VFX Project 2

Image Stitching

B03901056 Fan-Keng Sun, B03901119 Shang-Wei Chen

1 Description

In this project, we construct panoramas from a series of photographs. Run through following feature-based process in the order to create panoramas [1] [2].

- Feature detection: scale invariant feature transform (SIFT) or multi-scale oriented patches (MSOP), with the aid of exhaustive search.
- Feature matching: exhaustive search and Haar's method.
- Projection: cylindrical projection.
- Image stitching: bundle adjustment.
- Blending: multi-band blending.

2 Implementation

2.1 Environment

- Camera: Sony A6000 (lens: Sony SELP 18-105mm G)
- OS: Linux (Archlinux 4.10, Ubuntu 16.04)
- Tools/Libraries: gcc/g++ 6.3.1, OpenCV 3.2 (C++), Boost $\geq$ 1.5, Cmake $\geq$ 3.0, Ceres, Eigen3

2.2 Feature Detection

For feature detection, we have implemented both MSOP and SIFT to gather feature descriptors. MSOP method is easier to be implemented; though, it neglects keypoints that are near the edges, which cannot retreive sufficient amount of pixels for constructing descriptors. SIFT is intricate, but it brings about better performance. Both MSOP and SIFT algorithms are shown in the following paragraphs.

2.3 Feature Matching

For feature matching, we have implemented exhaustive search and Haar wavelet-based hashing to match feature descriptors. Exhaustive search may cost much time to compute matching pairs, and it shows no much better accuracy of matching. Haar wavelet-based hashing transforms descriptor patches to short 3-vector descriptors, and then quantizes each value into 10 overlapping bins in every dimensions (10^3 total entries).

Algorithm 1 MSOP algorithm for feature detection [3]

```
function MsopFeatureDetection(imgs)
    for all img in imgs do
        pyr ← build gaussian pyramid of img
        for lev in range(max_level) do
            kpt[lev] ← find possible keypoints by multi-scale Harris corner detecter
            apply ANMS method for filtering keypoints to be uniform-distributed.
            apply sub-pixel refinement to keypoints
            assign orientation to each keypoint
            desc[lev] ← MSOP feature descriptor
        end for
    end for
end function
```

Algorithm 2 SIFT algorithm for feature detection [4]

```
function SiftFeatureDetection(imgs)
    for all img in imgs do
        G, DoG ← build Gaussian and difference of Gaussian octaves
        for lev in range(max_level) do
            kpts ← local extrema in DoG[lev]
            kpts do discarding low contrast and curvature
            assign orientation to each keypoint
            desc[lev] ← SIFT feature descriptor
        end for
    end for
end function
```

Algorithm 3 Hashing algorithm for feature matching

```
function HaarFeatureMatching(imgs)
    bins ← new hashing buckets
    for all img in imgs do
        for lev in range(max_level) do
            for all patch in img.desc[lev] do
                d ← transform patch using a Haar wavelet
                push this keypoint into bins[d]
            end for
        end for
    end for
    matched ← new container for storing matched keypoint pairs
    for all bin in bins do
        for all kpt_i, kpt_j in bin do
            if pair(kpt_i, kpt_j) has minimum error then
                push pair(kpt_i, kpt_j) into matched
            end if
        end for
    end for
    return matched
end function
```

2.4 Image Projection

Rectilinear projection may cause dramatic distortion when stitching images, so cylindrical projection is applied to fix the problem. In our case, simply doing cylindrical projection cannot smoothen intersection connectivities enough when stitching. To fix the problem, we apply cylindrical projection in both vertical and horizontal direction.

2.5 Image Stitching

We assume that our camera rotates about its optical center. Under this assumption, the images may undergo a special group of homographies. According to [1][2], we parameterise each camera by a rotation vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ and focal length f.This gives pairwise homographies

$$\tilde{\boldsymbol{u}}_i = \boldsymbol{H}_{ij}\tilde{\boldsymbol{u}}_j$$

where

$$\boldsymbol{H}_{ij} = \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{R}_j^T \boldsymbol{K}_j^{-1}$$

$$\tilde{\boldsymbol{u}}_i = s_i[\boldsymbol{u}_i, 1]$$

and $\tilde{\boldsymbol{u}}_i$, $\tilde{\boldsymbol{u}}_j$ are the homogeneous image positions.
The four parameters camera model is defined by:

$$\boldsymbol{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and using the exponential map (in this case a matrix exponential) to transform an axis-angle representation of rotations to a rotation matrix by:

$$\boldsymbol{R}_i = e^{[\boldsymbol{\theta}_i]_\times},\ [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

Since $[\boldsymbol{\theta}_i]_\times$ is a skew symetric matrix, the exponential of it can be shortly formulated as:

$$\boldsymbol{R}_i = \boldsymbol{I} + sin(||\boldsymbol{\theta}_i||_2)[\boldsymbol{\theta}_i]_\times + (1 - cos(||\boldsymbol{\theta}_i||_2))[\boldsymbol{\theta}_i]_\times^2$$

Using this formula, we can perform bundle adjustment by Levenberg-Marquardt algorithm with the aid of Ceres, which can perform automatic derivatives.
Reference from [1] and [2], the error function is defined as:

$$e = \sum_{i=1}^{n} \sum_{j \in \mathcal{I}(i)} \sum_{k \in \mathcal{F}(i,j)} h(\boldsymbol{r}_{ij}^{kl})$$

where n is the number of images, $\mathcal{I}(i)$ is the set of images matching to image i, $\mathcal{F}(i, j)$ is the set of feature matches between images i and j, and $\boldsymbol{r}_{ij}^{kl}$ is the residual between a correspondence of the kth feature in image i and lth feature in image j. We also use the Huber robust error function and set $\sigma = \infty$ during initialization and $\sigma = 2$ pixels for the final solution.

2.6 Blending

Converting the image into high and low frequency images, multi-band blending shows a good recovery for edge connection. The algorithm is shown below.

Algorithm 4 Multi-band blending algorithm

$imgs \leftarrow$ images after warping perspective
function MULTIBANDBLENDING($imgs$)
  $w\_origin \leftarrow$ an array of linearly weighted function for each images
  $output, w\_sum \leftarrow$ blank image arrays
  for i in range($imgs.size$) **do**
    for all pixel p in $w\_origin[i]$ **do**
      if $w\_origin[i].p == \arg\max_n\{w\_origin[n].p\}$ **then**
        $w\_max[i].p \leftarrow 1$
      else
        $w\_max[i].p \leftarrow 0$
      end if
    end for
    $band[i] \leftarrow$ split $imgs[i]$ into different bands
    $w[i] \leftarrow$ compute blurred weight function corresponding to different bands
  end for
  for all $layer$ in range($band\_num$) **do**
    for i in range($imgs.size$) **do**
      $output[layer] \leftarrow output[layer] + band[i][layer] * w[i][layer]$
      $w\_sum[layer] \leftarrow w\_sum[layer] + w[i][layer]$
    end for
    $output[layer] \leftarrow output[layer]/w\_sum[layer]$
  end for
  $result \leftarrow$ merge layers in $output$
  return $result$
end function

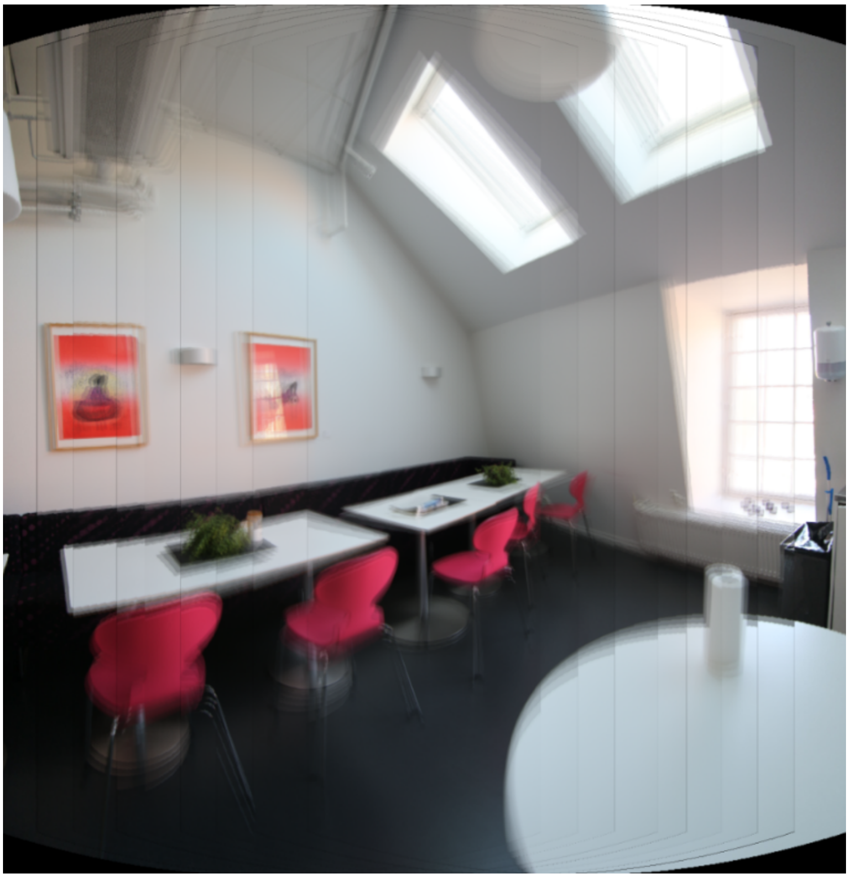
3 Reference

[1] M. Brown and D. G. Lowe. Recognising panoramas. *International Conference on Computer Vision*, 2003.

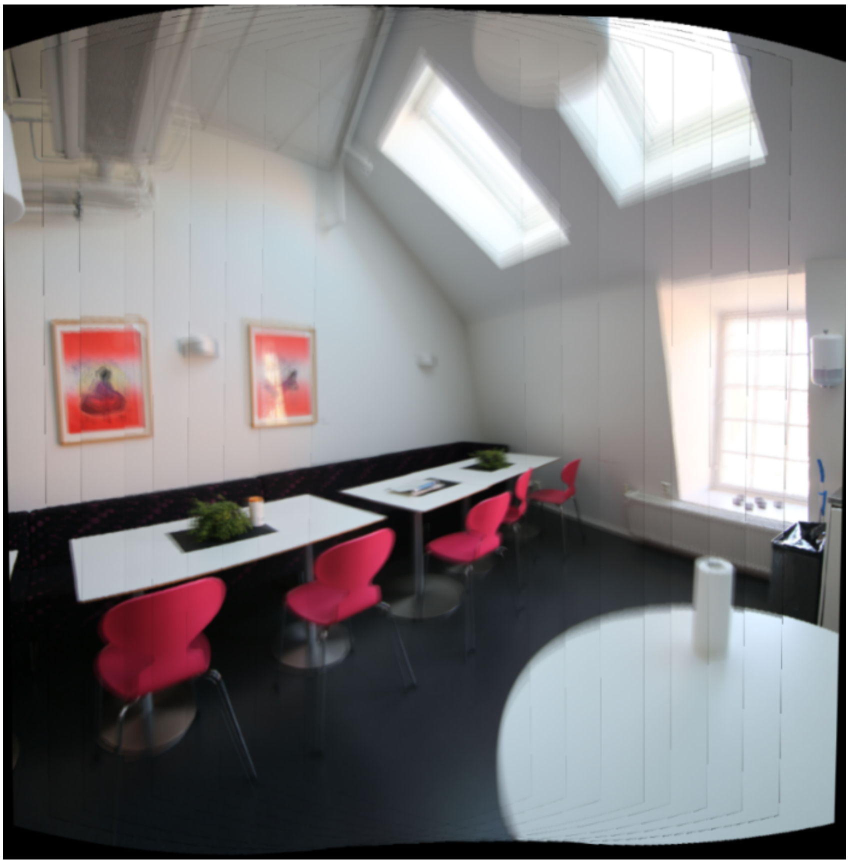
[2] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 2007.

[3] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-scale oriented patches. *MSR-TR*, (133), 2004.

[4] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.



(a) with only translation



(b) with bundle adjustment

Figure 1: Results of bundle adjustment with average blending

(a) with average blending

(b) with multi-band blending

Figure 2: Results of multi-band blending compare to average blending.