2017 Spring Digital Visual Effects
Project #2 — Image Stitching

Team Member: b03901032 Jason Kuo 郭子生, b02901001 S. L. Chung 鍾勝隆
Github: https://github.com/awinder0230/2017-Spring-Digital-Visual-Effect

Introduction:

Image stitching is useful when creating an image with larger field of view (FOV) by stitching a set of images altogether. A common application of image stitching is to create a panorama, which has a wide-horizontal-angle view. In this project, we implemented an end-to-end algorithm, with input as a set of images, and with the output being the panorama of these images.

Submission Files:

- **Resource**
  ./input_image/: a set of input images as well as a file named focal_length.txt
  ./result/: the output panorama of input images
  ./src/: Matlab script codes
  ./report.pdf
  ./readme.txt
- **Artifact**
  ./artifact.jpg

Usage:

1. Clone from here if you do not have our codes.
2. Change directory to input_image, put your input images inside as well as a txt file with the format as following:
   Line 1: <number of input images>
   Line 2: <file name of first input image> <space> <focal length of this input image>
   …
   Line N+1: <file name of N^{th} input image> <space> <focal length of this input image>
   Note that the first input image must be the one that you hope to be the leftmost image in your panorama.
3. Change directory to /src, open main.m with Matlab and run.
4. The output panorama will be generated under directory /result.

Implementation:

1. **Feature Detection**

   a. Get the initial features

   We use the method, Multi-Scale Oriented Patches[1], to extract the feature from the image. The main idea of this method is apply Harris Corner Detection[2] to different scale images of original one. To get the different scale image, use the Gaussian filter to avoid the noise problem when downsampling the image. With different scale images, we calculate the gradient to get the response of each pixel, choosing the pixels as the features which are the local maximal and larger than the threshold.

   b. Refine the features

   After getting a bunch of features, in order not to let the features be too close to each other, we apply Non-Maximal Suppression to make the number of features under certain value, which we set 500. The idea is increasing the the radius and delete the features until there is no two or more features in the same circle whose center is the features.

c. Generate the descriptors

For feature descriptor, we sample a 40 x 40 windows centered at the features and rotated by orientation of the gradient. The windows are sampled in the corresponding scale images of each feature. To reduce the dimension of the descriptor, we divide the window into 5 x 5 sub-windows and calculate the mean of those sub-windows. At the end, we concatenate the means and get a 64 dimension descriptor for each feature. Besides, we choose to abandon the feature whose 40 x 40 window is out of the boundary of the image, because descriptors would be incorrect if we filled them with 0.

2. Feature Matching

To speeding feature matching part, we use the kdtree-based function, knnsearch (k nearest neighbor), in Matlab. Thus, we are able to find the closest descriptors in one images for the other images. To claim the feature pairs are the best matches, we only choose the features which are each other's nearest neighbor as a pair. Moreover, we assure the error of 2-NN is larger than the error of 1-NN, that is, the second nearest neighbor is not too similar to the nearest neighbor.

3. Cylindrical Projection

We re-project the images onto a cylinder. First, We use the software, AutoStitch[3], to obtain the focal length for each image. Afterwards, we re-project the images with their focal length by inverse warping. When doing the inverse warping, we use the bilinear interpolation to do the reconstruction. What needs to re-project is not only the images but also the features.

4. Image Matching and Recognizing Panorama

At this stage, the goal is to figure out all the possible matching images and filter out those not suitable for output panorama. We apply RANSAC algorithm as M. Brown and D. G. Lowe's ICCV 2003 paper[4] did, and filter out images according to their empirical condition. The basic idea of RANSAC is described as following:

Run the following steps k times:

i. draw n samples
ii. fit parameters with these n samples
iii. count the number of inliers

Output the parameter with largest number of inliers

In our case, suppose we have N images, and for each image, it has a feature matching pair with other N-1 images. Also, assume that we have K feature matching pairs between each two images. The parameter we aim to find in RANSAC algorithm is the translation vector (x,y) for features in the candidate image to map on the reference image after translation, so that two image are able to being stitched together correctly. Thus, applying RANSAC algorithm to our condition, we let n = 1 and k = K. The reason for letting n = 1 is that we need only a pair of feature matching pair to decide the parameter — translation vector (x,y) — and the reason for letting k = K is that the number of feature pairs in our case is not much. Thus, we simply run the iteration over each feature matching pair.

In each iteration, we pick a feature matching pair (i.e. x_1,y_1,x_1',y_1') between two images, and compute their translation vector (x,y) such that $x = x_1'-x_1$, $y = y_1'-y_1$. Next, we translate all the other features (x_n , y_n) in the candidate image according to this vector, and compute the distance between each feature pairs after translation.

$$\text{distance} = \text{sqrt}((x_n' - x_n - x)^2 + (y_n' - y_n - y)^2)$$

If the distance is less than a threshold (10 in our case), we consider this feature pair as an inlier. After k iterations, we are able to find the optimal translation vector between these 2 images according to the number of inliers. Select the translation vector which results in largest number of inliers, and compute the ratio (# inliers / total # of feature pairs) to represent the correlation between this candidate image and the reference image.

Therefore, to complete the task of image stitching and recognizing panorama, we simply select one image as reference image in the beginning, and run RANSAC algorithm over other N-1 candidate images, and find the one with largest (# inliers / total # of feature pairs) ratio. If this image also satisfy the condition: # inliers > 5.9 + 0.22 x total # of feature pairs, we know that we could concat these two image together (the actual concatenation is competed in blending function) , and we select this newly selected candidate image as new reference image to run RANSAC algorithm over the remaining N-2 images. The iteration stops either when no images left or no images satisfy the condition # inliers > 5.9 + 0.22 x total # of feature pairs.

Last but not least, we also collect global coordinates of each images after translation so that we are able to compute the overlapping area to stitch images in the next step.

5. Blending

Linear Blending

In this stage, the goal is to actually stitch two selected neighboring images together unnoticeably. We first applied linear weighted blending function in our basic version. The function is designed as following: found the overlapping area of two images, and let the weight decrease from 1 to 0. An illustration of the weighted function is depicted down below. The red one is for the red image on the right side, and the blue one is for the blue image on the left side.

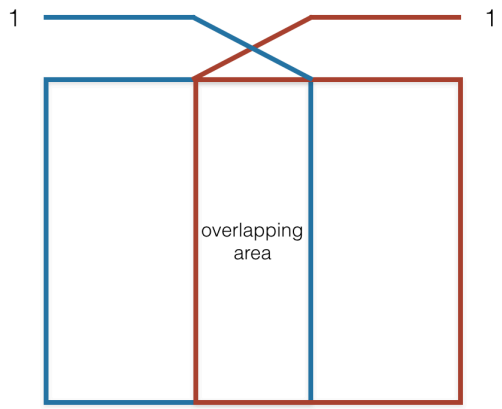

Figure 1. Linear weighted function illustration

Multiband Blending

Multiband Blending[5] is a method which decomposes the image into multi-band frequency and blends each band appropriately. First, for the overlapping parts of two images, we build the gaussian pyramid for each image. Thus, we can calculate the Laplician pyramid, the high frequency part of images, and reconstruct the image with the two Laplician pyramid of two overlapping images. However, the computation complex of multiband Blending is much higher than linear blending method.

6. Seam Carving

After image stitching and blending, the output panorama has irregular boundary due to global shifting (Figure 6 and Figure 7). In order to obtain a rectangular panorama image, we could simply crop a rectangular area of panorama image, but this method may results in a great amount of information lost. Therefore, according to reference [6], we tried to wrap the original image into a rectangular shape, without discarding informations in the image.

This approach is called seam carving. To run seam carving algorithm, we first find the longest boundary segment, which is defined in reference [6]. Next, we select the sub-image area according to the boundary segment. By computing the gradient of each pixel in this sub-image, we are able to obtain a gradient map, then find an optimal seam on this gradient map by dynamic programming. Since we need only to stretch the image in vertical direction, we only have to consider horizontal seams in our case. For more details of seam carving, please refer to reference [6] and [7].

Not to mention that originally we computed the gradient by Matlab default Sobel gradient operator. However, the approach failed as Figure 10 shows. We then compute only dI/dy for gradient map since we only consider horizontal seams, and this modification works well as Figure 11 shows.

Experiments:

1. Input images

Figure 2. Input images set in random order

2. Feature Detection

Figure 3. From left to right: original image, image with detected features, and image with reduced number of detected features by applying ANMS algorithm

3. Feature Matching

Figure 4. Feature matching pairs between two neighboring images

4. Cylindrical Projection

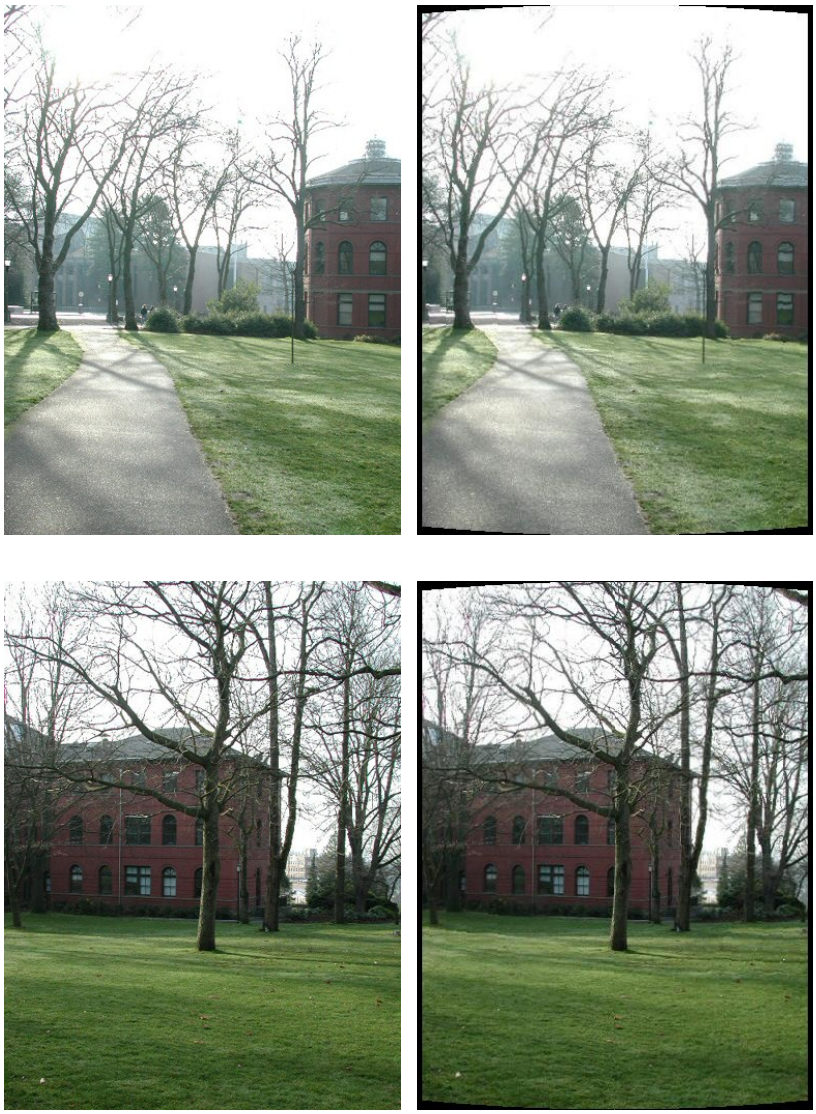

Figure 5. Original image and image after cylindrical projection

5. Image Matching and Blending

Figure 6. Panorama with original input image

Figure 7. Panorama with cylindrical projected input images

Figure 8. Panorama with cylindrical projected input images by linear blending

Figure 9. Panorama with cylindrical projected input images by multiband blending. Note that the leaves of the coconuts are much better than those in Figure 8.

6. Seam Carving

Figure 10. Seam carving result by applying Sobel gradient operator as energy function. Note that there exists a noticeable line which indicates the failure of this approach.

Figure 11. Seam carving result by computing only dI/dy as energy function. The right-upper black corner disappears without losing any information from the original image. Also, there is no noticeable lines in the image even after applying seam carving algorithm.

Figure 11. Input images of artifact

Figure 12. Artifact

Reference:

[1] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. Computer Vision and Pattern Recognition, 2005, p510-517.

[2] Chris Harris, Mike Stephens, A Combined Corner and Edge Detector, 4th Alvey Vision Conference, 1988, p147-151.

[3] AutoStitch: a new dimension in automatic image stitching (http://matthewalunbrown.com/autostitch/autostitch.html)

[4] M. Brown, D. G. Lowe, Recognizing Panoramas, ICCV 2003.

[5] http://cs.haifa.ac.il/hagit/courses/CP/Lectures/CP07_StitchingX4.pdf

[6] K. He, H. Chang, J. Sun, Rectangling Panoramic Images via Warping, SIGGRAPH 2013.

[7] S. Avidan, A. Shamir, Seam Carving for Content-Aware Image Resizing, SIGGRAPH 2007.