Image Stitching

2017 NTU DigiVFX　Project #2

B03901154 電機三 曾耕森
B03901163 電機三 鄭　煦

Project Overview

This project requires us to rebuild panorama images from a series of photos. This contains several techniques, including cylindrical warping, feature detection, feature matching, blending, etc. The procedure is shown below.

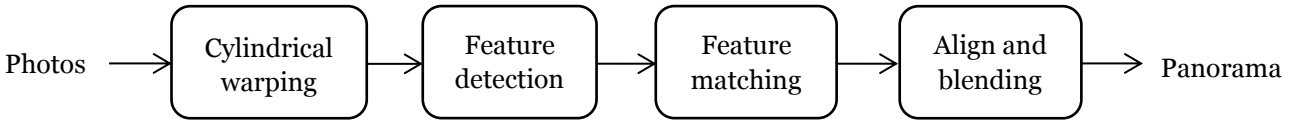

Fig.1　General procedure of creating panorama

We implemented our image stitching program in MATLAB. We use multi-scale oriented patches [1, 2] for our feature detection. Implementation details are shown in section 2. The results and comparison of different algorithms are shown in section 3. Finally, a guide to our program is shown in section 4.

Implementation Details

1. Cylindrical warping

Before starting to build the panorama, we should first warp the images onto a cylindrical plane. This is because the projection planes of each image are not on the same plane. We warp the image according to the formula on the lecture slides:

$$\begin{cases} x' = f \tan^{-1}\frac{x}{f} \\ y' = f\frac{y}{\sqrt{x^2+f^2}} \end{cases}$$

The pixel (x , y) on the original image is warped to the location (x' , y'). f is the focal length of the camera.

- **Focal length calculation**

The cylindrical warping function needs focal length as an input. Instead of using tools such as Autostitch to obtain the focal length, we calculate the focal length in pixel through the following formula

$$f = f_{camera}\frac{w_{image}}{w_{CCD}}$$

where f_{camera} denotes the focal length of the camera in mm, w_{image} denotes the width of the image in pixel, and w_{CCD} denotes the width of the camera CCD in mm. The camera we use is Canon 70D and its CCD size is 22.5 x 15 (mm).

2. Feature point detection

For feature detection, we apply Multi-Scale Oriented Patch [1, 2] to find feature points. First, we form a four-layer pyramid for each image. Then we calculate f_{HM} for each pixel (x , y) in each layer *I* and pick the local maxima in 3x3 area as the corner function value. If the value is larger than the threshold, the point would be recognized as a feature point. The threshold in our implementation is set to 7. We also use adaptive non-maximal suppression (ANMS) to assure that the feature points are spatially well distributed. To get a more precise location of the feature points, we do sub-pixel refinement as written in the paper and make sure the refined location(x , y) satisfies

$$\begin{cases} y > 21 \\ height - y > 21 \\ x > 21 \\ width < 21 \end{cases}$$

so that it can form a 40 x 40 patch in the later procedure. To get the orientation of the patch, we calculate the sub-pixel value of each feature point. The sub-pixel value is obtained through weighted average for each adjacent point. Finally, to get the feature

descriptor of a feature point, we rotate the image according to its orientation and get the 40x40 patch with the location of the feature point (a , b) at the center. Then we do down-sampling by averaging each 5x5 area to obtain a 8x8 descriptor. We choose at most 500 feature points for each layer.
Fig.2 shows an example of feature points extracted by our program.

Fig. 2 An example of the feature points detected by our program.

3. Feature matching

With the feature points extracted, we can now match the features between two images and align them accordingly. This contains two parts: patch matching and image aligning.

- **Patch matching**

We match the features between two images by comparing their patches. For each feature point in an image, we search for the most similar feature point in the other image in the same layer. This is done by calculating the root mean square error between two patches and choosing the one with the smallest error.

- **Image aligning**

After obtaining the matched pairs of features, we can now align two images. We only use translation model as the motion model. That is, assume the images can be aligned by simply moving them. We use RANSAC algorithm to avoid the impact of outliers.

4. Blending

We implement multi-band blending in our program [4]. To do multi-band blending, we have to calculate the Gaussian pyramid of the overlapped area for two images. Once we have the Gaussian pyramids, we can calculate its corresponding Laplacian pyramid through the following formula

$$L_i = G_i - upsample(G_{i+1})$$

where L_i , G_i denotes the Laplacian pyramid and the Gaussian pyramid in i^{th} layer, respectively. The reconstruction of the image is done by expanding and summing each layer in the Laplacian pyramid. Expand refers to upsampling the smaller Laplacian pyramid layer. For example, expand the n^{th} Laplacian pyramid and add it to the $n\text{-}1^{th}$ Laplacian pyramid. Then expand the temporary result and add it to the $n\text{-}2^{th}$ Laplacian pyramid.

Results

In this section, we will show our result images and compares to the result of Autostitch.

(a)

(b)

(c)

(d)

Fig. 3 Result images of Autostitch and our program. The upper ones are the result of Autostitch, and the lower ones are the result of our program.

Our program accurately matches the features and aligns the images, but shows some visible artifacts. For example, our program is not invariant to change in brightness as can be seen in (b), (c). And due to the simplified motion model, the program is sensitive to change of projection center and rotation and sometimes produce visible edges.

Fig. 4 Brightness error. The images are generated by our program and Autostitch, respectively.

Fig. 5 Projection error. The images are generated by our program and Autostitch, respectively.

How to Run Our Program

Our program was written and run on Matlab v.8.6.0 (R2015b) with 64bit Windows 7. To run our program, open Matlab and change to the directory of our code (i.e. Resource/src). We describe the step-by-step procedure as follow:

1. Move the photos into a folder and name the photos by its index according to its position **from left to right** in the final panorama. That is, the leftmost image should be named as "01.jpg", then "02.jpg", etc.
   You can refer to `input_image/` folder for example.
2. Execute image stitching script file in Matlab command line.

   ```
   >> Image_stitching
   ```

3. The program will show a prompt to ask you to enter the folder name and the focal length. Enter the path to your image folder and the focal length. The focal length of our image is 3648 (pixel).

   ```
   >> Enter the path to folder:  ../input_folder
   >> Enter the focal length:  3648
   ```

4. If the image is too large, the program will automatically resize the images. The program might take few minutes to compute the result.
5. The result panorama will be stored in the variable `panorama` and shown in a figure. The result image will be saved to the `result/` folder.

References

[1] Matthew Brown, Richard Szeliski, Simon Winder, *Multi-Scale Oriented Patches*, MSR-TR-2004-133, 2004

[2] Matthew Brown, Richard Szeliski, Simon Winder, *Multi-Image Matching using Multi-Scale Oriented Patches*, CVPR 2005

[3] M. Brown, D. G. Lowe, *Recognising Panoramas*, ICCV 2003

[4] Peter Burt, Edward Adelson, *A Multiresolution Spline with Application to Image Mosaics*, ACM Transactions on Graphics, Vol. 2, No. 4, 1983, pp217-236