CS172 Computer Vision I: Panorama Stitching

Ziqi Gao
2018533193
gaozq@shanghaitech.edu.cn

Abstract

In this Homework, an image stitching task is performed. Three pairs of images taken at Shanghaitech are stitched into one panorama in the end. It includes using existing library to extract SURF keypoints and descriptors from images, manually finding good matches between images, implementing RANSAC to calculate homography, and stitching warped images together with the help of cv2.warpPerspective().

1. Problem Formulation

Image alignment is an important and challenging task in computer vision, and one of its application is panorama stitching. Traditional ways of Image alignment includes pixel-based registration, feature-based registration, global registration, etc. In this paper, an robust feature-based alignment is implemented to construct a panorama of ShanghaiTech's library. Main methods used in this homework include using existing library to extract SURF keypoints and descriptors from images, manually finding good matches between images, implementing RANSAC to calculate homography, and stitching warped images together with the help of cv2.warpPerspective().

2. Algorithm

2.1. Feature Extraction

There are several feature extraction techniques in computer vision, including Harris Corner Detection, Shi-Tomasi Corner Detector, BRIEF (Binary Robust Independent Elementary Features), SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), etc. In this homework, SURF is used for feature extraction, which is a speed-up version of SIFT.

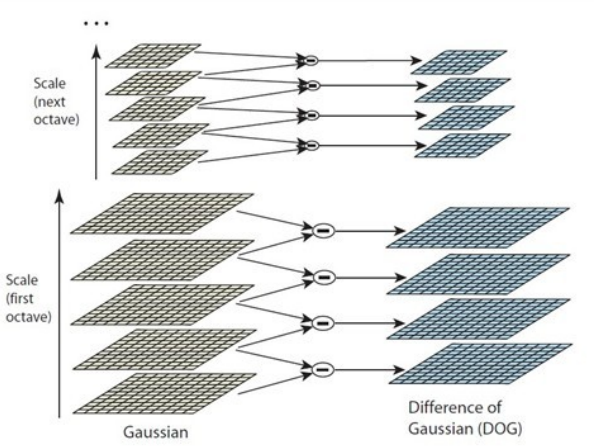

Figure 1. Gaussian paryamid generation in SIFT

2.1.1 Local Feature Detection and Description - SIFT

To be specific, BoF is based on Scale-Invariant Feature Transform (SIFT) for local feature detection and description, which extracts extracting distinctive features invariant to image scale and rotation. Following are the major stages of computation used to generate the set of image features[1]:

- Scale-space extrema detection: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
- Keypoint localization: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
- Orientation assignment: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location

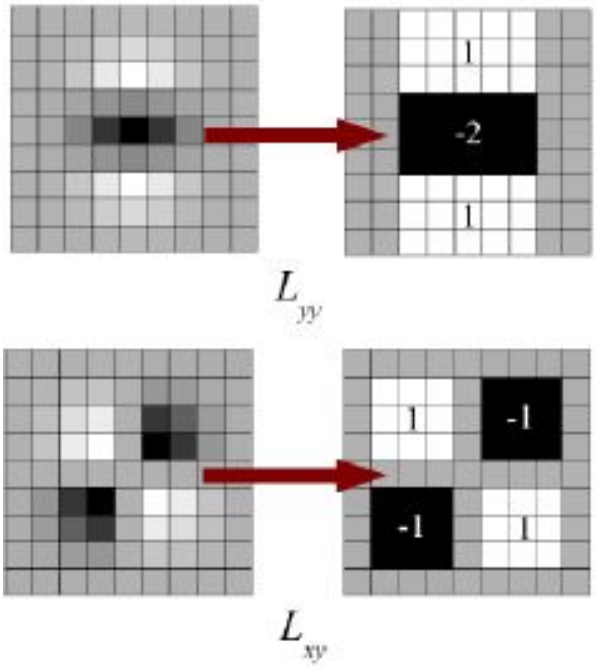

Figure 2. SURF's approximation of LoG

for each feature, thereby providing invariance to these transformations.

- Keypoint descriptor: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

In implementation, a SIFT descriptor is a 128-d vector for each keypoint.

2.1.2 Speeded-Up Robust Features - SURF

In SIFT, Difference of Gaussian is used to approximate Laplacian of Gaussian for finding scale-space, but SURF approximates LoG with Box Filter. This improve the efficiency notably, since convolution with box filter can be calculated easily with the help of integral images and allows parallel computing. A demonstration is shown in Figure 2.

2.2. Find Good Matching Among Features

SURF translates the information in patched-pixels level into descriptors, some vectors that can be used for comparing among keypoints. In this homework, a brute force feature matching is implemented. Its main idea is to compare each keypoint in an image with every keypoints in the other image, and thus find the best corresponding point for each keypoint in the first image. The comparison of 2 different matches is conducted using the L2-norm (Euclidean Distance) of the difference of two keypoints' corresponding descriptors in 2 images. Also, only good matches are preserved. 'Good' in this context means distinctive, ensured by applying ratio test on the first closet keypoint and the second closest keypoint. If their ratio is higher than a given threshold, we include that matching to good matchings.

2.3. Homography Calculation with RANSAC

2.3.1 Homography

Perspective transform, i.e. mapping between any two projection planes with the same center of projection is called Homography, which is represented as 3x3 matrix in homogenous coordinates as followed.

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Using cross product, we can eliminate $lambda$ in the quation. There are 8 parameters in H, so we need at least 4 pairs of points for solving it. After simplification, we can get

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'*x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'*x_1 & -y_1'y_1 & -y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'*x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'*x_n & -y_n'y_n & -y_n' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

n is the number of pairs we use. This equation defines a least square problem, thus homography matrix H can be found.

2.3.2 Finding good homography using RANSAC

Instead of working on individual features in 2.2, in this part we work with all features to find good matching between 2 images using RANSAC. Its basic procedure is as below.

Repeat N times:

- Select a random set of good feature matches (4 pairs);
- Fitting them into an homography (solving the equation above);
- Compute inliers: apply the transformation to all good features and compute the Euclidean distance between matching points after the transformation, $\|pi', Hp_i\| <$ THRESHOLD;
- Count number of inliers and if it is sufficiently large, refit all inliers into the homography.

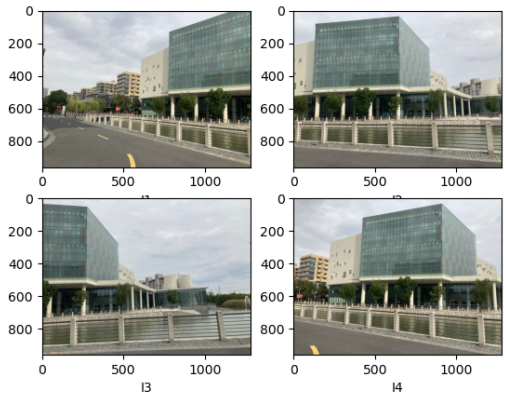

Figure 3. Original images

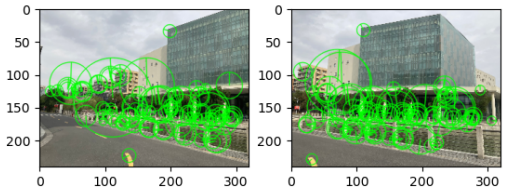

Figure 4. Feature extraction using SURF

2.4. Stitching 4 images into a panorama

We first warp the images using cv2.warpPerspective() is used for tranforming an image into another image's perspective of view. The transform matrix M is the homography matrix H we compute before. After that, some translation offsets need to be consider to stitching them together, where left stitch and right stitch make a small difference in implementation. By mixing left stitch and right stitch, a panorama will be formed.

3. Implementation and Result

3.1. Code Repository

- stitching.py: a Python file containing implementation of stitching.
- matching.py: a Python file containing implementations of matchers.
- lib: a repository containing the 4 original images.
- result: a repository containing all of the result images shown in this paper.

3.2. How To Run

python stitching.py

3.3. Implementation and result

3.3.1 Original images

The 4 original images are shown in Figure 3.

3.3.2 Feature extraction using SURF

Core code in this part is:
surf = cv2.xfeatures2d.SURF_create()

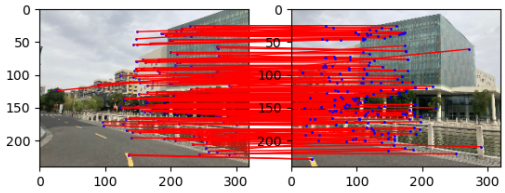

Figure 5. Good Feature Matchings

self.kp1, self.des1 = surf.detectAndCompute(self.img1, None)

For better visualization, only 100 keypoints for each image is shown in Figure 4.

3.3.3 Feature Matching

I choose 0.75 as the threshold ratio here according to the table in ppt and visualize good matches using matplotlib.pyplot.connectionpatch to draw line between 2 points on 2 images. The center coordinates are extracted from the SURF keypoints' pt. The result for compressed image1 and image2 with size (360, 240) is shown in Figure 5.

3.3.4 Panorama stitching

I choose the third image as the center image, i.e. not warped during the whole stitching. I first do left shift to I1 in accordance with H12 and warp it together with I2 into I12, whose result is shown in Figure 6. Then do the same to

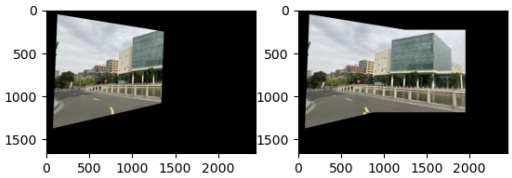

Figure 6. Warped I1 and Merged I12

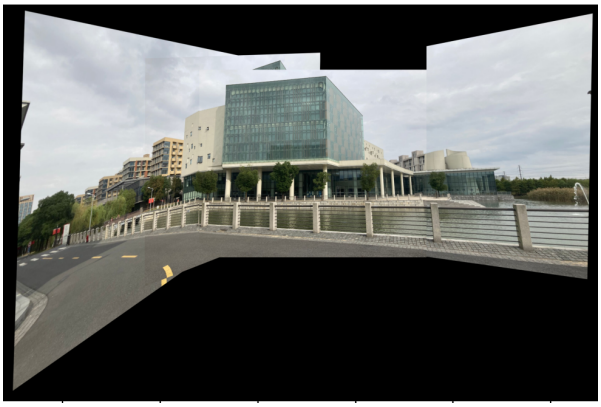

Figure 7. The Panorama

I12 and I3 and get I123. After that, I do a right shift from I4 to I123's perspective and get the final result. The final result is shown in Figure 7.