Image Mosaicing

Chicheng Zhang
zhang.chic@husky.neu.edu

Abstract

In this report, we present a framework for image mosaicing by estimating a homography between corresponding corner features. In this framework we apply a Harris corner detector to find corners in two images, automatically find corresponding features, estimate a homography between the two images, and warp one image into the coordinate system of the second one to produce a mosaic containing the union of all pixels in the two images. Our experimental results demonstrate that our framework has good performance.

1. Algorithm Description

In the first phase of our approach, we convert two RGB images to grayscale due to the data reduction and simplicity. Next, apply Harris corner detector to both images. Given two set of corners from the two images, we compute normalized cross correlation (NCC) of image patches centered at each corner and choose potential corner matches by finding pair of corners (one from each image) such that they have the highest NCC value. Since these correspondences are likely to have many errors, we should use RANSAC to robustly estimate the homography from the noisy correspondences. Finally, using the homography, we warp one image onto the other one, blending overlapping pixels together to create a single image (Figure 1).

2. Experiments and Parameters

The performance of our framework mainly depends on the parameters we used in the stages shown in part 1. Hence, we will give a detailed description of the parameter selection and put a reasonable effort to estimate the best possible parameters.

Figure 1: Image mosaicing example (DanaHallWay2).

2.1. Detecting Harris Corners

For detecting Harris corners, we first need to compute Harris R function

$$R = (I_x^2 I_y^2 - I_{xy}^2) - k(I_x^2 + I_y^2)^2$$

over the image, and then do non-maximum suppression to get a sparse set of corner features. As for Harris R function computing, we first use derivative operators $G_x = [-1 \quad 0 \quad 1]$ and $G_y = [-1 \quad 0 \quad 1]^T$ to get I_x^2, I_y^2 and I_{xy} of every pixel. Before computing R function, we use a Gaussian window function ($\sigma = 1.4$) to avoid the effect of noise. Then we implement our non-maximum suppression by choosing a threshold (0.01$\times$max(R)). See Figure 2.

Figure 2: Detecting Harris corners.

2.2. Computing Normalized Cross Correlation

In this stage, we first remove all key points near the boundary. Then we choose a 5×5 image patch centered at each corner and reshape it as a 25×1 feature descriptor. To make it partially invariant to illumination changes, we normalized each descriptor by using

$$I[n] = \frac{I[n] - \text{mean}(I)}{\text{std}(I)}, \quad n = 1, \ldots, 25$$

where I is the feature descriptor.

We compute normalized cross correlation using

$$NCC = \frac{\sum_{i=1}^{25} x[i]y[i]}{\sqrt{\sum_{i=1}^{25} x^2[i] \sum_{i=1}^{25} y^2[i]}}$$

where x is one of the descriptors of the first image and y is one of the descriptors of the second image.

Finally we choosing pair of corners such that they have the highest NCC value. Besides, we also set a threshold to keep only matches that have a large NCC score. See Figure 3.

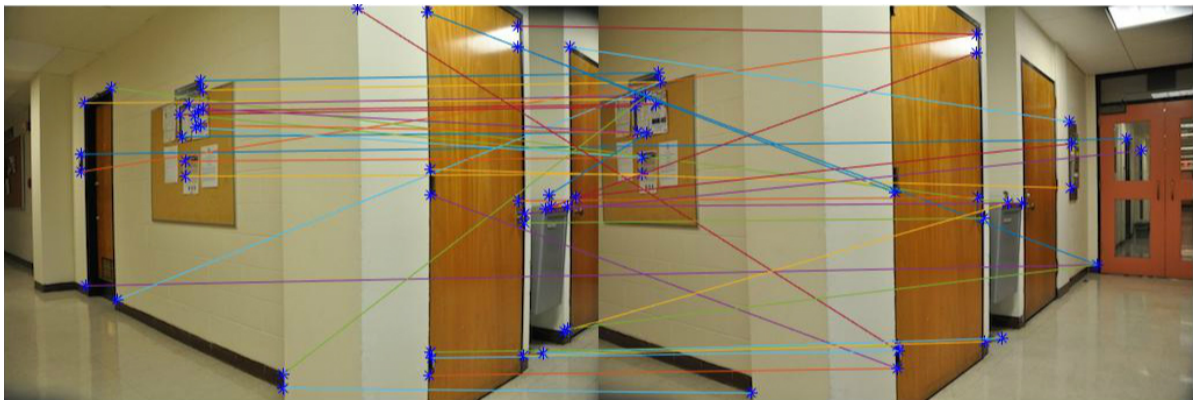

Figure 3: Computing NCC.

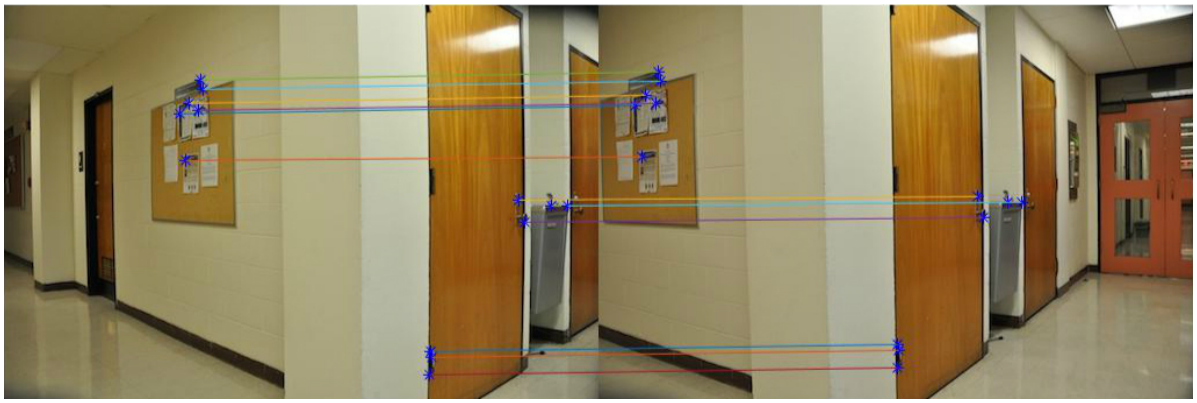

Figure 4: Estimating the homography.

2.3. Estimating the homography

Note that these correspondences obtained in last stage are likely to have many errors. We can use RANSAC to robustly estimate the homography from the noisy correspondences:

- Repeatedly sample 4 points needed to estimate a homography.
- Compute a homography from these four points.
- Map all points using the homagraphy and comparing distances between predicted and observed locations to determine the number of inliers.
- Compute a least-squares homgraphy from all the inliers in the largest set of inliers.

As for computing homography matrix H from the four corresponding points (p_1, p_1'), (p_2, p_2'), (p_3, p_3'), (p_4, p_4'), we can write each pair of points as

$$p_i = \begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & x_i x_i' & y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & x_i y_i' & y_i y_i' \end{pmatrix}$$

Then stack them into a matrix P to compute:

$$PH = D$$

where

$$P = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1 x_1' & y_1 x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1 y_1' & y_1 y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2 x_2' & y_2 x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2 y_2' & y_2 y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3 x_3' & y_3 x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3 y_3' & y_3 y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & x_4 x_4' & y_4 x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & x_4 y_4' & y_4 y_4' \end{pmatrix}, \quad H = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{pmatrix}, \quad D = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{pmatrix}$$

Once we get your homography matrix H, you can compute the projected coordinates $p(x', y')$ of any point $p(x, y)$ by using

$$\begin{pmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1}$$

Then we can map all points and compare the distances between predicted and observed locations, from which we can choose a best homography matrix.

2.4. Blending Images

The final step is to warp one image onto the other one, blending overlapping pixels together to create a single image. The steps are as follows:

- Determine how big to make the final output image so that it contains the union of all pixels in the two images.
- Copy the image that does not have to be warped into the appropriate location in the output.
- Warp the other image into the output image based on the estimated homography.

Figure 5: Blending images.

3. Conclusions

This report introduces a simple framework for image mosaicing. We give many details about each stages and recommend specific value for these parameters. The experimental results demonstrate that our framework has good performance.

For some other example of image mosaicing, see Figure 6.

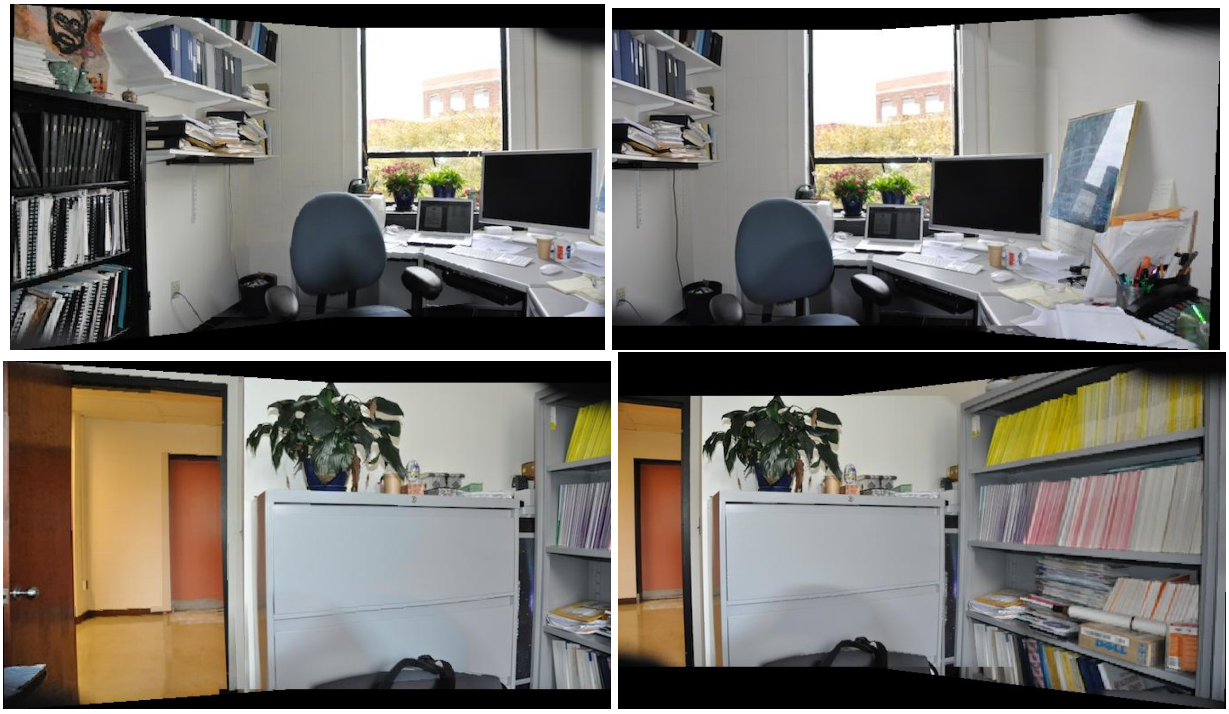

Figure 6: Image mosaicing of DanaOffice (part)

Figure 7: Image mosaicing of DanaHallWay1