Dept. of CST, Tsinghua University

Image Stitching

Project Report

Name	Yuxin Wu
Student No.	2011011271
Class	CST-14
Mail	ppwwyyxxc@gmail.com

Contents

1 Introduction **1**
1.1 Compilation 1
1.2 Run 1
1.3 Examples 2

2 Algorithms **2**
2.1 Feature Detection 2
2.2 Warping 5
2.3 Transformation 7
2.4 Straightening 7
2.5 Blending 8
2.6 Cropping 8

3 More Results **9**

4 References **10**

1 Introduction

1.1 Compilation

Dependencies:

1. gcc >= 4.7
2. GNU make
3. Magick++
4. boost MTL (Matrix Template Library) installed in /usr/include/boost/numeric/. It can be downloaded from http://www.simunova.com/node/145.

Compilation:

```
$ make
```

1.2 Run

Various parameters are saved in `src/config.cfg`. Without special needs, we only have to modify `PANO, TRANS, CROP`.

The program does some extra work to beautify the output if knowing the input pictures were taken by a camera moving in pure translation or pure rotation.

`PANO = 1` tells that the camera moved in pure rotation. A panorama is expected to be the output;

`TRANS = 1` tells that the camera moved in pure translation. Result will be better than the one with `TRANS` unset;

`CROP` decides whether to crop the final image to a rectangular;

Use `./main <file1> <file2> <file3> ...` in the command line to run the program. Output file is `out.png`.

Usually, input images should not exeeds $20(files) \times 1500(pixels) \times 1000(pixels)$, since this program consumes a lot of memory.

The input file names given in the command line need to be well ordered, to make sure the adjacent images can be stitched together.

1.3 Examples

1. TRANS = 1

```
$ ./main ../data/flower/small*
```

2. PANO = 1

```
$ ./main ../data/ground/small*
```

The result is shown in Fig.14.

2 Algorithms

2.1 Feature Detection

Lowe's SIFT algorithm[2] is implemented. The procedure of the algorithm is briefly described as followed. For more details, please refer to the original paper.

1. **Scale Space**

   A Scale Space consisting of $S \times O$ grey images is built. The original image is resized in O different sizes (AKA. octaves), and each is then Gaussian-Blured by S different σ.

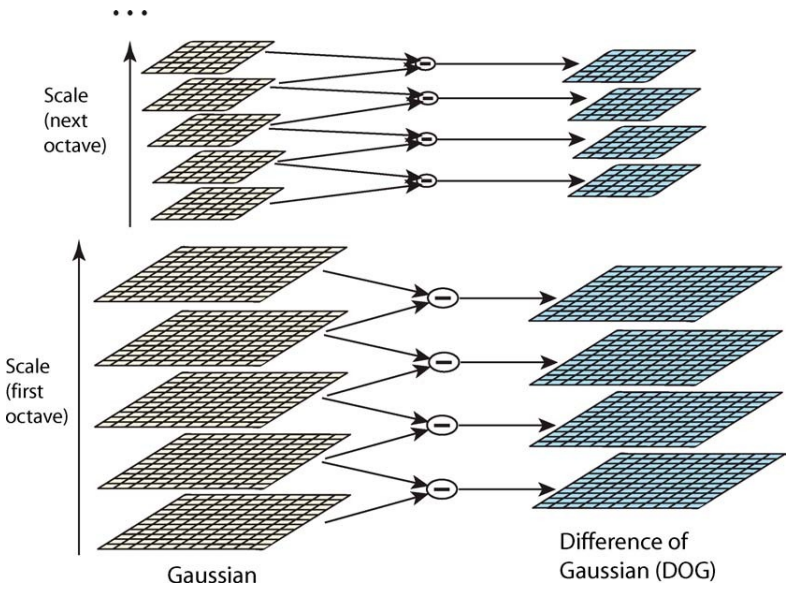

Figure 1: Scale Space and DOG Space

2. **DOG Space**

   In each octave, calculate the differences of every two adjacent blured images, to build a Difference-of-Gaussian Space. Therefore, DOG Space consists of $(S-1) \times O$ grey images. As shown in Fig.1.

3. **Extrema Detection**

   In DOG Space, detect all the minimum and maximum by comparing a pixel with its 26 neighbors in **three directions**: x, y, σ. See Fig.2 and Fig.3

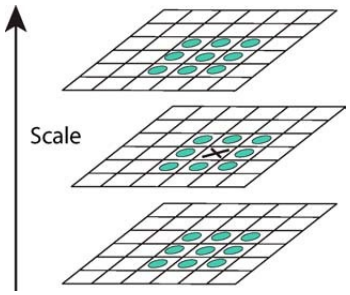

Figure 2: Extrema Detection

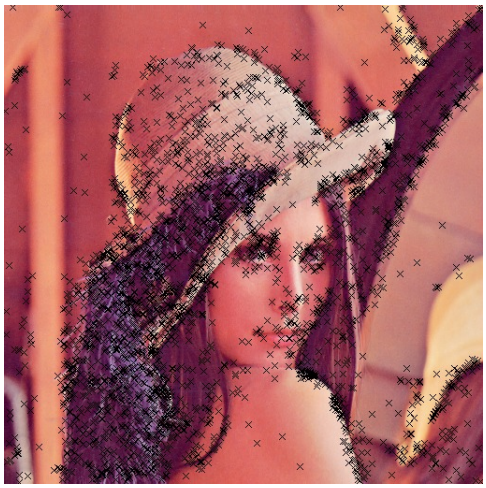

Figure 3: Extrema Example

4. **Keypoint Localization**

   Calculate the offset by **linear interpolation**, to move the extrema to a more accurate location. Then reject the points with **low contrast** or **on the edge**, to get more distinctive features. See Fig.4, Fig.5, Fig.6.

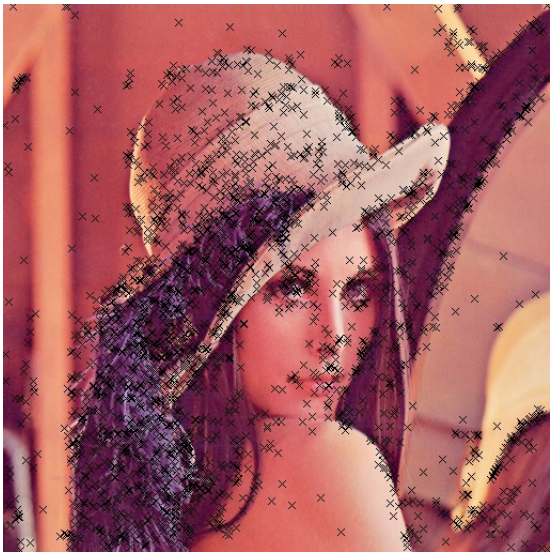

Figure 4: After Localization

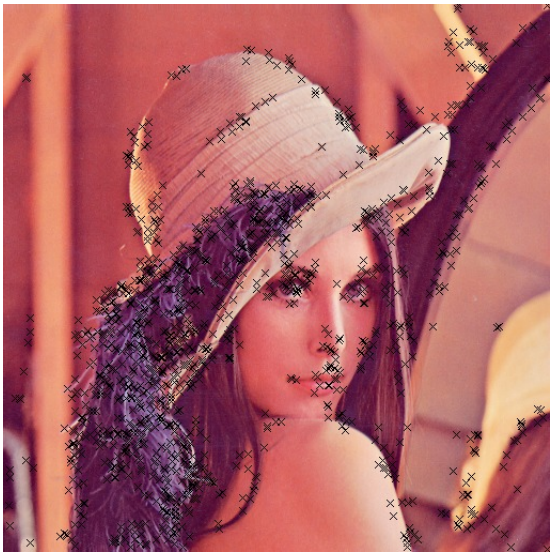

Figure 5: After Rejecting Low Contrast

Figure 6: After Eliminating Edge Point

Figure 7: After Assigning Orientation

5. **Orientation Assignment**

   First, we have to calculate **gradient and orientation** for every point in the Scale Space. For each keypoint detected by the previous procedure, the orientations of

its nearby points will be collected and used to build an **orientation histogram**, weighted by the magnitude of the gradient at each point, as well as an a gaussian kernel centered at the keypoint. The peak in the histogram is chosen to be the main orientation of the keypoint, as shown by the arrows in Fig.7.

6. **Descriptor Representation**

   D. Lowe suggested choosing 16 points around the keypoint, and building orientation histograms for each point, each histogram with 8 different possible values(bins). Therefore the final SIFT feature is a **128-dimensional** array. Since the major orientation of the keypoint is known, by using relative orientation to the keypoint, this feature is roatation-invariant.

7. **Feature Matching**

   Euclidean distance of the 128-dimensional descriptor is the criteria for feature matching between two images. A match is considered not convincing and therefore rejected, if the distances from it to its closest neighbor and second-closest neighbor are similar. The result is shown in Fig.8.

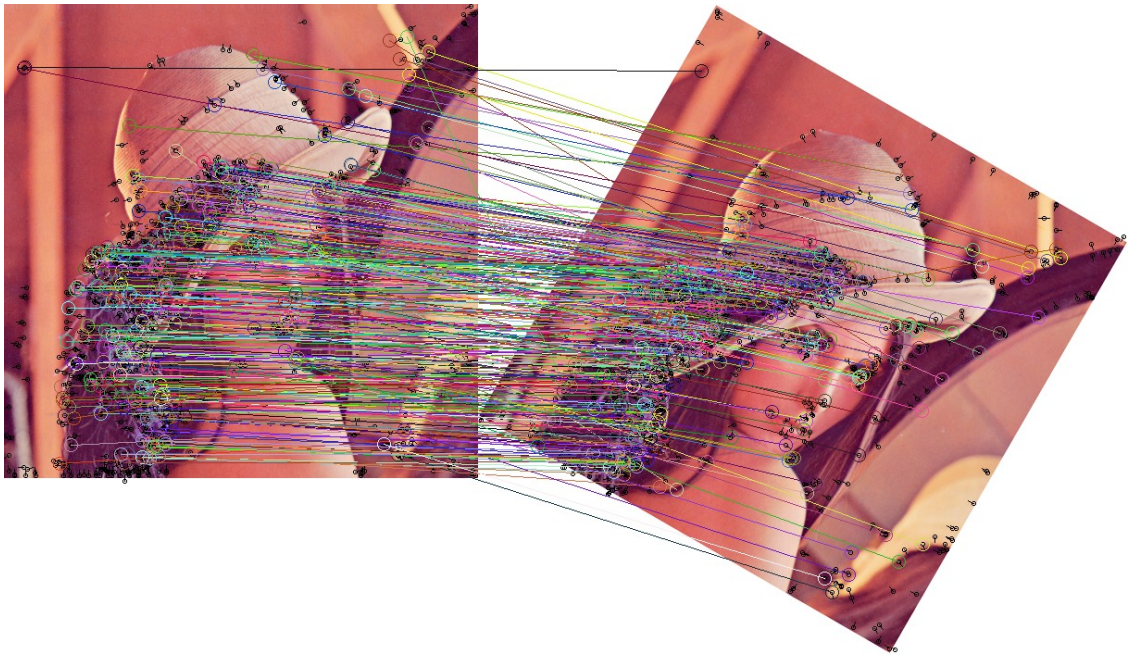

Figure 8: Matching Result

2.2 Warping

A **homography transform** is needed to match the keypoints. But a pure homography transform on a planar works only if the camera moves in pure translation or, moves toward a fixed center.

If a rotational input is given, as most panoramas are built, using homography on a planar leads to **vertical distortion**, such as Fig.9.

Figure 9: Vertical Distortion with Homography on a Planar

Therefore, if the option `TRANS = 0` is set, every input image will be first projected to a cylinder [1] by the following formula:

$$\begin{cases} x' = k \arccos \dfrac{x}{\sqrt{x^2 + z^2}} \\ y' = k \dfrac{y}{\sqrt{x^2 + z^2}} \end{cases}$$

where z is chosen to be $\max\{width, height\}$. The result is in shown in Fig.10.

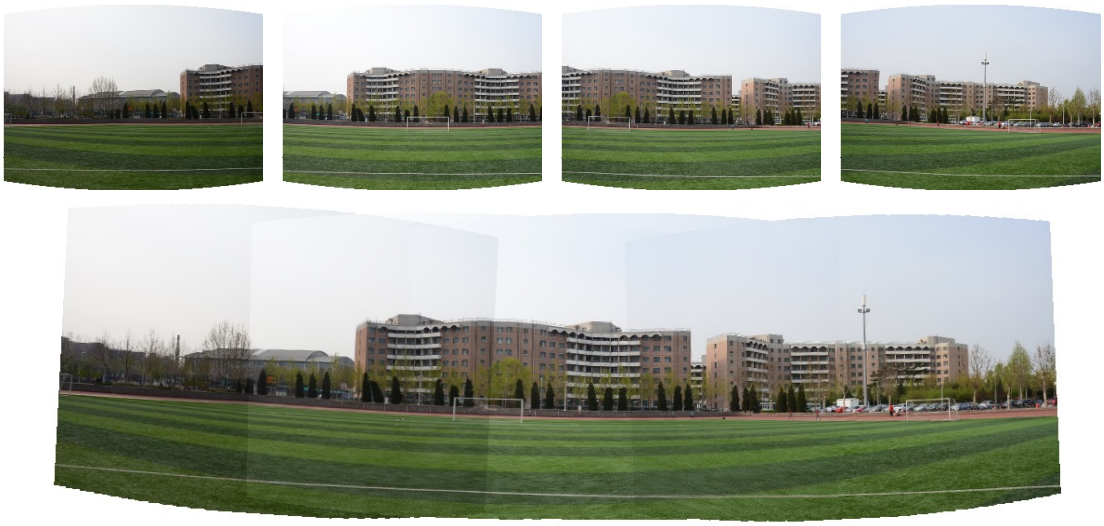

Figure 10: Stitching Result After Projection

[1] A comprehensive app revealing the reason of this projection can be seen at http://graphics.stanford.edu/courses/cs178-12/applets/projection.html

2.3 Transformation

RANSAC (Random Sample Consensus) algorithm[1] is used to calculate a transformation matrix. In every iteration, several matched pairs is randomly chosen to calculate a best-fit transformation matrix, and the number of pairs it fits within all the data is calculated.

I tried three kinds of transformation:

1. Homography I: $H = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix}$
2. Homography II: $H = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ and $\|H\| = 1$
3. Affine: $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$

Given a set of matches, affine and the first kind of homography transformation can be solved by least-squares fitting an over-determined linear system. The second kind of homography transformation can be solved by singular value decomposition. The two homography lead to similar results, which are used when `TRANS = 1`. For warped images, all the three types of transformation work well. In this project, affine transformation is used for generating panoramas.

2.4 Straightening

Figure 11: Bended Panorama

Since the tilt of camera is unknown, the output panorama might be bended as shown in Fig. 11. Instead of using the first image as the pivot, and calculating all the other transformation relative to it, using the image in the middle as the pivot can help reduce the tilt effect. Moreover, when the option `PANO = 1` is set, an automatic straightening process will be executed.

First, a binary search on the tilt factor is performed, aiming to make the result "longer". Second, we try to concatenate the first image after the last one, and then do a global rotation to ensure that the head and tail are aligned horizontally. After this two processes, the result is like Fig.12.

Figure 12: Straightened Panorama

2.5 Blending

The size of the final result is determined immediately after calculating all the transformations. And the pixel value in the result image is calculated by an inverse transformation and interpolation with nearby pixels, in order to reduce alias effect.

As for the overlap effect, the distance from the overlapped pixel to each image center is used to calculate a weighted sum of the pixel value. The result is almost seamless, as shown in Fig.13.

Figure 13: Blended Result

2.6 Cropping

When the option `CROP = 1` is set, the program simply manage to find a rectangular with largest area within the original irregular result.

A classic $O(n \times m)$ algorithm is used, where n, m is the height and width of the original result.

For each row i, the update

$$h[j] = \begin{cases} 0, & if\ i = 0\ or\ A[i][j]\ is\ outside the area \\ h[j] + 1, & otherwise \end{cases}$$
$$r[j] = \max\{k \in [0, m) \cap \mathbf{N} : h[t] \geq h[j], \forall j \leq t \leq k\}$$
$$l[j] = \min\{k \in [0, m) \cap \mathbf{N} : h[t] \geq h[j], \forall k \leq t \leq j\}$$

for every $j \in [0, n)$ can be done in amortized $O(1)$ time, and the corresponding area is $(r[j] - l[j] + 1) \times h[j]$. After the iteration, the rectangular $[i - h[j] + 1, i] \times [l[j], r[j]]$ with maximum area is our result.

The cropped final result is shown in Fig.14

Figure 14: Cropped Result

3 More Results

By adding a polar transformation, we can simply turn the panorama into this:

4 References

[1] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[2] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.