Real-Time Multi-camera Video Stitching Based on Improved Optimal Stitch Line and Multi-resolution Fusion

Dong-Bin Xu[1,2](✉), He-Meng Tao[3], Jing Yu[3], and Chuang-Bai Xiao[3]

[1] Research Institute of Highway Ministry of Transport, Beijing 100080, China
[2] RIOH Transport Consultants LTD., Beijing 100191, China
db.xu@rioh.cn
[3] College of Computer Science and Technology, Beijing University of Technology, Beijing 100124, China

Abstract. In this paper, we propose a multi-camera video stitching method based on an improved optimal stitch line and multi-resolution for real-time application. First, phase correlation is used to estimate overlapping fields between multiple videos, where SURF feature points are extracted for image registration to save computation time and improve the matching accuracy. Then, a fusion algorithm combining the improved optimal stitch line and multi-resolution algorithms is devised to improve visual effects by eliminating ghosting and any visible seams. In the fusion stage, GPU acceleration is employed to speed up the video stitching. Experiments show that the proposed algorithm has better and real-time performance compared with traditional video stitching methods.

Keywords: Multi-camera stitching · SURF feature · Stitch line · GPU

1 Introduction

Video stitching is one of the most popular topics in digital image processing and computer vision, where real-time video stitching still remains a challenge. Video images being stitched are often with significantly varying viewpoints and viewing directions [1]. In real scene, video stitching algorithms should also be sufficiently automatic, robust and fast.

Video stitching generally falls into two categories, namely, video sequence stitching and multi-camera real-time video stitching. In video sequence stitching, image processing techniques are used to stitch an existing sequence of videos with the same view into a panorama video. Because of offline processing and allowing a long period of background processing, it usually gives satisfactory results but it takes too much time. Xu and Mulligan [2] attempted to stitch a 45-minute video sequence (1920×1080) from three cameras. Even with his improved video sequence stitching method, it takes five days to complete the stitching. Multi-camera real-time video stitching aims to stitch videos collected from multiple live cameras in real-time and produces a panorama video. The stitching algorithms should be fast enough for real-time. For example, to outputs a 10-fps video, the fusion process must be completed within 100 ms.

The paper mainly deals with the feature-based real-time stitching techniques for panorama videos, which involves the use of image processing and multi-pipeline parallel programming model. To overcome the seams in overlapping regions, a method based on improved optimal stitch line is proposed. Some algorithms are used to improve the real-time performance including matching points obtained from a circular region, multi-resolution fusion algorithms and especially GPU acceleration of CUDA [3] programming in fusion stage.

2 Determining Overlapping Regions

Most traditional SURF [4] algorithms are time-consuming because they detect features from the whole image. To solve this problem, the phase correlation algorithm is firstly used to calculate the approximate overlapping region between the two stitched images. The procedures of phase correlation are described as follows:

If a displacement $(\triangle x, \triangle y)$ exists between image $I_1(x,y)$ and $I_2(x,y)$, then the relations between them can be written as:

$$I_1(x,y) = I_2(x - \triangle x, y - \triangle y) \tag{1}$$

After normalization, the cross power spectrum is:

$$\frac{\hat{I_1}(u,v)\hat{I_2}^*(u,v)}{|\hat{I_1}(u,v)\hat{I_2}^*(u,v)|} = e^{-j2\pi(u\triangle x + v\triangle y)} \tag{2}$$

where $\hat{I_1}(u,v)$ and $\hat{I_2}(u,v)$ are the results of Fourier transform of $I_1(x,y)$ and $I_2(x,y)$,while $\hat{I_1}^*(u,v)$ and $\hat{I_2}^*(u,v)$ is the complex conjugate of $I_1(x,y)$ and $I_2(x,y)$. By performing inverse Fourier transform for Eq. (2), the results are shown as

$$\delta(x - \triangle x, y - \triangle y) = F^{-1}[e^{-j2\pi(u\triangle x + v\triangle y)}] \tag{3}$$

The position of the peak value of the function corresponds with the translational parameter $(\triangle x, \triangle y)$ about the two images. If there is only a translational relationship between the two images, then the peak value of the function shows the extent of correlation between them, where the range of its value is $[0, 1]$. Through the translational parameters $(\triangle x, \triangle y)$, the approximate overlapping region between the two images can be obtained.

By detecting feature points only in the overlapping region, the feature points detection algorithm is accelerated because of fewer feature points being searched, and the accuracy is also improved in subsequent matching. After the images are matched, the RANSAC algorithm is used to eliminate mismatched pairs of points and calculate the coordinate transformation matrix between video images.

3 Image Fusion

3.1 Exposure Adjustment

A problem of automatic image stitching is exposure differences between images, which will bring seams when the images are blended within overlapping regions. To eliminate exposure differences between the stitched images, a block-based exposure adjustment method is adopted, which is proposed by Uyttendaele in 2001 [5]. Image is divided into 32 × 32 blocks. Within each block, compute a quadratic transfer function in a least-squares sense, this image block to the composite of images overlapping this block. To smooth the variation in the transfer function distributions, use a combination of two techniques. First, average the functions in each patch with those of their neighbors, use an iterated separable kernel of (1/4, 1/2, 1/4), and typically use 2 iterations. Second, for each pixel, the results are blended by applying the transfer function from neighboring patches, which can be performed using bilinear interpolators as shown in Eq. (4). The results are the same as interpolating a separate function for each pixel, but since we implement the transfer function using a look-up table, the computational cost is much smaller.

$$p(x,y) = \sum_{v=-1}^{v=1} C_v(\frac{y}{N}) \sum_{u=-1}^{u=1} C_u(\frac{x}{N}) f_{m+u,n+v}(p(x,y)) \tag{4}$$

where $p(x,y)$ is the pixel value at block location, N is the block size, $C_n(x)$ is the bilinear coefficient, $f_{u,v}(x)$ is the transfer function of block u and v.

3.2 Optimal Stitch Line

The optimal stitch line [6–8] is based on a notion that there exists a stitch line in the overlapping region, which can be used to minimize the difference in both color and geometry between the two images, so that only pixels on one side of this stitch line are needed for producing a panorama image. Hence, the principle of finding an optimal stitch line is:

$$E(x,y) = E_{color}(x,y)^2 + E_{geometry}(x,y) \tag{5}$$

where E_{color} represents the color difference of pixels while $E_{geometry}$ represents the geometric difference of pixels in the overlapping region.

According to the principle of dynamic programming, the detailed steps are as follows:

(1) Establish stitch lines with all pixels in the first row as the starting points. Calculate the intensity value E of each pixel. Then, the stitch line extends to the next row of pixels.
(2) Compare the intensity values of three pixels in the next row neighboring to the current point of the stitch line. Take the pixels with the smallest intensity value as the direction of the extension. And then, this point becomes

the current point of the stitch line. Update the total intensity value of the stitch line and make the current point the one corresponding to the smallest intensity value of the next row.

(3) Select a stitch line with the smallest total intensity value out of all stitch lines as the best stitch line.

To reduce the visibility of the seam, we propose an improvement to the basic algorithm based on the optimal stitch line between the neighboring images $I_1(x, y)$ and $I_2(x, y)$.

First, define binary images $R_1(x, y)$ and $R_2(x, y)$denoting the initial weight matrix for the two neighboring images $I_1(x, y)$ and $I_2(x, y)$. Let the values of the pixels on the two sides of the stitch line be 1 and 0, respectively.

Second, define a distance transform function to calculate the distances from all non-zero pixel points to the nearest zero pixel point, where p is the set of non-zero pixels and q is the set of zero pixels.

$$dis(p(x_1, y_1), q(x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2| \tag{6}$$

Then, calculate the new transition fusion weight $\alpha_1(x, y)$ and $\alpha_2(x, y)$ for corresponding images $I_1(x, y)$ and $I_2(x, y)$, a threshold value $\varepsilon \in (0, 1]$ is used to set the size of the smooth transitional zone. The equation is written:

$$\begin{cases} \alpha_1(x, y) = \varepsilon^* R_1(x, y) & if \varepsilon^* R_1(x_0, y_0) > 1, \alpha_1(x_0, y_0) = 1 \\ \alpha_2(x, y) = \varepsilon^* R_2(x, y) & if \varepsilon^* R_2(x_0, y_0) > 1, \alpha_2(x_0, y_0) = 1 \end{cases} \tag{7}$$

Finally, calculate the fused image:

$$I_{res} = \frac{\alpha_1(x, y) * I_1(x, y) + \alpha_2(x, y) * I_2(x, y)}{\alpha_1(x, y) + \alpha_2(x, y)} \tag{8}$$

Bias in the registration process before stitching and object movements may lead to ghosting and distortion in the overlapping region. Using the optimal stitch line method can avoid generating a stitch line in the region with high contrast. Instead, the optimal stitch line often exists in a region with a smooth transition. Hence, pixels are selected for value assignment from a region on each side of the optimal stitch line, respectively, which helps avoid generating ghost.

3.3 Multi-resolution Image Fusion

To ensure that the transition in the overlapping region is as smooth as possible, the paper proposes multi-resolution fusion method based on Gaussian pyramid. Gaussian pyramid is a series of low-pass filtering images obtained by convoluting an image and a weight function. Let G_0 be the initial image, then

$$G_l(i, j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) G_{l-1}(2i + m, 2j + n) \tag{9}$$

where $1 \leq l \leq N, 0 \leq i \leq R_i, 0 \leq j \leq C_l$, R_i and C_l represent the numbers of rows and columns in level l, respectively. The two-dimensional discrete Gaussian convolution kernel function is denoted by $w(m, n)$.

In order to get the multi-band-pass filtered image required for multi-band fusion, we subtract the next level from each level of the pyramid. Since these images have different sampling precision, image interpolation is required to obtain new samples:

$$G'_l(i,j) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} (\frac{2i+m}{2}, \frac{2j+n}{2}) \tag{10}$$

Suppose that the image on Level l of the Laplacian pyramid is L_l and the top-level image is L_N, then the formula for constructing the Laplacian pyramid is:

$$\begin{cases} L_l = G_l - G'_{l+1} & 0 \leq l < N \\ L_N = G_N & l = N \end{cases} \tag{11}$$

Similarly, as opposed to the construction process, image reconstruction can be written as:

$$\begin{cases} G_l = L_l + G'_{l+1} & 0 \leq l < N \\ G_N = L_N & l = N \end{cases} \tag{12}$$

4 GPU-accelerated Fusion

Image mosaic algorithms cannot be usually used in real-time applications because it is time-consuming. Since the algorithm part has now been optimized, this paper tries to solve the problem through programming models. By using multithreading principles and novel GPU programming models to optimize the image, it proposes a real-time panorama video stitching approach that features good visual effects and a satisfactory frame rate.

The first group of synchronous frames are used to obtain the transformation matrix H that performs projection transformation for the image. Computation relying solely on CPU takes a long time and is therefore unable to process image frames in real-time. To solve this problem, the task of image coordinate transformation is assigned to GPU. Since the projection transformation performs the same coordinate transformation for all pixels in the whole image and copies color values of pixel points, this process has good parallelism.

Suppose that during feature matching, the matched feature points obtained are $p_1^i = (x, y)$ and $p_2^j = (x', y')$. According to the pinhole imaging theory, a point in a three-dimensional space corresponds to pixels in different positions in image $I_1(x, y)$ and $I_2(x, y)$. So, there is a one-to-one relationship between them. Image matching can be conducted through the perspective projection mapping function and a 3×3 homography matrix. The matrix is used to calculate the position of a point from a three-dimensional space projected on different two-dimensional images, which is an one-to-one mapping. Its 8-parameter matrix is written as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \approx \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

and simplifying gives:

$$\begin{aligned} x' &= \frac{a_1 x + a_2 y + a_3}{c_1 x + c_2 y + 1} \\ y' &= \frac{b_1 x + b_2 y + b_3}{c_1 x + c_2 y + 1} \end{aligned} \tag{14}$$

To avoid blanks in the image obtained after coordinate transformation, the inverse process of coordinate transformation is performed, which includes the following steps. First, find each pixel in the original image corresponding to that in the image obtained after coordinate transformation and then assign its color value to the pixels in the transformed image.

$$\begin{aligned} x &= \frac{(b_2 - b_3 c_2)x' + (a_3 c_2 - a_2)y' + a_2 b_3 - a_3 b_2}{(b_1 c_2 - b_2 c_1)x' + (a_2 c_1 - a_1 c_2)y' + a_1 b_2 - a_2 b_1} \\ y &= \frac{(b_1 - b_3 c_1)x' + (a_3 c_1 - a_1)y' + a_1 b_3 - a_3 b_1}{(b_2 c_1 - b_1 c_2)x' + (a_1 c_2 - a_2 c_1)y' + a_2 b_1 - b_1 a_2} \end{aligned} \tag{15}$$

Conduct the REDUCE operation over the transformed image so as to generate a Gaussian pyramid. Perform the EXPAND operation and let neighboring levels subtract each other, which produces a Laplacian pyramid. Fuse images on each Laplacian pyramid level by using the optimal stitch line algorithm. At last, perform the COLLAPSE operation on the image, expand and accumulate the target image to generate the stitched image.

5 Results

An experiment is conducted to stitch the videos collected from three live cameras in real time. The hardware is as follows: computer CPU with Intel i5-4590 3.30 GHz and 8 Gb memory, GPU with NVIDIA Quadro K620.

Image (a), (b) and (c) in Fig. 1 are video images collected from three cameras with their resolution of 816×612. The displacement from Image (a) to (b) is $(565, -30)$. The displacement from Image (b) to (c) is $(558, 26)$. Image (d), (e) and (f) are the SURF feature points extracted from the overlapping region among the three images.

Figure 2 shows the matched images based on the nearest and second nearest neighbor method. As shown in Fig. 3, image (b) is the output of image (a) after exposure adjustment. The distance transform weighted map is shown in Fig. 4.

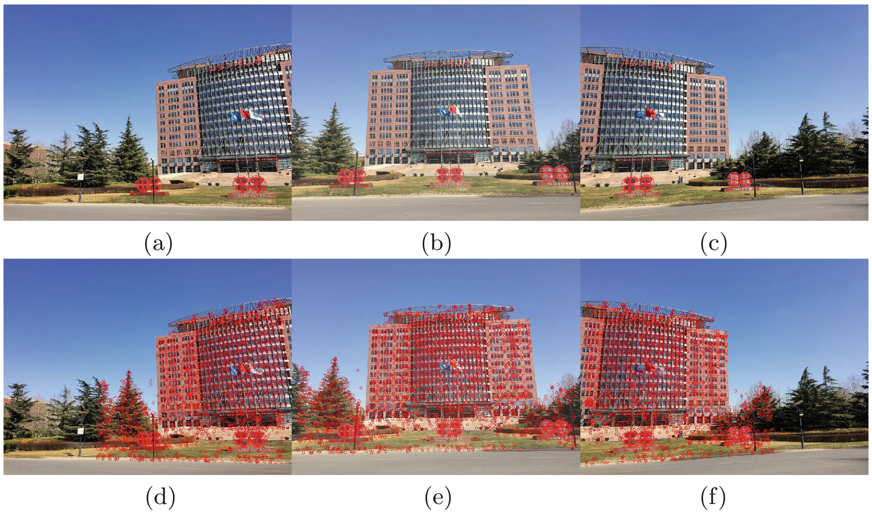

(a) (b) (c)
(d) (e) (f)

Fig. 1. Surf keypoints of multiple video images

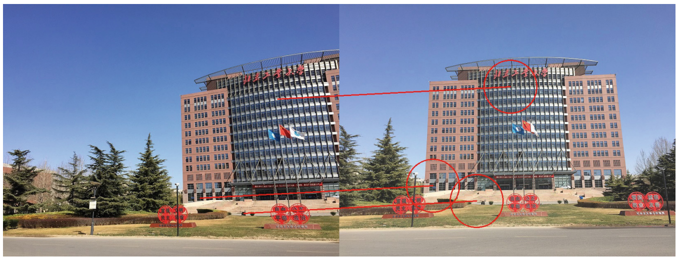


Fig. 2. The schematic plot of matched feature points

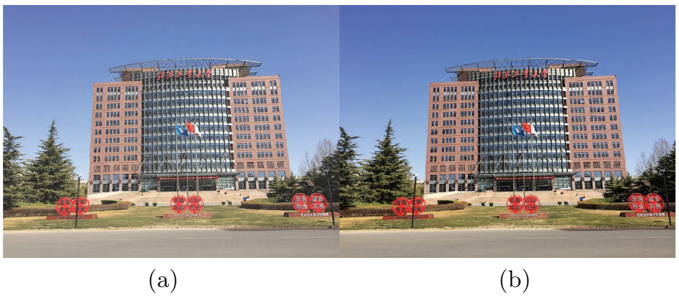
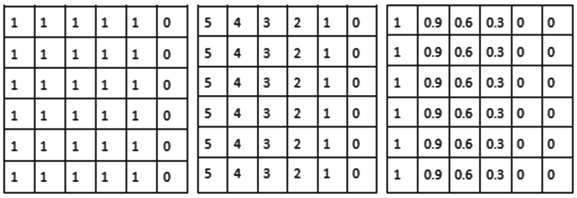

(a) (b)

Fig. 3. The schematic plot of exposure adjustment



1	1	1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	0

5	4	3	2	1	0
5	4	3	2	1	0
5	4	3	2	1	0
5	4	3	2	1	0
5	4	3	2	1	0
5	4	3	2	1	0

1	0.9	0.6	0.3	0	0
1	0.9	0.6	0.3	0	0
1	0.9	0.6	0.3	0	0
1	0.9	0.6	0.3	0	0
1	0.9	0.6	0.3	0	0
1	0.9	0.6	0.3	0	0

Fig. 4. The distance transform weighted map

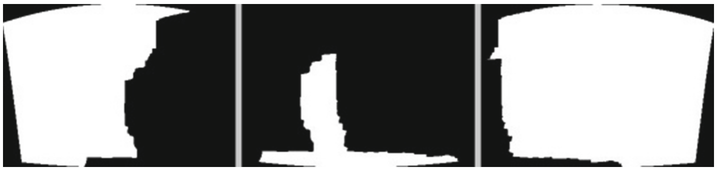


Fig. 5. The mask of the best stitch line

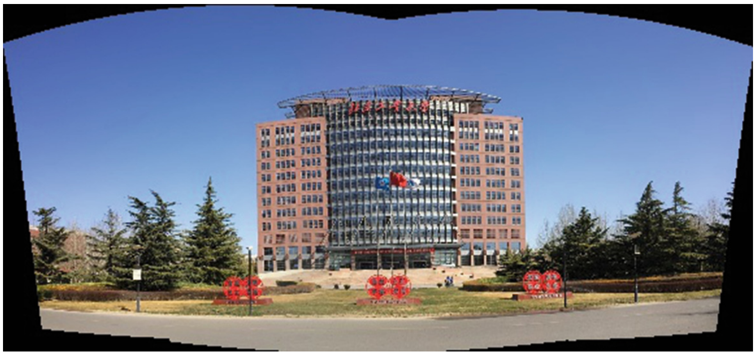


Fig. 6. The result of multi-resolution image fusion

Figure 5 shows the mask of the best stitch line. Figure 6 shows the result of multi-resolution image fusion.

The data collected from different stages of the experiment is shown in Tables 1 and 2. It can be seen from the data in Table 1 that using the phase correlation algorithm to estimate the overlapping region can effectively limited the number of detected feature points, improve the matching accuracy in subsequent steps, and boost matching efficiency. As shown in Table 2, the real-time stitching stage only involves the homography matrix H of space projection transformation, exposure adjustment parameters, and fusion matrix obtained during the offline stage. Since projection transformation and the fusion stage require relatively complex computation for each pixel in the panorama image the total time it takes for each frame on projection transformation, fusion, and color calibration using color parameters averages about $13 + 9 + 28 = 50$ ms with a frame rate of 20 fps.

Table 1. The performance of using Phase Correlation or not

Implementation	No Phase Correlation (ms)	With Phase Correlation (ms)
Translation parameters Calculation	N/A	936
SURF Points Extraction	1953	1470
SURF Points Matching	1246	171
The number of SURF Points	3256	2602
Total Time	3199	2577

Table 2. The performance of using CUDA or not

Implementation	No CUDA (ms)	With CUDA (ms)
Projection transformation	221	13
Exposure adjustment	9	9
Image fusion	38	28

Without the use of the CUDA parallel architecture, it would take about $221 + 9 + 38 = 268$ ms for each frame, with a rate of only 3.7 fps.

6 Conclusions

The paper proposes a real-time multi-video stitching approach based on improved stitch line and multi-resolution fusion. Multiple live cameras are used to collect videos in real time. Firstly, phase correlation is adopted to calculate translation parameters between different video images from multiple live camera. Secondly, under the guidance of the parameters, the overlapping region between images is estimated, and then SURF feature points are for extracted only in overlapping region to save computation time and improve matching accuracy. Thirdly, by considering geometry and color, the heuristic search algorithm is used to find the optimal stitch line to eliminate seams in fusion stage, perform distance function transformation for the nearby area, and obtain the weight matrix. Finally, the multi-resolution image fusion is performed to deliver a smooth and seamless panorama video stream. CUDA is used to accelerate projection transformation and image fusion so that the video stitched by the system achieves a frame rate of 20 fps.

References

1. Haenselmann, T., Busse, M., Kopf, S., King, T., Effelsberg, W.: Multi perspective panoramic imaging. Image Vis. Comput. **27**(4), 391–401 (2009)
2. Xu, W., Mulligan, J.: Panoramic video stitching from commodity HDTV cameras. Multimed. Syst. **19**(5), 407–426 (2013)

3. Nvidia Corperation.: CUDA C Programming Guide. http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html. Accessed 26 May 2017
4. Bay, H., Ess, A., Tuytelaars, T., et al.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
5. Uyttendaele, M., Eden, A., Seliski, R.: Eliminating ghosting and exposure artifacts in image mosaics. In: Proceedings of IEEE Conference Computer Vision and Pattern Recognition, vol. II, pp. 509–516. Prague, Czech Republic (2001)
6. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **1**(11), 1222–1239 (2001)
7. Kerschner, M.: Seamline detection in colour orthoimage mosaicking by use of twin snakes. ISPRS J. Photogramm. Remote Sens. **56**(1), 53–64 (2001)
8. Kwatra, V., Schödl, A., et al.: Graphcut textures: image and video synthesis using graph cut. ACM Trans. Graph. **22**(3), 277–286 (2003)