International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 37340-37345
© Research India Publications. http://www.ripublication.com

Real-time Spatial Video Panorama using Iterative Compositing

P. Venkat Rangan
Professor, Amrita Center for Wireless Networks and Applications,
Amrita Vishwa Vidyapeetham, Kollam, Kerala, India
venkat@amrita.edu

Balaji Hariharan,
Assistant Professor, Amrita Center for Wireless Networks and Applications,
Amrita Vishwa Vidyapeetham Kollam, Kerala
balajih@am.amrita.edu

G Uma
Research Associate, Amrita Center for Wireless Networks and Applications,
Amrita Vishwa Vidyapeetham, Kollam, Kerala, India
umag@am.amrita.edu

Ramkumar N,
Research Associate, Amrita Center for Wireless Networks and Applications,
Amrita Vishwa Vidyapeetham Kollam, Kerala
ramkumar@am.amrita.edu

Rahul Krishnan
Research Associate, Amrita Center for Wireless Networks and Applications,
Amrita Vishwa Vidyapeetham Kollam, Kerala, India
rahulkrishnan@am.amrita.edu

Abstract
Real-time stable panoramic video stitching is a challenging research problem. Reducing computational complexity of stitching algorithm, increasing the quality of the stitched video and a stable repeatable technique for stitching are the major challenges addressed in this paper. We present two major techniques: iterative compositing and pipelined execution. In a scenario where cameras are static, this algorithm dynamically measures the redundancies in the feature detection, feature matching and transformation model estimation stages and reuses the resultant transformation matrices for stitching the subsequent frames. In our experiments, we have determined that our unique algorithm is able to improve the real-time stitching of standard definition videos by a factor of 4.5. Moreover, the quality of the video was found to be satisfactory, devoid of any visible seams, very low jitter and no other visual artifacts.

Keywords—Video stitching, real-time, blending and warping, GPU parallelization, iterative compositing, pipelining

Introduction
Over the last two decades, lot of research has gone into the area of multimedia. The various modalities, both video and audio has undergone tremendous development. Improvements in the quality of video, in terms of resolution of capturing devices, specialized multimedia storage devices, large increase in the transmission bandwidth, exponential improvements in the compression and decoding techniques, advent of GPU processing and more immersive display devices have proliferated into homes, offices and mobile devices. These improvements in quality of media, devices and processes have made it possible to offer immersive applications in the field of entertainment, healthcare, education, data analysis etc.
Currently, the most immersive experience that the users have is a recreated world of visually stimulating scenes, like an iMax movie, a virtual airplane training simulation, a live soccer match with online editing. In terms of visual presentation the research has been focused on improving the quality of display devices, encoding and decoding algorithms and reducing computational time for high-end graphics applications. The various techniques used to improve the immersive experience have been limited to increasing the graphical resolution, and improving the video production techniques such as merging different scenes, simultaneous display of scenes in big and small windows. These effects are introduced through offline editing with manual intervention.
One of the major ways in which the immersive experience can be increased is by providing the user a larger field of view and thereby giving the user the feeling of being surrounded by a virtual world. This requires stitching of videos in spatial domain. Lot of research has been done in terms of creating virtual environments such as the work done in paper [1], but not much research has gone into creating high resolution stitched videos in real time.
Most of the work till now has only been able to create solutions that work on low resolution videos with very low playback rate in terms of frames per second. It is also noted that very little research has gone into real-time online video stitching. One of the earlier works in panorama generation for

tele-immersive applications [2] offered various techniques for a full implementation, whereas fundamental improvements in video stitching algorithm were not explored.
A combination of efficient algorithm for image stitching, pipelined execution and use of GPUs is explored in this paper to solve these challenges.

A. Challenges in video stitching

The problem of video stitching is dealt with in various ways. One of the approaches is to decompose the problem into an iterative image stitching sub-problem. Traditionally image stitching algorithms [3], repeats the steps of feature detection, feature matching, transform model estimation and image re-sampling and transformation. In video stitching, these steps are repeated on all sets of frames that can be stitched. This approach is not able to produce stitched videos in real-time, since the associated algorithms are computationally expensive. A decrease in stitching time can be achieved by reducing the quality of the video, and this has remained to be a trade-off. Some of the latest works[4] has only achieved frame rates of up to 12 fps using low resolution videos.
In a scenario where the cameras remain static, many of the steps involved in image stitching were found out to be redundant. Our work deals with panoramic stitching based on decomposing the video into its basic units, i.e, frames and then using our novel image stitching algorithm to stitch image-by-image and re-packaging the stitched frames to produce a standard definition panoramic video, in real-time.

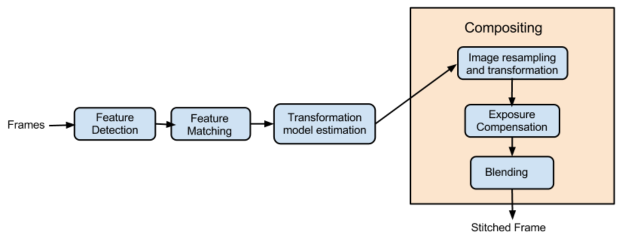


Fig 1. The sequential steps in the image stitching process

This paper presents a novel algorithm, which dynamically detects the redundancies and reuses the calculated transformation matrices to stitch the subsequent frames. The stitching process is further improved by parallelizing the execution using GPUs and using a pipelined approach in the stitching algorithm. Experimental results have shown that this algorithm improves stitching over the existing methods. Hence, this algorithm can be used for real-time video stitching applications in standard definition.
The rest of the paper is arranged in the following manner. Section 2 of this paper deals with the related work in the area of panoramic video stitching. In section 3, we introduce our work on video stitching. The methods to speed up video stitching through iterative compositing, pipelining and GPU acceleration are discussed in section 4. The testing premise and the evaluation metrics are described in section 5. The results and observations are presented in section 6 and in section 7 the conclusion and the future work.

Related Work

Studies have been conducted on stitching multiple standard definition videos to achieve panoramically stitched video, which is stable and at the same time devoid of visual artifacts. Paper [4] proposes a panoramic video construction algorithm that exploits temporal information to stitch videos on a frame-by-frame basis. It uses information from two independently and freely moving capturing devices (mobile camera), stitches them and outputs the video. This non-real time approach does not deal with occlusion problems, and efficiency is compromised in terms of standard definition videos. The low recall-precision fraction [4] indicates that this technique needs large improvement for it to become a stable one.
An offline video stitching method based on global motion tracking in compressed domain is explored in Shimizu et al.'s [5] work. They achieved real time stitching in compressed domain using fast MV based GMV calculation and SIMD based block matching. The work in [5] describes fast techniques for stitching videos, but the authors only showed objective alignment results for pure translation transformation videos, which is not a realistic situation.
Papers [6][7] discuss a system for stitching videos streamed from mobile phones. They consider video stitching problem as an image-stitching problem. In the work done by Motaz et al. [6], the system receives video streams coming from two mobile phones, time synchronizes the streams and produces a single composite mosaic video in real time. These works deal with the moving camera problem, and the corresponding solutions, though the results show that these techniques cannot be used for a stable algorithm for panorama creation.
Solving the image-stitching problem is extensively studied in the literature [8][9]. Zitova et al. [3] has done extensive research in image registration techniques, especially categorizing the various approaches in image stitching. An algorithm that performs registration, and an optimized method to make the projection transform is discussed by Linqiang et al.[16]. Their work which uses less-exhaustive blending method to blend images, with feeds from two VGA cameras, is computationally intensive as well as non-scalable.
Paper [15] describes GPU-based implementation of image stitching in real-time, which can stitch high definition images at constant frame rates. Michael et al.'s [2] work uses stitch map and belief propagation to achieve real-time stitching, though this is not an online solution.
Paper [10] proposes a video stitching system for streams coming from webcams. Their system requires an initialization phase, which is not real-time and is needed whenever the webcams positions change, apart from the fact that low resolution makes it an unviable option for practical applications in the present high bandwidth networks.

Speeding Up Video Stitching

Our work focuses on solving video stitching problem by considering it as an image stitching sub-problem. In our scenario, the videos are captured in standard definition using static cameras. There is no constraint on the positioning of the cameras, except that the different frames should have enough matching feature points between them. The captured videos are decomposed into frames at a standard frame rate of 25 fps and fed into the stitching algorithm, which stitches the frames into a single panoramic image. The resultant panoramic images are buffered and displayed at the same frame rate, thereby creating a real-time panoramic video. The following section describes the image stitching algorithm and the optimizations introduced in this paper.

B. Iterative Compositing in Image Stitching

Image registration, according to Zitova et al. [3], is a four step process: feature detection, feature matching, transformation model estimation and finally, image re-sampling and transformation. The fourth step includes compositing, which consists of warping and blending. We have used the above classification while describing our work in the later parts.
Feature detection is the process to find out the prominent features, viz. a viz., corners, lines, points and gradient areas in the image, which can be described using unique feature descriptors. In our algorithm, features from different frames are detected using SURF [11] algorithm, which is robust and computationally faster compared to SIFT [12]. After feature detection, the feature points (FPs) in one frame are matched with the FPs in the other. A pair-wise matching is performed to find out if the two frames can be stitched, i.e., whether they are part of a single panorama. In case of more than two frames, the algorithm uses the FP matching to find out the order of frames in the panorama.

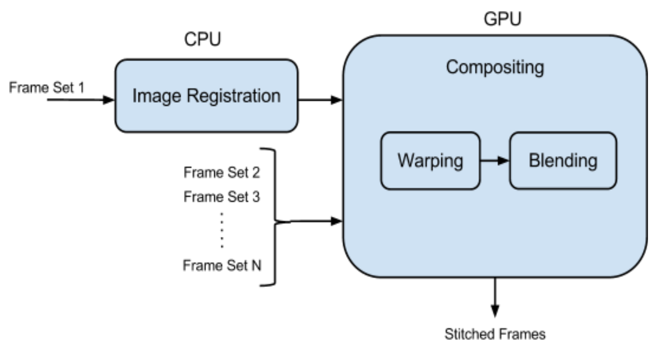


Fig. 2. Depicts the idea of iterative compositing. Image registration is done on the first frame set and iterative compositing done on frame set 2, 3… n.

The properties of the camera, such as the focal length, aspect ratio, rotation and translation are found out in transformation model estimation. These values are refined to include wave correction factor and panorama scale estimation. The warping and blending matrices are calculated from these values. The steps of feature detection, feature matching and camera parameter calculation are performed only on the first set of frames that can be stitched. These three steps are redundant in case of static cameras, since subsequent frames show temporal redundancy, i.e, the overlap region and the relative positions of the scenes remain constant. Therefore, in subsequent frames, the algorithm reuses the warping and blending matrices for stitching. We call this technique iterative compositing.
A plane warper [13] is used for warping the set of frames, since it is computationally faster than other warping techniques such as cylindrical or spherical warping. . A projection map is created from camera parameters, which is used to remap the pixels from the source image to the warped frame. The warped images are corrected for exposure differences. After warping the individual images, they are blended using multi-band blending [14]. Our analysis showed that multi-band blending creates visually superior results when compared to feather blending. The blending technique is further enhanced by dynamically detecting the blending width, which is directly proportional to the overlap region between the images in a panorama. The blending algorithm creates a laplacian pyramid with number of levels equal to the number of bands, and also creates a weight map gaussian pyramid. Blending involves using these weight maps to normalize the frames in the panorama. The end result of blending is a seamless panorama of individual frames.The warped images are blended using multi-band blending [14]. When compared to feather blending, this technique produces smoother resultant images with no visible seam.
The stitched frames are buffered and sequentially played back to produce the panoramic video. This algorithm is optimized to outperform the existing techniques to produce panoramic video in terms of reduced computational complexity and increased visual quality.
The algorithm is modified to run on a GPU. Both warping and blending modules were run on the GPU. The plane warper speeds up by a factor of 12 when run on a GPU, while the multi-band blender is only marginally improved. GPU is not used for feature detection, feature matching and camera parameter calculations, since the GPU overhead associated with the single time calculation of these matrices is greater when compared to using CPU to perform the same operations.

C. Pipelining Compositing Modules

The stitching process discussed in the above section, performs the operations sequentially. Though iterative compositing is faster than existing methods, it is still not real-time. To further speed up this algorithm, multiple instances of iterative compositing modules are put in a pipeline. A synchronization module is implemented to ensure that each of these instances are well co-ordinated and stitch the frames in the correct sequence.
The pipeline implementation is found to increase the stitching speed by a factor of 2, and thereby achieving real-time stitching. Suppose t_{single} ms is the time taken to stitch one set of frames using a single instance of compositing module and let fps represent the frames per second of the received camera feed. The optimal number of instances (N_{opt}) of compositing modules that can be included in the pipeline is given by:

$$N_{opt} = (t_{single} * fps) / 1000 \qquad (1)$$

In our experiments, we found that N_{opt} instances of compositing module can produce stitched videos in real-time.

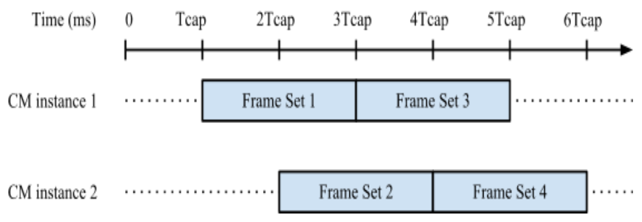
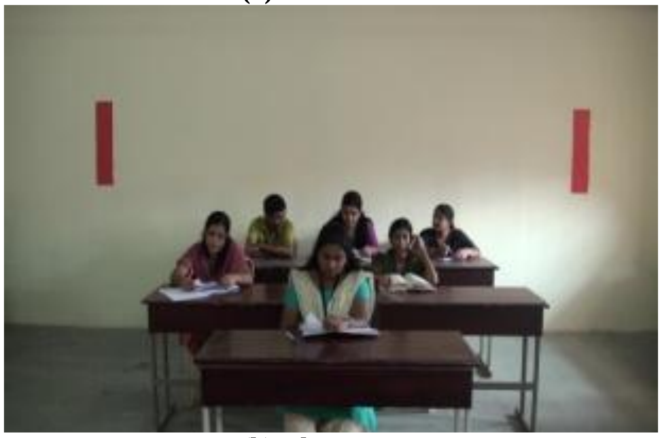


Fig. 3. The pipeline implementation. Tcap is the time taken for capturing a set of frames. The diagram shows how two instances of compositing modules (CM) works in the pipeline.

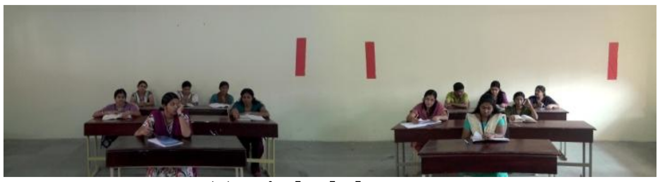
Testing and Evaluation

In a previous work[4] on video stitching based on freely moving devices, the lack of standard test set for generating panoramic video was mentioned, and the current scenario of fixed-camera based stitching also presents a similar problem. For testing this algorithm, different sets of videos were captured with different lighting conditions, indoor and outdoor setups as well as different types of cameras, both standard and high definition. Fig. 4. shows a sample set of frames captured from indoor classroom setup, and the resultant stitched image.

(a) Classroom 1



(b) Classroom 2



(c) Stitched classroom

Fig. 4.Top (a and b) are classrooms captured through two 720p cameras. (c) is the snapshot of the stitched video with both the classrooms in view.

A 3.2GHz 4-core processor with 8GB memory along with a 144-core 3GB dedicated GPU was used to run the algorithm. In the results presented here, two cameras with a resolution of 720p were used to capture the videos. We also ran the algorithm on recorded videos of durations of 10s, 20s, 1 min, 2 min and 10 min. Apart from this, we also performed stitching of videos from live feed from these two cameras. The results are presented below.

Results and Observations

Initial experiments using off-the-shelf solutions such as batch processing using Hugin [17] and Panorama Factory [18] were able to produce stitched frames at the rate of less than 2 frames per second. Other solutions like AutoStitch [19] and StitcHD [20], which does online stitching were highly unstable and not a viable option.

The analysis showed that over 80% of the computational time is spent on the first three steps of image registration, i.e., on feature detection, feature matching and transform model estimation. Hence we employ the technique of iterative compositing to speed up the stitching process.

In the algorithm presented here, the quality of the resultant stitched image and the computational complexity of the stitching process are directly related to the warping and blending techniques used. Hence, a comparative analysis of the various warping and blending algorithms on both CPU and GPU is presented in section 6.1. The results are based on individual videos streamed at 720p, which are stitched into a single panorama.

D. ***Blending and Warping Analysis***

The iterative compositing, which consists of warping and blending frame by frame is the central process presented in this paper. The results show that a combination of feather blender and plane warper when run on a GPU takes 23% lesser time compared to multi-band blender and plane warper implementation. But the feather blending technique was found to be not as effective in removing visible seams, and hence the paper explores the implementation of the later technique. For the later sections, the paper concentrates only on the plane warper and multi-band blender combination. Table 1 and 2 shows the analysis of various warper-blender combinations on CPU and GPU respectively.

TABLE I. **Time take in ms for stitching one set of frames for different combinations of warper-blender combination on a CPU.**

Property	**Plane warper (ms)**	**Cylindrical warper (ms)**	**Spherical warper (ms)**
Multi-band blender	185	266	369
Feather blender	158	244	334

TABLE II. **Time take in ms for stitching one set of frames for different combinations of warper-blender combination on a GPU.**

Property	**Plane warper (ms)**	**Cylindrical warper (ms)**	**Spherical warper (ms)**
Multi-band blender	97	100	100
Feather blender	75	76	77

E. Iterative Compositing Analysis

When all the steps of image registration, i.e., feature detection, matching, transform model estimation and compositing are done on all the frames, the resultant stitched video had a frame rate of less than 2 fps (frames per second). The method of iterative compositing improved this frame rate to 11fps.

An earlier work [4] has achieved a similar frame rate, but their results were based on low-resolution images apart from the low recall-precision factor. Iterative compositing cannot be measured against the metric of recall-precision since all the frames, starting from the first stitchable frame, are stitched. To measure the quality of the stitched video, the jitter was also measured, as a standard deviation in compositing time. Since the playback of stitched frames is a faster process when compared to stitching, the jitter value of compositing represents a metric to measure the video quality. When the compositing module was run on a GPU, the jitter value was found to be 2 ms, which is less than 2% of the total time required to playback at 11 fps. This result has a direct relationship to the visual quality of the playback and hence proves the stability of the algorithm in generating stitched frames. Apart from the quality measure using jitter, the panoramic videos were also analysed by users for visual quality. The results show that there were no visible seams or other introduced artifacts in the resultant video.

A single instance of compositing module was not able to produce real-time panoramic video. Analysis showed that the GPU utilization was less than 10%, and hence more instances could be run in parallel. Through the use of pipelined iterative compositing, the stitching process improved from the earlier value of 11 fps to a real-time rate of 25 fps.

Conclusion and Future Work

This paper described the various optimizations done in the image stitching process to achieve real time stable video stitching in the scenario of static cameras at a resolution of 720p. The removal of redundant image registration steps resulted in an improvement in the computation time by a factor of 2.5. Further improvement was achieved by using the processing capabilities of GPU to parallelize and perform the complex matrix calculations involved in the stitching process. Together these optimizations resulted in an improvement by a factor of 4.5 giving an effective playback frame rate of 11 fps. Our results also shows that real time video stitching can be achieved by pipelining the compositing module. To our knowledge this is the first attempt at using iterative compositing technique to accomplish real-time online video stitching at standard definition and promises to be an area that needs more research. Future work is expected to bring end-to-end algorithms and implementation which will enhance the field of view based immersive experience of users.

Acknowledgment

The authors would like to express gratitude for the immeasurable motivation and guidance provided by Sri. Mata Amritanandamayi Devi (AMMA), Chancellor of Amrita University. This work was support by the NKN project, funded by DIT, Govt. of India. The authors would also like to acknowledge the contributions of Dr. Maneesha V Ramesh and many other colleagues and friends as well.

References

[1] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, “Surround-screen projection-based virtual reality: the design and implementation of the CAVE“, SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques, Pages 135 – 142.

[2] Wai-Kwan Tang, Tien-Tsin Wong, Pheng-Ann Heng, “A System for Real-time Panorama Generation and Display in Tele-immersive Applications”, IEEE Transactions on multimedia, Vol. 7, No. 2, April 2005, Pages 280 – 292.

[3] Barbara Zitova and Jan Flusser, “Image registration methods: a survey”, Image and Vision Computing, Vol 21 Issue 11, October 2003, Pages 977–1000

[4] Motaz El-Saban, MostafaIzz and AymanKaheel, “Fast Stitching of Videos Captured from Freely Moving Devices by Exploring Temporal Redundancy”, International Conference on Image Processing, September 2010

[5] T. Shimizu, A. Yoneyama and Y. Takishima, “A fast video stitching method for motion-compensated frames in compressed video streams”, International Conference on Consumer Electronics, 2006.

[6] Motaz El-Saban, Mahmoud Refaat, AymanKaheel and Ahmed Abdul-Hamid, “Stitching Videos Streamed by Mobile Phones in Real-time”, MM’09, October 19–24, 2009, Beijing, China.

[7] A. Kaheel, M. El-Saban, M. Refaat, and M. Izz, “Mobicast - A system for collaborative event casting using mobile phones”, in ACM MUM '09.

[8] R. Szeliski, “Image Alignment and Stitching: A Tutorial”, MSR Tech Report

[9] M. Bujnak and R. Sara, “A Robust Graph-Based Method for The General correspondence Problem Demonstrated on Image Stitching”, ICCV 2007.

[10] M. Zheng, X. Chen and L. Guo, “Stitching Video from Webcams”, Lecture Notes In Computer Science; Vol. 5359 archive Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II, 2008

[11] Herbert Bay, TinneTuytelaars, and Luc Van Gool, “SURF: Speeded Up Robust Features”, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I, Pages 404 - 417

[12] Luo Juan, and OubongGwun,“A Comparison of SIFT, PCA-SIFT and SURF”, International Journal of Image Processing (IJIP) Volume (3), Issue(4)

[13] C A Glasbey, and K V Mardia, “A review of Image Warping Methods”, Journal of Applied Statistics - 1998, Vol 25, Pg 155 – 171

[14] Janos Vida, Ralph R. Martin, TamasVarady, “A Survey of Blending Methods that use Parametric Surfaces”, Computer - Aided Design, Vol 26, Issue 5, May 1994, Pages 341 – 365

[15] Michael Adam, Christoph Jung, Stefan Roth, Guido Brunnett, “Real-time Stereo-Image Stitching using GPU-based Belief Propagation”,Vision, Modeling, and Visualization Workshop 2009 in Braunschweig, Germany, November 2009

[16] Linqiang Chen, Xiaoqiang Wang, and Xu Liang, “An Effective Video Stitching Method”, International Conference On Computer Design And Applications (ICCDA 2010)

[17] Web presentation of the Hugin – Panorama photo stitcher, http://hugin.sourceforge.net/

[18] Web presentation of Panorama Factory - Panorama photo stitcher - http://www.panoramafactory.com/

[19] Web presentation of AutoStitch automatic panorama maker software

[20] Web presentation of StitcHD source code – https://github.com/lukeyeager/StitcHD