A fast and robust real-time surveillance video stitching method

Tao Yang*, Fenlin Jin and Jianxin Luo
Command & Control Engineering College, Army Engineering University of PLA, Nanjing, Jiangsu, 210007, China
azhaoyuexin@stu.ahu.edu.cn yang_tao_lgd@163.com

Abstract. Real-time video stitching can build a wider field of view for surveillance, which faces a compromise between stitching speed and visual quality. A fast and robust real-time surveillance video stitching method is proposed to deal with the ghosting effect caused by moving objects and misalignments caused by background change or slight camera shift through automatic updating. By stitching key frames, parameters such as pix mapping table, stitching seams and blending weights are calculated, and most of subsequent frames are directly blended with CUDA acceleration based on the pre-calculated stitching parameters. Fast and effective algorithms are designed to detect the change of stitching seam and background during the whole stitching process, which determines whether to update the stitching seam or recalculate stitching parameters. Experiments show that this method can robustly and automatically solve the ghosting and misalignments to improve visual quality and achieve satisfactory real-time performance.

1. Introduction

With the development of multimedia technology, demand for large-view, high-resolution images and videos has gradually increased. As a result, image and video stitching technology has also received more and more attention. After years of researching, image stitching has been well studied, from Auto Stitch[1] with a global 2D projective warp, to Dual-Homography Warping(DHW) [2] algorithm using two homography to align two planes separately and Smoothly Varying Affine (SVA) [3] with multiple affine transformations, to As-Projective-As-Possible(APAP) [4] with a local projective warp in each cell and more advanced methods [5-6] based on APAP, the ability to handle parallax is enhanced, while the stitching effects are becoming more natural. Seam cutting based methods [7-8] do not need to strictly align the overlapping areas, but only need to find the best stitching seam, which can relieve the ghost caused by parallax.

Video stitching is based on image stitching but confronted with more complicate problems. Real-time video stitching mainly suffers from the balance between stitching effect and stitching speed. Stitching with frame-by-frame registration achieves accurate alignment, but is time-consuming. Besides, jittering is another problem, especially for shaking videos captured from hand-held cameras. For fixed cameras, LIU[9] used the stitching model calculated with first few frames to stitch subsequent frames and didn’t consider objects moving through overlapping areas, as is the case with many surveillance video stitching methods [10]. Certainly, they cannot deal with the ghosting of objects moving through the overlapping area and stitching seam. HE [11] proposed the layered warping to align the background and selective seam updating to avert ghosting but it could not deal with unexpected background change.
nor camera shift automatically, neither can the method in [12]. So all of the methods above are not robust enough.

In this paper, both the ghosting effect from moving objects and background change are taken into consideration to improve the fast stitching based on the fixed stitching parameters calculated from the initial model. We present a fast and robust surveillance video stitching method with automatic stitching seam and parameters updating. It detects the change of stitching seam and background at a small cost and only updates the stitching seam and parameters when necessary, which will not affect the real-time performance of video stitching and can robustly deal with the ghosting caused by moving objects and misalignment caused by unexpected background change.

2. Key frame stitching and parameters calculating

Key frames are the ones where the stitching parameters are calculated during the stitching process. We implemented the stitching of key frames based on the stitching pipeline of OpenCV stitching module. The stitching of key frames can be roughly divided into three stages: image registration, exposure compensation and seam finding, and image blending. Image registration can be further divided into three steps: feature extraction and matching, camera parameters estimation, and image projecting. We customize the stitching pipeline showed in figure 1. Specifically, we use the classical combination of SURF [13] feature detecting, 2-NN matching, RANSAC [14] purification and cylindrical projection in image registration. Block compensation and graph cut seam finder are adopted in exposure compensating and seam finding. Feature blending is applied to synthesize the final image. Feature detecting and matching, image projecting, warping and blending are accelerated with CUDA.

3. Automatic seam updating

When a moving object passes through the stitching seam, ghosting effects as shown in figure 2 would appear if the stitching seam is fixed. If the object stays at stitching seam, the visual defect will be long-term. To address this problem, a fast and effective detection method for moving objects passing through stitching seam along with the seam updating rules are designed as follows:

Step 1: Whenever stitching seam is found or updated, save the pixel values and coordinates of all the points on the stitching seam.

Step 2: Before directly blending each frame on the basis of existing parameters, check whether there are objects passing through the stitching seam according to the changing rate of points on the stored stitching seam. Specifically, a point is defined as changed (if the changing rate of its pixel value exceeds a certain threshold $\delta$ as in equation (1); then count the number of changed points and calculate the ratio of changed points to the total number of points on the seam $p_{\text{seam, change}}$ as in equation (2).
where \( v_{prev}^i \) and \( v_{cur}^i \) represent the pixel value of the \( i \)th point on the stored seam and that of the point in the same position on current image and respectively.

Updating rule: If \( p_{seam\_change} \) reaches a certain threshold \( \eta \), we believe that there are objects moving across the seam and it should be updated right now. The sensitivity of seam detecting and updating can be adjusted by setting the value of the threshold \( \delta \) and \( \eta \). In our experiments, they are set to 0.1 and 0.2 empirically.

4. Automatic stitching parameters updating

In methods of HE [12] and many others, the background is supposed to be static and the stitching parameters calculated at first won’t be updated any more. However, intentional great change in background or accidentally slight shift of cameras by external force are not impossible. Video stitching methods with fixed stitching parameters calculated by initial models are not robust enough to deal with these unexpected changes, thus misalignments like figure 3 would yield. To make the stitching more robust, we designed a background change detection method that combines grid vertex based change detection and feature point tracking, through which stitching parameters can be updated automatically when necessary.

4.1. Grid vertex based change detection

To detect the change of the background, we divide the image into uniform grids and determine whether there exists background change or camera shift by the change rate of the grid vertices’ pixel values between adjacent frames. As the same method we use in section 3, a grid vertex is defined as changed if the change rate of its pixel value exceeds a certain threshold as in equation (1). The following rule, similar to equation (2) is used to judge if changes in background or camera shift occur:

\[
P_{vertex\_change} = \frac{\sum_{i=1}^{N} C(i)}{N_{total\_vertex\_num}}
\]
where $p_{\text{vertex change}}$ refers to the change rate of grid vertices. If $p_{\text{vertex change}}$ reaches a threshold $\lambda$, we consider that the background has changed or the camera has shifted.

![Figure 3. Misalignment caused by camera shift. (a)before camera shift, (b)after camera shift](image)

Take figure 4(a) and 4(b) as an example, camera at frame $F_t$ is slightly shifted to the left with respect to that at frame $F_{t-1}$. The two frames are both divided into 40×40 grids. In figure 4(c), the changed vertices detected by equation (1) are filled with blue and their corresponding vertices in figure 4(d).

![Figure 4. Example of grid vertex based change detection. (a) frame $F_t$ divided into grids, (b) frame $F_{t+1}$ divided into grids; (c) and (d): changed vertices filled in blue](image)

In the example above, a 640×480 resolution image has 221 grid vertices in total, and the number of changed vertices is 106. Taking into consideration that when a moving object passes by, some grid vertices will also change, so the threshold cannot be too small to avoid misjudgment. In experiments, we found that setting the threshold $\lambda$ to about 25% better reflects the facts.

4.2. Feature tracking based change detection

Since the feature points have been extracted when stitching key frames, the optical flow can be used to track the movement of feature points between the key frame and current frame, so as to judge whether the background has changed or the camera shifted. To reduce computation and speed up detection, we sort the feature points according to the response value and track the top 100 ones. Considering that areas with dense feature points may be blocked by moving objects, only when the tracking success rate is larger than 90%, we continue to calculate the square of the displacement of each feature point.

$$S_i = (p_i - p'_i)^2 + (p_i - p'_i)^2$$

(4)

where $p_i$ and $p'_i$ refers to the $i$th feature point on the key frame and its corresponding point tracked on current frame respectively. Since errors may exist in optical flow tracking, we consider the coordinates of the feature point has changed when $S_i > \mu$ (in experiments set to 9). Similar to 4.1, we calculate the change rate of the 100 feature points $p_{\text{feature change}}$. If the change rate reaches a certain threshold $\tau$ (in experiments set to 0.4), we consider the background changed or camera shifted. Figure 5 is an example that shows the optical feature tracking change detection. In figure 5(a), the green circles are the 100 good feature points, and in figure 6(b), the red circles are the successfully tracked points. Since the horizontal shift of camera in this example is much larger than 3, $p_{\text{feature change}}$ is close to 1.
4.3. Joint updating strategy

The grid vertex based detection method is fast and has uniform distribution of vertices, but it is not accurate enough and sensitive to threshold setting, which makes the probability of misjudgement higher than that of the feature point tracking method. In contrast, the feature point tracking method is more accurate but relatively time-consuming (100 feature points about 6ms). Moreover, the nonuniform distribution of feature points may lead to the failure of tracking when the dense area of feature point are blocked by moving objects. Therefore, we adopt a joint updating strategy as Algorithm 1 in Table 1:

Algorithm 1: Joint updating strategy

1. The background change detection is only performed when change in the stitching seam has been detected.
2. Make use of the real-time performance of grid vertex based detection method and “short-cut” rule of “||”: if change in background has been detected by \( p_{\text{vertex, change}} \), \( p_{\text{feature, change}} \) won’t be checked any more.
3. Since stitching parameters updating is time-consuming, so once updated, the next 5 frames will not be detected to keep the fluence of video.

Table 1. Joint updating strategy

| Condition          | Action                          |
|--------------------|---------------------------------|
| \( p_{\text{seam, change}} > \eta \) | \( \text{// checked each frame} \) |
| \( p_{\text{vertex, change}} > \lambda \) \& \( p_{\text{feature, change}} > \tau \) | update stitching parameters |
| else               | update stitching seam            |
| else               | pass                            |

5. Experiments and analysis

5.1. Test of automatic updating

The stitching program was implemented in C++ in Ubuntu 16.04 with Cmake 3.5.1, OpenCV 3.4 and CUDA 9.0. We mainly tested the automatic updating of stitching seam and parameters through stitching 2 channels of 640×480 video captured by USB cameras.

As shown in figure 6(a), frame 334 in the video stitched, a person was moving across the stitching seam and slight ghosting appeared as change rate had not reached the threshold, while in figure 6(b), frame 335, change rate exceeded the threshold, so the stitching seam was updated in time and worse ghosting was avoided.
5.2. Real-time performance analysis

To better understand the impact of stitching seam and parameter updating on real-time performance, the time consumption of each step is listed in Table 2. The values are measured when 2 channels of 640×480 video are stitched and may vary slightly in experiments.

Table 2. Time consumption of each step

| Step                          | No updating(ms) | Update seam(ms) | Update stitching parameters(ms) |
|-------------------------------|-----------------|-----------------|---------------------------------|
| feature extraction            | /               | /               | 65                              |
| feature matching              | /               | /               | 15                              |
| camera parameter estimating   | /               | /               | 5                               |
| image projecting              | /               | 20              | 20                              |
| exposure compensating         | /               | /               | 15                              |
| seam finding and saving       | /               | 60              | 60                              |
| total weight calculating      | /               | 7               | 7                               |
| seam change detection         | 10              | 10              | 10                              |
| background change detection   | /               | <6              | <6                              |
| other                         | 2               | 10              | 10                              |
| total                         | appr. 12        | appr. 110       | appr. 210                       |

Table 1 shows that each frame takes about 12ms when no updating is performed, so the frame rate can reach the capture frame rate (25 fps in our experiment), up to 80fps. When seam updating is performed, the frame rate falls to approximately 10fps, which will not obviously affect the fluency of the video. However, stitching parameters updating is relatively time-consuming. To avoid continuous updating, we stop detecting the next several frames once parameters updating is performed.

Figure 6. Test of automatic seam updating. (a) before updating, (b) after updating.

To test the automatic stitching parameters updating, we slightly moved the left camera during stitching. As shown in figure 7(a), frame 145, the camera had not been moved; in figure 7(b), frame 146, shift started and slight artifact appeared. Frame 147 was not stitched because background change was detected and started to update stitching parameters. Then in figure 7(c), frame 148, background alignment was repaired.

Figure 7. Test of automatic stitching parameter updating. (a) before shift. (b) after shift, before updating, (c) after updating.
6. Conclusion
In this paper, a fast and robust surveillance video stitching method with automatic stitching seam and parameters updating is proposed. While most frames are stitched based on the calculated stitching parameters, seam change caused by moving objects is detected by the change rate of points on the seam and background change is examined by the change rate of grid vertices or feature points, so as to determine whether to update them or not. Experiments demonstrate its robustness to various changes and good real-time performance.

Acknowledgments
The authors would like to thank the reviewers for their constructive comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grant 61573376.

References
[1] Brown M, Lowe D G. (2007) Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision, 74(1):59-73.
[2] Gao J, Kim S J, Brown M. (2011) Constructing image panoramas using dual-homography warping. In: Computer Vision and Pattern Recognition. Colorado.pp.49-56
[3] Lin W Y, Liu S, Matsushita, et al. (2011) Smoothly varying affine stitching. In: Computer Vision & Pattern Recognition. Colorado.pp.345-352
[4] Zaragoza J, Chin T, Tran Q, et al. (2014) As-Projective-As-Possible Image Stitching with Moving DLT[J]. IEEE Trans Pattern Anal Mach Intel, 36(7):1285-1298.
[5] Chang C H, Sato Y, Chuang Y Y. (2014) Shape-Preserving Half-Projecting Warps for Image Stitching. In IEEE Conference on Computer Vision and Pattern Recognition. Columbus.pp.3254-3261
[6] Chen Y S, Chuang Y Y. (2016) Natural Image Stitching with the Global Similarity Prior. In European Conference on Computer Vision. Springer International.Amsterdam.pp.186-201
[7] Gao, J, Li, Y, Chin, T.J. (2013) Seam-Driven Image Stitching[C]. In: Eurographics. Tutorials.pp.45-48
[8] Lin K, Jiang N, Cheong L F, et al. (2016) SEAGULL: Seam-Guided Local Alignment for Parallax-Tolerant Image Stitching. In: European Conference on Computer Vision. Amsterdam.pp.370-385
[9] Liu H, Tang C, Wu S. et al. (2011) Real-Time Video Surveillance for Large Scenes. In: International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, pp. 1–4.
[10] He B, Zhao G, Liu Q. (2012) Panoramic video stitching in multi-camera surveillance system[C]// Image & Vision Computing New Zealand. pp.1-6.
[11] BOTAO H, SHAOHUA Y. (2015) Parallax-Robust Surveillance Video Stitching. Sensors.16(1): 7-18
[12] Yin X Q, Li W L, Wang B, et al. (2014) A Novel Video Stitching Method for Multi-Camera Surveillance Systems. KSII Transactions on Internet and Information Systems.8(10) :3538-3556.
[13] Bay H. (2006) SURF: Speeded Up Robust Features. Computer Vision & Image Understanding, 110(3):404-417.
[14] Fischler M A. (1981) Random sample consensus. Comm of Acm. 24(6): 381-395.