ICIC Express Letters
Part B: Applications
Volume **7**, Number **7**, July **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **1397–1403**

IMAGE STITCHING BASED ON ORB FEATURE AND RANSAC

Yongqing Wu, Xiuqin Su and Xiaoli Hu

Xi'an Institute of Optics and Precision Mechanics of CAS
University of Chinese Academy of Sciences
No. 17, Xinxi Road, Xi'an 710119, P. R. China
{ wuyongqing; huxiaoli }@opt.cn; suxiuqin@opt.ac.cn

Received November 2015; accepted February 2016

Abstract. *In this paper, we introduced an image stitching algorithm which is based on ORB (Oriented FAST and Rotated BRIEF) feature and RANSAC (Random Sample Consensus) algorithm. First, we use ORB feature and brute force algorithm to find correspondences between images. Second, we apply the RANSAC algorithm to estimating homography robustly. Finally, the algorithm blends the images according to the homography and the image feathering algorithm was engaged to remove the visible seams due to exposure differences. From the results of experiments, we can draw a conclusion that the algorithm we proposed can stitch images seamlessly with a low computation time.*
Keywords: Image stitching, ORB feature, Homography estimation, RANSAC

1. **Introduction.** Image stitching is the process of combining multiple images with overlapping fields of view to produce a segmented panorama or high-resolution image. It is also known as image mosaicing. Image stitching has been an attractive research area because of its wide range of applications.

In the research literature, a number of image stitching algorithms have been proposed over the last two decades and most of those algorithms are feature-based [1], such as SIFT-based [2] and SURF-based [3]. Since the performance of feature-based stitching algorithms depends on the underlying feature algorithm [4] and both SIFT (Scale Invariant Feature Transform) and SURF (Speeded Up Robust Features) impose a large computational burden [5], SIFT and SURF based image stitching algorithms are time consuming.

In 2011, Rublee et al. proposed the ORB [5] feature which is an order of magnitude faster than SURF and over two orders faster than SIFT while it has similar matching performance with SIFT. So the ORB can be an efficient alternative to SIFT or SURF.

In this paper we introduced an image stitching algorithm combined ORB with RANSAC [6]. Our main motivation is to reduce the time of image stitching applications on standard PCs. Since both ORB and feathering used by our algorithm are fast and effective and RANSAC used by our algorithm guaranteed the robustness, our algorithm can stitch images seamlessly and it is almost 5 times faster than SURF-based algorithms and 8 times faster than SIFT-based algorithms.

2. **The ORB Feature.** The ORB feature combined the FAST (Features from Accelerated Segment Test) keypoint detector [7] and the BRIEF (Binary Robust Independent Elementary Features) descriptor [8]. For this reason it is called ORB (Oriented FAST and Rotated BRIEF). The FAST keypoint detector and the BRIEF descriptor are attractive because of their good performance and low cost.

2.1. **The FAST keypoint orientation.** FAST features are widely used because of their computational properties. However, FAST features do not have an orientation component. In [5] Rublee et al. added an efficiently-computed orientation for FAST.

FAST takes the intensity threshold between the center pixel and those in a circular ring with the center as the only parameter. In [5], Rublee et al. used FAST-9 (circular radius of 9), which has good performance. FAST does not produce a measure of cornerness and it has large responses along edges. To order the FAST keypoints, a Harris corner measure is applied. To get N keypoints, first set the threshold low enough to get more than N keypoints, and then order them according to the Harris measure, and pick the top N points.

To produce multi-scale features, a scale pyramid of the image is employed, and produce FAST features at each level in the pyramid.

A simple but effective technique, the intensity centroid [9], is used to measure the corner's orientation. The intensity centroid assumes that a corner's intensity is offset from its center, and this vector can be used to impute an orientation. Rosin defines the moments of a patch as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y), \tag{1}$$

and with those moments we can find the centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \tag{2}$$

A vector can be constructed from the corner's center, O, to the centroid, $\overrightarrow{OC}$. Then the orientation of the patch simply is:

$$\theta = atan2(m_{01}, m_{10}), \tag{3}$$

where $atan2$ is the quadrant-aware version of arctan. To improve the rotation invariance of this measure they empirically choose r to be the patch size, which makes sure that moments are computed with x and y remaining within a circular region of radius r.

2.2. **The rBRIEF: Rotation-aware BRIEF.** BRIEF's performances are similar to SIFT in many respects, including robustness to blur, lighting, and perspective distortion. While, BRIEF is cheaper to compute, more compact to store, and faster to compare with each other. However, it is very sensitive to in-plane rotation. In [5], Rublee et al. introduced steered BRIEF and rBRIEF.

2.2.1. *BRIEF.* The BRIEF descriptor is one of the binary visual descriptors; it is a bit string description of an image patch constructed from a set of simple binary intensity tests. Given a smoothed image patch, p, a binary test τ is defined by:

$$\tau(p; x, y) := \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) \geq p(y) \end{cases}, \tag{4}$$

where $p(x)$ is the intensity of p at point x. Then the feature is defined as a vector of n binary test:

$$f_n(p) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i). \tag{5}$$

There are many different types of distributions of tests in [8]. In [5] a Gaussian distribution around the patch center is used and a vector length $n = 256$ is chosen.

2.2.2. *Steered BRIEF.* To make BRIEF to be invariant to in-plane-rotation, an efficient method is to steer BRIEF according to the orientation of the keypoints. For any feature set of n binary tests at position (x_i, y_i), the matrix is defined as:

$$S = \begin{pmatrix} x_1, \ldots, x_n \\ y_2, \ldots, y_n \end{pmatrix}. \tag{6}$$

Using the patch orientation θ and the corresponding rotation matrix R_θ , a steered version S_θ of S can be constructed:

$$S_\theta = R_\theta S. \tag{7}$$

Then the steered BRIEF operator becomes

$$g_n\left(p,\theta\right) := f_n\left(p\right) \mid \left(x_i, y_i\right) \in S_\theta. \tag{8}$$

A lookup table of precomputed BRIEF patterns by discretizing the angle to increments of $2\pi/30$ (12 degrees) was constructed. As long as the keypoint orientation θ is consistent across views, the correct set of points S_θ will be used to compute its descriptor.

2.2.3. *rBRIEF.* One of the pleasing properties of BRIEF is that each bit feature has a big variance and a mean near 0.5. High variance makes feature more discriminative, since it responds differently to inputs. However, once BRIEF is oriented along the keypoint direction to give steered BRIEF, the means are shifted to a more distributed pattern and the variance becomes significantly low. Another desirable property is to have the tests uncorrelated, and since then each test will contribute to the result.

To recover from the loss of variance in steered BRIEF, and to reduce correlation among the binary tests, Rublee et al. [5] developed a learning method for choosing a good subset of binary test. Their strategy is to search among all possible binary tests to find ones that both have high variance, as well as being uncorrelated.

Rublee et al. used greedy search to search for a set of uncorrelated tests with means near 0.5. The result is called rBRIEF. The rBRIEF has significant improvement in the variance and correlation over steered BRIEF.

3. **Feature Matching and Homography Estimation.**

3.1. **Feature matching.** Binary features are compared using Hamming distance, which for binary data can be computed by performing a bitwise XOR operation followed by a bit count on the result. This involves only bit manipulation operations which can be performed quickly, especially on modern computers where there is hardware support for counting the number of bits that set in a word.

In this paper, we employ brute force algorithm to match the features. For each descriptor in the first set, the algorithm computes the Hamming distance with every descriptor in the second set. For descriptor a and descriptor b, the Hamming distance between a and b can be defined as:

$$d_H = \sum_n a_n \oplus b_n, \tag{9}$$

where n is the vector's length, a_n and b_n represent the nth element of the vector.

The Hamming distance represents the total number of mismatches between the two descriptors. So each time the algorithm chooses the two descriptors with the least Hamming distance as the matched descriptors.

3.2. **Homography estimation.** A homography is a projective transformation, which is a non-singular linear transformation of homogeneous coordinates. It describes what happens to the perceived positions of observed objects when the point of view of the observer changes. In more formal terms, a homography is an invertible transformation from the real projective plane to the projective plane that maps straight lines to straight lines.

Typically, homographies are estimated between images by finding feature correspondences in those images. The most commonly used algorithms make use of point feature correspondences.

In homogeneous coordinates, the relationship between two corresponding points x and x' can be written as:

$$c \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{10}$$

where c is any non-zero constant, $\begin{pmatrix} x' & y' & 1 \end{pmatrix}^T$ represents x', $\begin{pmatrix} x & y & 1 \end{pmatrix}^T$ represents x, and $H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$. From Equation (10) we can get the following two equations:

$$-h_1x - h_2y - h_3 + (h_7x + h_8y + h_9)x' = 0 \tag{11}$$

$$-h_4x - h_5y - h_6 + (h_7x + h_8y + h_9)y' = 0 \tag{12}$$

So for one pair matching feature, there are two linear equations.

3.2.1. *DLT algorithm.* The DLT (Direct Linear Transform) [10] algorithm is a simple algorithm used to solve for the homography matrix H given a sufficient set of point correspondences.

Equations (11) and (12) can be written in matrix form as:

$$A_ih = 0, \tag{13}$$

where

$$A_i = \begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{pmatrix} \text{ and}$$

$$h = \begin{pmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{pmatrix}^T.$$

Since each point correspondence provides 2 equations, 4 correspondences are sufficient to solve for the 8 degrees of freedom of H. The restriction is that no 3 points can be collinear. For 2×9 A_i matrices can be stacked on top of another to get a single 8×9 matrix A. Then the 1D null space of A is the solution space for h.

3.2.2. *Robust estimation: Dealing with outliers.* The DLT algorithm is only robust with respect to noise if the source of this noise is in the measurement of the correspondence feature positions. However, there will be other situations where the input will be corrupted with completely false correspondences, meaning that the two features in the images do not correspond to the same real world feature at all. We use RANSAC to distinguish inlier and outlier correspondences so that the homography can be estimated robustly using only inlier matches.

RANSAC is the most commonly used robust estimation method for homographies according to [11]. The idea of the algorithm is simple. For a number of iterations, a random sample of 4 correspondences is selected and a homography is computed from those 4 correspondences. Then, depending on each other correspondence's concurrence with H, each correspondence is classified as an inlier or outlier. After all of the iterations are finished, the iteration that contained the maximum of inliers is selected. Then H can be recomputed from all of the correspondences that were considered as inliers in that iteration.

Given the probability that a feature match is correct between the two matching images (the inlier probability) is p_i, the probability of finding the correct transformation after n iterations is

$$p\,(H \text{ is correct}) = 1 - (1 - (p_i)^r)^n. \tag{14}$$

After a large number of iteration the probability of finding the correct homography is very high. According to [2], for an inlier probability $p_i = 0.5$, the probability that the correct homography is not found after 500 iterations is approximately 1×10^{-14}.

4. **Image Blending.** One problem in image stitching is exposure differences between images. Exposure differences are a common occurrence, especially with digital photographs. If the differences are not corrected, the stitched image will appear to have visible seams, even when the images are blended in overlapping regions.

To remove the visible seams between the images, image blending techniques like average blending, alpha blending, image feathering, pyramid blending have been proposed. Average blending is the simplest technique, and it simply takes an average value at each pixel. However, it usually does not work very well.

A better approach to averaging is to weight pixels near the seam more heavily while to down-weight pixels far away from the seam. Weighted averaging with a distance map is often called feathering and does a reasonable job of blending over exposure differences.

In this paper, we apply image feathering to blending two images with overlapping regions. The equation used to compute the destination pixel is:

$$I(i,j) = [1 - w(i,j)] * I_1(i,j) + w(i,j) * I_2(i,j), \tag{15}$$

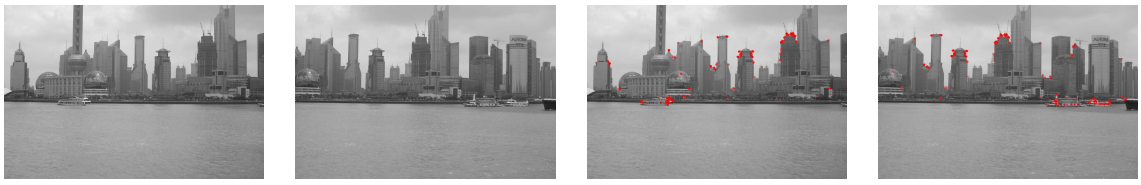
where $w(i,j)$ is the weight computed according to the shortest distance between $I_2(i,j)$ and the seam; $I_1(i,j)$ and $I_2(i,j)$ are the pixel in point (i,j) from the two images respectively.

5. **Experimental Results.** We test our algorithm on a set of 181 images from the Adobe Panoramas Data Set [12], and there are exposure differences between images. Experiments are conducted on a standard PC with a 3.4GHz Intel(R) Core(TM) i3-2130 CPU and 4G RAM.

Here we give one of the tests. Figure 1(a) and Figure 1(b) were captured in Shanghai, each image being 1024×683 resolutions. Figure 1(c) and Figure 1(d) show the ORB features detected, and we can see that there are a lot of features from the same object detected in the two images. The results prove that ORB has a good repeatability property.

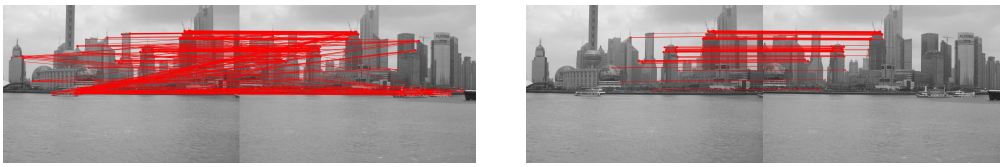
Figure 2(a) shows matched features without RANSAC, there exist some wrong matches, while Figure 2(b) shows matched inliers with RANSAC engaged. From those two images we can see that RANSAC can do a good job of deleting outliers; thus, a more robust estimation can be computed from inliers which guaranteed the algorithm's robustness.

Figure 3 is the stitching result, one engaged feathering while the other not. We can see that there is visible seam in Figure 3(a) and it is removed from Figure 3(b). The



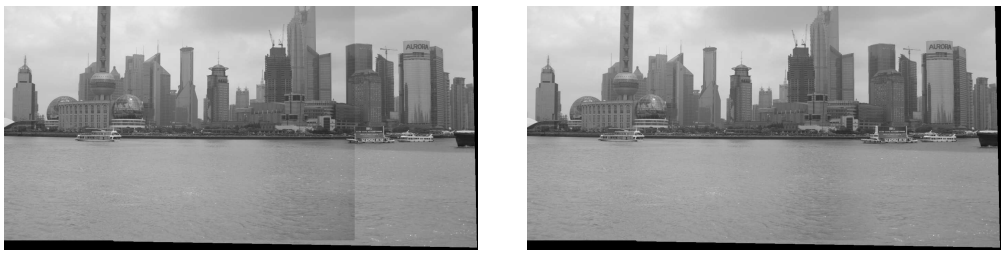
(a) Image 1 (b) Image 2 (c) ORB in Image 1 (d) ORB in Image 2

FIGURE 1. Images of Shanghai and ORB features detected in them



(a) ORB matches without RANSAC (b) Matched inliers with RANSAC

FIGURE 2. Matched features



(a) Blending without feathering (b) Blending with feathering

FIGURE 3. Image blending results

TABLE 1. Time performance of different feature-based stitching algorithms

	SIFT-based	SURF-based	ORB-based
Time(s)	5.60504	3.68512	0.712629

results show that the feathering algorithm performs reasonably well. Since feathering is very fast, it contributes a lot in reducing time.

By replacing ORB with SIFT and SURF we compared the time performance of those algorithms, the results shown in Table 1, from which we can see that our algorithm is almost 8 times faster than SIFT-based and 5 times faster than SURF-based algorithm.

From all those results we can conclude that our ORB-based stitching algorithm can stitch images seamlessly and it is much faster than algorithms based on SIFT or SURF.

6. **Conclusion.** In this paper, we proposed an image stitching algorithm based on ORB and RANSAC. We used ORB algorithm and brute-force matching algorithm to get the point correspondences between images. We engaged the DLT algorithm which used the point correspondences to compute the homography between images. Moreover, we applied RANSAC to deleting wrong point correspondences so we can compute a more robust homography only from the inliers. We also used the image feathering algorithm which can blend images without visible seams. The experiments results show that the algorithm we proposed can stitch images seamlessly and it is much faster than algorithms based on SIFT or SURF.

In the future, we will test our algorithm on cellphones with Android platform and we will try to optimize the algorithm to make sure that it performs well on low-power and low-frequency devices.

REFERENCES

[1] D. Ghosh and N. Kaabouch, A survey on image mosaicing techniques, *Journal of Visual Communication and Image Representation*, vol.34, pp.1-11, 2016.

[2] M. Brown and D. G. Lowe, Automatic panoramic image stitching using invariant features, *International Journal of Computer Vision*, vol.74, no.1, pp.59-73, 2007.

[3] W. Rong, H. Chen, J. Liu and Y. Xu, Mosaicing of microscope images based on SURF, *International Conference on Image and Vision Computing New Zealand*, pp.271-275, 2009.

[4] T. Lindeberg, Image matching using generalized scale-space interest points, *Journal of Mathematical Imaging & Vision*, vol.52, no.1, pp.3-36, 2015.

[5] E. Rublee, V. Rabaud and K. Konolige, ORB: An efficient alternative to SIFT or SURF, *2011 IEEE International Conference on Computer Vision*, pp.2564-2571, 2011.

[6] M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, vol.24, no.6, pp.381-395, 1981.

[7] E. Rosten and T. Drummond, Machine learning for high-speed corner detection, *Computer Vision – ECCV 2006*, pp.430-443, 2006.

[8] M. Calonder, V. Lepetit, C. Strecha and P. Fua, BRIEF: Binary robust independent elementary features, *Computer Vision – ECCV 2010*, pp.778-792, 2010.

[9] P. L. Rosin, Measuring corner properties, *Computer Vision and Image Understanding*, vol.73, no.2, pp.291-307, 1999.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[11] J. J. Lee and G. Y. Kim, Robust estimation of camera homography using fuzzy RANSAC, *Computational Science and Its Applications – ICCSA 2007*, pp.992-1002, 2007.

[12] J. Brandt, Transform coding for fast approximate nearest neighbor search in high dimensions, *IEEE Conf. on Computer Vision and Pattern Recognition*, pp.1815-1822, 2010.