Proc. 1st International Conference on
Machine Learning and Data Engineering (iCMLDE2017)
20-22 Nov 2017, Sydney, Australia
ISBN: 978-0-6480147-3-7

Real-Time Panorama Image Stitching based on Multi-Threaded System

Ahyun Lee and Insung Jang

IoT Research Division, Electronics and Telecommunications Research Institute (ETRI),
Daejeon, South Korea

Corresponding author's E-mail: ahyun@etri.re.kr

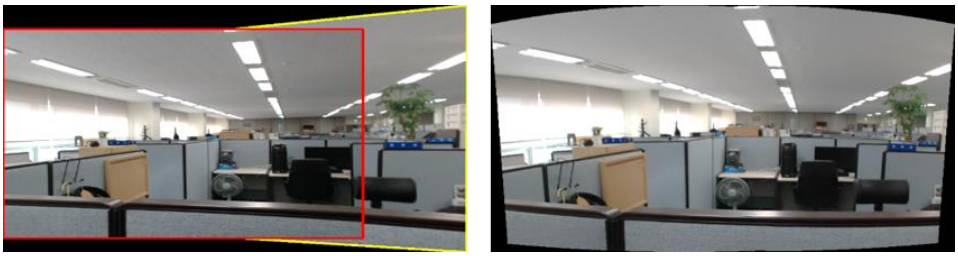
Abstract

This paper proposes a user-friendly panorama image stitching system with a real-time preview. Previous approaches do not have a choice but to the select source images while the user predicts the stitching results through trial and error. Our contribution shows a real-time preview of the stitching results based on the multi-threaded tracking and the blending system. It can help the user easily generate a desired panorama image such as a wide-angle view or building. In our system, the object image designated as an initial frame is tracked to generate a real-time preview. We evaluated the accuracy of the proposed tracking method as compared with a scale-invariant feature transform, such as speeded-up robust features, and oriented FAST, rotated BRIEF. Our experiment results show that our approach can robustly track the object image and provide quality real-time preview images.

Keywords: image-stitching, multi-thread, ORB.

1. INTRODUCTION

Panorama image stitching is a well-known method for creating a wide-angle view image with limited tools or environments. Previous approaches have been concerned with the quality of the panorama image stitching result or fully automated systems as suggested by Cha (2012) and Zhang (2014). And they select the source images, while users predict the stitching results using a guideline for helping capture the source image. To make a view of a user-desired shape, trial and error is required. In this paper, we describe a multi-threaded system based on invariant features for panorama image stitching with a real-time preview. Our contribution is implementing a user-friendly and real-time based stitching system. The predicted stitching result can be shown beforehand in real-time according to the user's controls, as shown in Figure 1(a). This can help the user easily generate a desired panorama image such as a large land-view or building image. The proposed system is composed of detection and blending parts. The blending part is difficult to execute in real time, and thus we configured a real-time executable multi-threaded system. We hope that our multi-threaded approach will be useful to users unaccustomed to such systems.
The rest of this paper is structured as follows. In Section 2, we describe the proposed method in detail in the context of a multi-thread based system. In Section 3, we describe our experiment results. Finally, in Section 4, we provide some concluding remarks regarding our proposal.



(a) **(b)**

Figure 1. An example of panorama image stitching: (a) real-time preview image and (b) stitching result

2. MULTI-THREAD SYSTEM

Our proposed system is implemented as a multi-threaded system to achieve a real-time performance as suggested by Lee and Hollerer (2009). It can be divided into two threads: the tracking thread tracks the objet image, and the blending thread blends the user-selected source images. The tracking thread is the main thread and is conducted for every frame. The blending thread using an invariant feature based panorama image stitching approach incurs significant computational costs. It can only be applied when the user selects a source image for blending. Prior to the present study, we assume that we calibrate the camera, and its captured image is un-distorted through Zhang's camera calibration as suggested by Zhang (2000).

2.1. Tracking Thread

For the tracking, we use such concepts as an object and scene image. In the initial frame, the user sets the object image for tracking the target. The tracking thread tracks the object image in consecutive frames after the set-up of the initial frame. Our approach uses oriented FAST, rotated BRIEF (ORB) as suggested by Rublee et al (2011) for detecting and matching the features. ORB is a well-known object tracking technique that is rotation invariant and resistant to noise. Moreover, its computational cost is low for real-time applications. However, when using ORB alone, frequent tracking failures occur. In the proposed method, the object features p^{obj} are first detected in an object image using ORB in the initial frame. In addition, we set p^{obj} to object features l^{obj} for the pyramidal Lucas-Kanade tracker (LKT) as suggested by Bouguet (2001). In later frames, l^{obj} is tracked as shown in Table 1.

Table 1. Pseudocode for the tracking thread

FUNCTION: TRACK( )
  Detect object features p^{obj} in the initial frame by ORB
  Set p^{obj} to LKT object features ${l^{obj}}_0$
  i = 0;
  BEGINLOOP:
    Track ${l^{obj}}_i$ by LKT, its tracked results are l_i
    Outlier filtering with ${l^{obj}}_i$ and l_i.
    n = number of ${l^{obj}}_i$
    IF: $n < 8$
      Detect scene features p_i by ORB
      Match p^{obj} and p_i, ${m^{obj}}_i$ and m_i is matched features
      FOR: each feature $k \in {m^{obj}}_{i,k}$
        FOR: each feature $j \in {l^{obj}}_{i,j}$
          IF NOT: check same feature $({l^{obj}}_{i,j}, {m^{obj}}_{i,k})$
            Set $({m^{obj}}_{i,k},\ m_{i,k})$ to $({l^{obj}}_{i,n},\ l_{i,n})$
            $n = n + 1$
    $i = i + 1$
  ENDLOOP

A failed match leads to a preview failure. For robust tracking, we use outlier filtering as suggested by Lee et al (2015) for choosing the good features to track. As a result, if the number of inliers of l_i is lower than eight, which is degree of freedom for a perspective projection transformation, detection of the ORB features is conducted to add more features for robust tracking. The matching results of $m^{obj}{}_i$ and m_i in the *i*-th frame are checked to avoid duplication with features already being tracked. If newly detected features are not tracked, they are added to $l^{obj}{}_i$ and l_i. A warped image is generated as a tracking result in every frame for a real-time preview. To warp the image, a perspective projection matrix **H** is necessary. **H** is calculated based on the relationship between the object features $l^{obj}{}_i$ and the scene image features l_i in the *i*-th frame with a random sample consensus (RANSAC) as suggested by Fischler and Robert (1981).

Tracking reliability depends on the result of computing a perspective projection matrix between an object and the scene image features. Failed tracking causes a failure in computing a perspective projection matrix. In addition, a failed visualization hinders the concentration of the users. In this paper, we use outlier filtering for robust tracking. The proposed outlier filtering classifies the detected features into inliers and outliers based on the relationship of the features between an object image and a scene image as shown Figure 2.

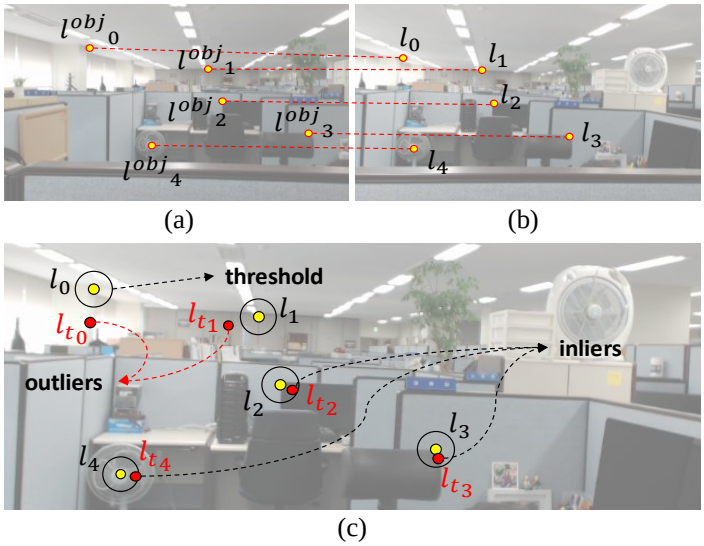


Figure 2. Compute Euclidean distances for outlier filtering: (a) detected features in the object image, (b) tracked features in the scene image, and (c) if a transformed feature in the scene image is out of range, it is an outlier.

Table 2. Outlier filtering for classifying into inliers and outliers

1) Compute a perspective projection matrix **H** from between features $l^{obj}{}_i$ in an object and l_i in the i-th scene image using RANSAC.
2) Compute Euclidean distance $d_{i,j}$ between the j-th feature $l_{i,j}$ and a transformed feature $l_{t_{i,j}} = \mathbf{H} \cdot l^{obj}{}_{i,j}$.
3) If $d_{i,j}$ is smaller than a case-specific threshold, it is an inlier. In other cases, they are outliers.

The main issue is determining the RANSAC distance threshold according to the execution environment for outlier filtering as suggested by Fischler and Robert (1981). An adaptive RANSAC threshold decision enables a case-specific threshold to be determined for each specific frame. In this paper, we update the threshold for every 100 frames.

Table 3. Adaptive RANSAC distance threshold decision

1) Set the initial threshold to $t_0 = s/100$, where the scale s is the length of the diagonal of the camera frame.
2) Euclidean distances $d_{i,j}$ between $l_{i,j}$ and $l_{t_{i,j}}$ are computed until the *n*-th frame. In the n-th frame, $d_{j_{mean}} = \frac{1}{\mathrm{n}}\sum_{i=1}^{\mathrm{n}} d_{i,j}$ is calculated from each *j*-th feature .
3) If $d_{j_{mean}}$ is twice the value of $d_{mean} = \frac{1}{n}\sum_{j=1}^{n} d_{j_{mean}}$, $d_{j_{mean}}$ is removed to exclude impulse errors.
4) Update the case specific threshold t_1 through d_{mean}, the dataset is computed using Otsu threshold decision method as suggested by Otsu (1975).
5) At every *n*-th frame interval, steps 2) through 4) are repeated.

2.2. Blending Thread

The blending thread begins when the user selects a scene image to be stitched with the object image in the tracking thread. In this work, we use automatic panoramic image stitching with invariant features as suggested by Brown and Lowe (2007). This method uses invariant local features to find matches between all of the images. A probabilistic model verifies the image matches in unordered image sets, and stitches them automatically. Figure 3 shows the procedural flowchart and conceptual time sequence for each process. The first step is to detect and match the scale-invariant feature transform (SIFT) features as suggested by Lowe (2004) in all image sources, which are the selected scene images and the object image. In addition, we find the connected components of the image matches. For each connected component, a bundle adjustment as suggested by Triggs et al (1999) is performed to estimate the camera position, which is difficult to do in real-time. Therefore, we create a new thread for blending. When the blending thread is complete, the stitching result is updated as the object image. ORB features and descriptors are extracted in the new object image for subsequent tracking. Finally, the stitching result is reflected in the real-time preview in the tracking thread.

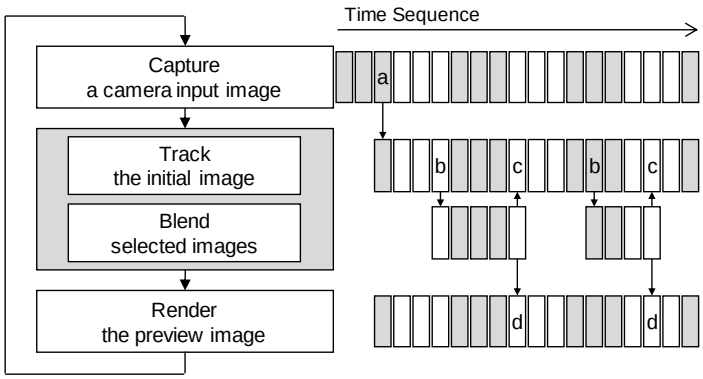
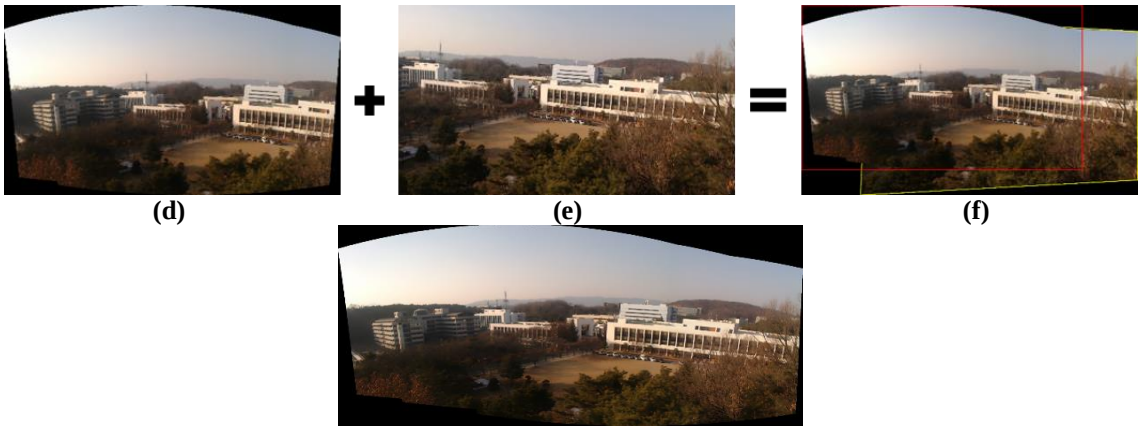


Figure 3. Procedural flowchart and conceptual time sequence for each process. In this figure, a, b, c, and d are as follows: (a) set the initial frame, (b) select the scene image, (c) update the object image, and (d) apply the blend result to the real-time preview image.

(a) **(b)** **(c)**



(d) **(e)** **(f)**

(g)

Figure 4. Example of gradual steps for generating a panorama image using our approach with a real-time preview: (a) the first object image, (b) a source image, (c) a real-time preview by (a) and (b), (d) the stitching result of (c) and the second object image, (e) a source image, (f) a real-time preview by (d) and (e), and (g) the final stitching result of (f).

3. EXPERIMENT

The experiments were conducted on a desktop computer with a 4 GHz Intel® core™ i7 CPU using a USB 2.0 camera (Logitech webcam C920). The camera was calibrated using Zhang's calibration method as suggested by Zhang (2000), and its captured images are undistorted to remove the lens distortion. Figure 4 shows an example of the gradual steps used for generating a panorama image by our proposed method with a real-time preview. The real-time preview helps the user generate a user-desired shape. Table 4 shows the process time per frame of each step in our multi-threaded system. The time depends on the camera resolution. For real-time execution with a camera based on 30 fps, the entire process has to be finished within 33*ms*. Thus, we use a resolution of 640 × 480 for real-time previews in the tracking thread. The final panorama image will be generated into a 1920 × 1080 resolution, which we already selected for the source images.

Table 4. Comparison of reliability in each tracking approach. The RMSE is the mean value of the x, y, z-axes.

	Time per frame (ms)		
	640×480	1280×720	1920×1080
Tracking thread	**12.4**	**34.71**	**87.18**
Capture & non-distortion	5.07	17.61	37.4
Optical flow tracking	3.8	7.91	13.32
Calculate H	0.82	1.15	3.85
Outlier filtering	0.03	0.04	0.05
Generate preview image	2.68	8.0	32.56
Blending thread	**1662.33**	**2412.4**	**5074.33**
Estimate transform	743.5	1096.0	2467.5
Compose panorama	878.0	1227.4	2449.67
Update the object image	40.83	89.0	157.17

Table 5. Comparison of reliability in each tracking approach. The RMSE is the mean value of the x, y, z-axes.

Index	SIFT	SURF	ORB	Proposed
RMSE (mm)	18.37	39.34	111.39	4.16

In another experiment, we examined the accuracy of the proposed tracking method. We use ORB features for detection and matching. LKT tracks the detected ORB features with outlier filtering in the tracking thread. In our experiments, camera input images were captured while the camera remained fixed. We generate occlusions by hand or through changes in illumination. The root mean square deviation (RMSE) is computed between the camera pose in the first frame without an occlusion and camera poses in the other frames according to SIFT, the speed up robust feature (SURF) as suggested by Bay et al (2006), ORB, and our proposed approach. Because the camera is fixed, the closer the RMSE is to zero, the more stable it is. A camera pose is the fourth column vector of the camera extrinsic matrix using Equation 1. The experiment results illustrate that the proposed method with outlier filtering achieves better results than the other cases, as shown in Table 5.

$$p_i = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [r_1 \quad r_2 \quad r_3 \quad (x, y, z, 1)^{\mathrm{T}}] \cdot p^{obj} \tag{1}$$

4. CONCLUSION

This paper presented a user-friendly panorama image stitching approach with a real-time preview. The tracking and blending threads are multi-threaded for a real-time performance. A prediction of the panorama results can be shown through the selection of the scene image in real-time. Therefore, a user can more easily generate a panorama image than in off-line based approaches. To evaluate the accuracy of the proposed tracking method, we compared it with SIFT, SURF, and ORB. The experiment results show that our approach can robustly track an object image and provide quality preview images. We believe that our proposed multi-threaded approach can be useful to users who are unaccustomed to panorama image stitching applications.

ACKNOWLEDGMENTS

This work was supported by MOLIT (Ministry of Land, Infrastructure and Transport), Korea, under the UPA (Urban Planning & Architecture) research support program supervised by KAIA (Korea Agency for Infrastructure Technology Advancement) (grant 13 Urban Planning & Architecture 02).

REFERENCES

Bay H, Tuytelaars T, Gool LV (2006). Speeded-up robust features (SURF), European Conference on Computer Vision, Springer Berlin Heidelberg, 404-417.

Bouguet JY (2001). Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm, Intel Corporation, 5, 1-10.

Brown M, Lowe DG (2007). Automatic panoramic image stitching using invariant features, International journal of computer vision, 74(1), 59-73.

Cha JH, Jeon YS, Moon, YS, Lee SH (2012). Seamless and fast panoramic image stitching, IEEE International Conference on Consumer Electronics (ICCE), 29-30.

Fischler MA, Robert CB (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM, 24(6), 381-395.

Lee T, Hollerer T (2009). Multithreaded hybrid feature tracking for markerless augmented reality, IEEE Transactions on Visualization and Computer Graphics, 15(3), 355-368.

Lee A, Lee JH (2015). Multi-threaded tracker with outlier filtering for spatial augmented reality, International Technical Conference on Circuits Systems, Computers and Communications (ITC-CSCC), 494-495.

Lowe D (2004). Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60(2), 91-110.

Otsu N (1975). A threshold selection method from gray-level histograms, IEEE Trans. Sys. Man. Cybern., 9, 62-66.

Rublee E, Rabaud V, Konolige K (2011). ORB: An efficient alternative to SIFT or SURF, Computer Vision (ICCV), 2011 IEEE International Conference on, 2564-2571.

Triggs B, McLauchlan PF, Hartley RI. Fitzgibbon AW (1999). Bundle adjustment: A modern synthesis, International workshop on vision algorithms, Springer Berlin Heidelberg, 298-372.

Zhang Z (2000). A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), 1330-1334.

Zhang F, Liu F (2014). Parallax-tolerant image stitching, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3262-3269.