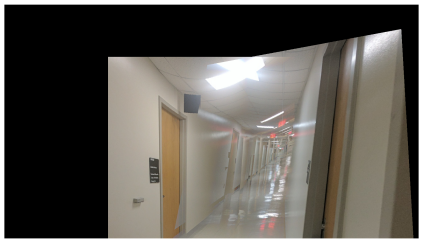

Fig. 13: Result of testset3

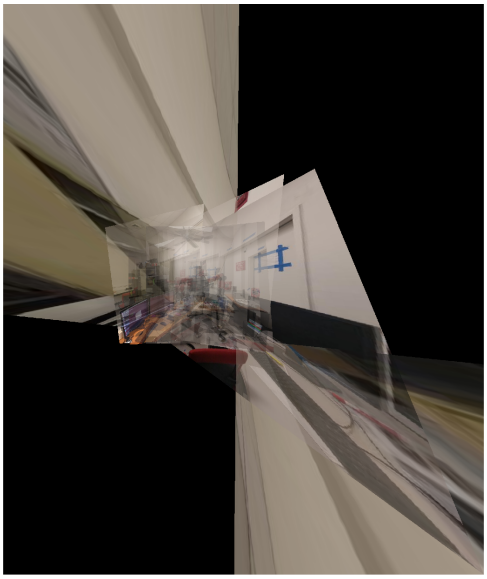

Fig. 14: Failed panorama