REAL-TIME VIDEO STITCHING

Shuo-Han Yeh and Shang-Hong Lai

Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

ABSTRACT

This paper presents a real-time video stitching system which can stitch videos acquired from multiple moving cameras. Most conventional stitching methods were developed under the assumption of static scenes without considering moving objects. However, cameras could move freely for general video stitching and the homography matrices between different camera views could vary from frame to frame. The proposed algorithm estimates the refined homography in both spatial and temporal domains. To be more specific, we first estimate homography between images acquired by different cameras by using RANSAC in the spatial domain at some key frames separated by a fixed interval. Then, we obtain temporally smooth homography transformations by using temporally linear interpolation for the frames between the key frames. For the image stitching, we correct the exposure of the stitched image by linear blending in the overlapping region and generate a panoramic view with cylindrical warping. To achieve real-time video stitching, we speed up our system with CUDA parallel programming. The experimental results show that the stitched views by using the proposed algorithm compare favorably with other methods. In addition, our video stitching system can achieve real-time performance and it is much more efficient compared to the other methods included in our experimental comparison.

Index Terms— Image stitching, video stitching, panoramic video

1. INTRODUCTION

The goal of this paper is to develop an accurate and real-time video stitching system that can stitch several videos acquired from several cameras with overlapping fields of view in real time. One possible application is to stitch the videos acquired from several cameras mounted on a remotely controlled car into a panoramic video, and then transmit the panoramic video to the control center for the driver to visualize the road conditions in real time. This system could be applied to shuttle vehicles and car sharing system, and it may also be potentially applied to virtual reality and tele-presence.

Different from image stitching, video stitching contains temporal information and may involve complex changes in the scenes, such as moving objects. The motions between cameras also affect the stitched view. As mentioned in [1], the unfixed optical centers of cameras and objects moving across cameras are the challenges to video stitching. Besides, parallax problem exists when objects are close to cameras. Therefore, video stitching is a challenging task, which is not just applying image stitching for every frame. Another challenge is that projective transformation usually produces shape distortion in non-overlapping regions, which cause wrong depth perception to the viewer. The other difficulty is that exposures for images captured by different cameras may be inconsistent due to different light directions, thus making the video stitching more challenging.

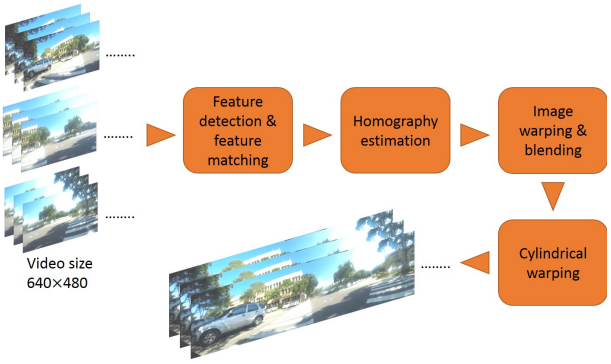

Fig. 1. The pipeline of our algorithm.

Our work first detects feature points by SURF, and then feature matching is accomplished by the KNN method. RANSAC is subsequently applied to estimate homography transformation from the extracted feature pairs in the spatial domain. A smooth homography transformation depends on the current and past homography matrices by using the moving average model in the temporal domain. Next, linear blending is along the horizontal direction in the overlapping regions to adjust exposures. In our experiments, we stitch three videos acquired from three cameras placed on a linear setting and assume the middle view is the reference view. We apply cylindrical warping to simulate the situation that the user is at the center of the panorama view. On the basis of [2], cylindrical warping could improve the perceptual experience. Considering the real-time constraint on the video stitching, we need to find a trade off between alignment accuracy and computation efficiency. In addition to GPU-based feature

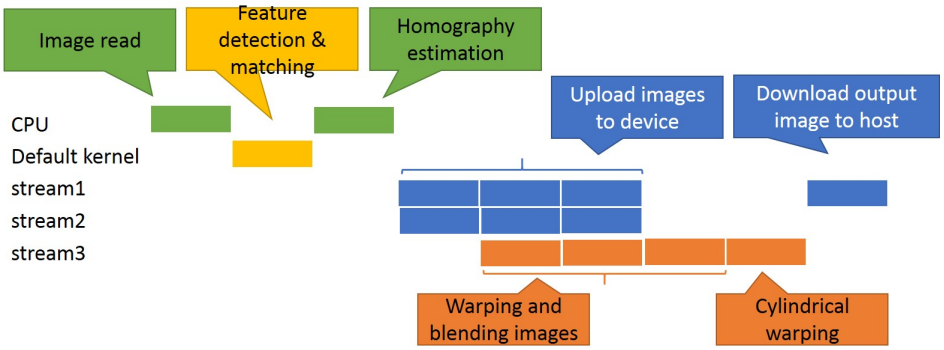

Fig. 2. Kernel timeline: Feature detection and matching are implemented by OpenCV library. Homography estimation is implemented in CPU. The first and second streams are used to upload data to the third stream for warping and blending images block-by-block. While the first and second streams are uploading data, the third stream can still execute the assigned jobs concurrently.

extraction, CUDA parallel programming is applied to speed up our video stitching system to achieve real-time stitching. In order to reduce computational time, we run streams to upload data to device while calling kernel on the streams concurrently. In this way, we can accelerate the speed of the video stitching system.

2. RELATED WORK

Traditional image stitching algorithm estimates a 2D homography transformation between two images [3, 4]. These works assumed that cameras were set in co-axis or the camera motion is simply rotation. Nevertheless, when objects are close to cameras or cameras are placed far away, input images are accompanied with large parallax effect. In this situation, post-processing methods, such as deghosting and blending, can be used to alleviate the artifacts.

Distortion in non-overlapping regions results in visually unpleasant stitching results. [5] proposed that smoothly extrapolating the projective transformation of the overlapping regions into the non-overlapping regions could improve the perceptual level. [6] constructed a seamless panorama view by using dual-homography warping which separated an input image into a distant plane and a ground plane. In recent years, local homography transformations help to deal with parallax. By using the distance from a pixel to the boundary between the overlapping and the non-overlapping regions, [7] constructed a weight map to settle local warps. Content preserving warping is also used to minimize the image distortion [8].

To achieve efficient and temporally smooth video stitching, we propose to estimate the homography transformations between different videos at some key frames sampled at a fixed interval, and then apply temporally linear interpolation to estimate the homography transformations for all the frames inside the intervals. In addition, the linear blending method is applied to alleviate the problem of different exposures between frames acquired from different cameras. Cylindrical warping is employed to generate the panoramic view of the stitched images and reduce the perspective image distortion.

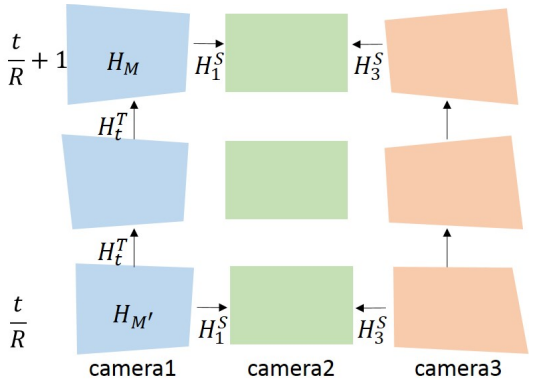

Fig. 3. Consider stitching three videos, and the middle camera, Camera2, is the reference view. We first use RANSAC to optimize the spatial homography estimation at key frames. Then, by using temporally linear interpolation, we can generate temporally smooth homography transformations for video stitching.

3. PROPOSED METHOD

In this paper, we assume the cameras are placed roughly along a horizontal line with overlapped field of views between neighboring cameras. Our video stitching system framework includes feature detection, spatial and temporal homography estimation, image warping, blending, and cylindrical warping as depicted in Figure 1

Fig. 4. Experimental results for video stitching with and without using temporal interpolation for homography estimation. The left column is with temporal interpolation while the right column is the result of homography estimation at every frame. Each column means 3 consecutive frames over time. The locations for the red blocks are fixed for all frames. The stitched images in the left column appear gradual changes along time, while the trees in the red blocks at the right column vary largely.

3.1. Feature detection and matching

Based on the linear camera setting, We first partition an input image into two parts and detect feature points for both parts. Then, we can restrict the feature point matching between two views based on the left/right partitioning. This way not only reduces mismatching pairs but also reduces the computational time, since the left part of the middle view only corresponds to the right part of the left view, and the right part of the middle view corresponds to the left part of the right view. Moreover, we take advantage of GPU SURF in default kernel to speed up the computation and use the KNN feature matching.

with temporal interpolation	without temporal interpolation
0.038 sec	0.14 sec

Table 1. Execution time per frame for video stitching with and without temporal interpolation.

3.2. Spatial and temporal homography estimation

We estimate the homography transformation from feature point correspondences in the spatial domain by using RANSAC to remove outliers. The equation is given as follows:

$$\arg\min_{H^S} \sum_{k=1}^{n} \left\| X_i^r - H^S X_k^t, \right\| \tag{1}$$

where $X_k^r = [x_k^r, y_k^r, 1]^\top$ and $X_k^t = [x_k^t, y_k^t, 1]^\top$ are the k-th feature points in homogeneous coordinate of the reference frame and target frame. H^S is the spatial homography. K is the total number of feature point correspondences. For the purpose of real-time system, we repeat the image alignment step after a period of time. However, sudden changes of scene cause unacceptable views for users. In order to reduce severe changes between homography transformations, we utilize the concept of moving average in temporal domain. We calculate the average homography transformations from the recorded spatial homography transformations.

Besides, to further improve the smoothness of perceived scene, we use linear interpolation to compute the homography transformation. The smoothed homography is:

$$H_M = \begin{cases} \frac{\sum_{i=0}^{i=n} H^S_{\frac{t}{R}-i}}{n+1}, \frac{t}{R} \in N \\ H^S \qquad\qquad , \frac{t}{R} < 1 \end{cases} \tag{2}$$

where H_M is moving average homography which takes previous homography into consideration, thus n is the number of reference homography. R is the interval that a new spatial homography needs to be computed. We now construct homography by linear interpolation as follows:

$$H_t^T = \begin{cases} H_{M'} + \frac{H_M - H_{M'}}{R}(t - M'), M' \le t \le M \\ H_M \qquad\qquad\qquad\qquad , t \le R \end{cases} \tag{3}$$

where $H_{M'}$ is the previous moving average homography.

3.3. Image warping and blending

The target views are aligned to the reference view by image warping with the estimated homography. Performing linear blending in x coordinate only in the overlapping region shows that luminance gradually changes in the panorama view in column space. In the warping and blending steps, we make use of kernel concurrency. A stream can be a sequence of kernel launches and host-device memory copies. We divide the process into three parts by streams: memory copy from host(CPU) to device(GPU), execute GPU functions and memory copy from device to host. Figure 2 shows the kernel timeline. We use the first and the second streams to upload images and homography information to GPU memory. As soon as one of images and homography have uploaded, GPU can

Fig. 5. We compare our results with openCV [2, 3, 9, 10, 11, 12], PTGui [13] and AutoStitch [14]. The second row shows our results from the input images (1a), (2a). We have better alignment than OpenCV (1c,2c), PTGui (1d,2d) and AutoStitch (1e,2e).

	Our results	OpenCV	PTGui	AutoStitch
testcase1	0.036	0.040	1.0	1.79
testcase2	0.038	0.043	0.94	1.68

Table 2. Time table. The computational time is reported in seconds per panorama.

start warping and blending by the third stream. GPU performs image blending block-by-block rather than pixel-by-pixel. In this way, we can save a considerable amount of time.

To alleviate the perspective distortion in the stitched image, we apply cylindrical warping to generate the panoramic view of video stitching. For real-time stitching, the cylindrical warping on the stitched images is implemented in the kernel.

4. EXPERIMENTAL RESULTS

In our experiments, we stitch three videos acquired from three cameras mounted on a vehicle. The optical distortion and uneven luminance problems are quite obvious in the testing videos. By applying linear blending and cylindrical warping, we generate a panoramic view for the stitched video. We show some sampled frames of the stitched videos by using different methods in Figure 4 and 5. We compare the stitched results by using our method with and without using temporal interpolation for the homography estimation. Figure 4 depicts the stitched panoramic views. From the example shown in Figure 4, we can see that our video stitching algorithm with the temporal interpolation for homography provides more temporally consistent video stitching results than that without using temporal interpolation. The execution time table for the two cases is also supplied in Table 1. It is evident that the proposed algorithm with temporal interpolation for homography is more efficient compared to the one without applying temporal interpolation. [1].

We conducts experiments on GTX 970 mini. Our input video size is 640×480 and the output video is 1700×480. In Figure 5, we compare our results with the results by using OpenCV [2, 3, 9, 10, 11, 12], software PTGui [13] and AutoStitch [14]. We depict two sample stitching results in Figure 5. In these two cases, it is obvious that the proposed algorithm provides more accurate stitching than the other three methods. The red block indicates ghost artifacts in the corresponding region in the panoramic images. For the example on the right, AutoStitch and PTGui failed to stitch the third view automatically, but we manually select feature points in PTGui to accomplish the stitching. In addition, the computational time for all the methods included in the comparison is summarized in Table 2. From the execution time comparison, we can see that our algorithm is fastest and it is close to real-time video stitching.

5. CONCLUSION

In this paper, we propose a real-time and temporally consistent video stitching algorithm. Our experiments show that the proposed video stitching algorithm provides better image alignment than other methods. Moreover, the streams used to upload data and calling kernel concurrently can speed up the execution time. By using CUDA parallel programming, we can achieve real-time video stitching.

[1] http://cv.cs.nthu.edu.tw/php/2017ICIP/

6. REFERENCES

[1] W. Jiang and J. Gu, "Video stitching with spatial-temporal content-preserving warping," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 42–48.

[2] R. Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[3] 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.

[4] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *Annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 251–258.

[5] C.-H. Chang, Y. Sato, and Y.-Y. Chuang, "Shape-preserving half-projective warps for image stitching," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 3254–3261.

[6] J. Gao, S. J. Kim, and M. S. Brown, "Constructing image panoramas using dual-homography warping," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 49–56.

[7] C.-C. Lin, S. U. Pankanti, K. N. Ramamurthy, and A. Y. Aravkin, "Adaptive as-natural-as-possible image stitching," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 1155–1163.

[8] F. Zhang and F. Liu, "Parallax-tolerant image stitching," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3262–3269.

[9] M. Uyttendaele, A. Eden, and R. Skeliski, "Eliminating ghosting and exposure artifacts in image mosaics," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2001, vol. 2, pp. II–II.

[10] W. Xu and J. Mulligan, "Performance evaluation of color correction approaches for automatic multi-view image and video stitching," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 263–270.

[11] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.

[12] H.-Y. Shum and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," in *Panoramic vision*, pp. 227–268. Springer, 2001.

[13] "Ptgui," `https://www.ptgui.com/`.

[14] "Autostitch," `http://matthewalunbrown.com/autostitch/autostitch.html`.