VFX Image Stitching Project Report

廖婕吟(R09922136) r09922136@csie.ntu.edu.tw
鄒汶橙(R09922131) r09922131@csie.ntu.edu.tw

Project Description

Image stitching is a technique to combine a set of images into a larger image by registering, warping, resampling and blending them together. A popular application for image stitching is creation of panoramas. In this project, we implement a feature-based method including feature detection, feature matching, image matching ,image blending and bundle adjustment to create a panorama.

Algorithms and Implementation Details

I. Cylindrical projection

The image is projected to the cylindrical coordinates, so that the subsequent image matching has only translation relations.

1. Use AutoStitch software to get the approximate focal length of each image.
2. Refer to the cylindrical coordinate formula in the course slide and use inverse warping to find the projection image.

II. Feature Detection

We used the **Harris corner detector** method to find the features

1. Do a slight Gaussian Blur on the source image
2. Compute x and y derivatives of image to get I_x, I_y

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

3. Compute products of derivatives at every pixel to get I_{x^2}, I_{y^2} and I_{xy}

$$I_{x^2} = I_x \cdot I_x \quad I_{y^2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

4. Compute the sums of the products of derivatives at each pixel using Gaussian Blur

$$S_{x^2} = G_{\sigma'} * I_{x^2} \quad S_{y^2} = G_{\sigma'} * I_{y^2} \quad S_{xy} = G_{\sigma'} * I_{xy}$$

5. Define the matrix M at each pixel

$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

6. Compute the response R of the detector at each pixel

$$R = det(M) - k(trace(M))^2$$

7. Threshold on value of R; compute nonmax suppression.

III. Feature Descriptor

After finding the features in the previous step, we must describe features to find similar features in different images. We use the **SIFT** descriptor (local descriptor) to find the orientation of each point

1. For each feature points, we use a 16×16 window to describe it
2. There are sixteen 4×4 subwindows in a 16×16 window, and for each subwindow, use 16 pixels to describe it with each pixel voting a direction dividing by 8 bins. It implies that one feature point can be described by a vector with 128 dimensions.
3. After establishing a vector(so called descriptor), normalize it. However, if there exist any element ≥ 0.2 in a descriptor, set it to 0.2 and normalize the descriptor again.
4. There may exists many feature point in one graph, therefore, we set a constraint that filters the pixel which is too close to the border to avoid mismatching effect.
5. Moreover, we split one graph into left and right, and use two descriptors to let the matching more precise.
6. It is worth mentioning that we didn't implement the rotate-invariant part, because we have a assumption that our data won't have rotation based on using tripod.

IV. Feature Matching

Feature matching is to find the corresponding relationship between the feature points in two images. We directly use the Brute-force search to find the smallest distance(l2-norm) between the feature point vectors in different images as a corresponding relationship

V. Image Matching

Use the feature matching obtained in the previous step to calculate how much translation must be performed between the images to stitch the images together. To reduce the error caused by feature mismatching, we use the **RANSAC** algorithm. We assume that the correct rate of feature matching is p and the desired correct rate is P. Then the iterations k can determine from the following function.

$$k = \frac{\log(1-P)}{\log(1-p^n)}$$

1. Randomly choose a subset of data points to fit model (a sample), here we sample n match of feature matching corresponding to the two pictures (the smallest sample that can produce translation)

2. We will count the number of inlier points c from the transform which we sampled from the feature matching set of the images.

3. Repeat for k iterations, the feature matching sample which have the largest number c takes into account the translation correspondence between the two pictures.

VI. Image Blending

In the previous step, after calculating the translation relationship that stitches the images together, the overlapped part must be processed to avoid obvious boundaries. The method we use is **linear blending**. That is, in the overlapping area, pixels will have a smaller weight value while it is closer to the boundary. In this way, we can obtain a smooth panoramic image.

VII. Bundle Adjustment

Since the best corresponding position of each picture will drift up and down, the positions of height that the first picture and the last picture are different. We use the `cv2.warpPerspective` of cv2 module to wrap the image after image blending.

VIII. Implementation details (parameters)

fuction	parameter	value
`Compute_Response`	`kernel`	5
	`sigma`	3
	`k`	0.04
`get_local_max_R`	`rthres`	0.06
`orientation`	`ksize`	9
`descriptor`	`up`	15
	`down`	15
	`left`	15
	`right`	15
`ransac`	`n`	1
`ransac`	`K`	1000

Results

I. Source images

II. Result Image

Bonus

1. Bundle adjustment

Reference

1. M. Brown, D. G. Lowe, **Recognising Panoramas**, ICCV 2003.
2. Christopher G. Harris and Mike Stephens, **A COMBINED CORNER AND EDGE DETECTOR**, 1988.
3. Tony Lindeberg, **Scale Invariant Feature Transform**, Scholarpedia 2012.