Parallax-Tolerant 360 Live Video Stitcher

Miko Atokari, Marko Viitanen, Alexandre Mercat, Emil Kattainen, Jarno Vanne
Computing Sciences, Tampere University, Tampere, Finland
{miko.atokari, marko.viitanen, alexandre.mercat, emil.kattainen, jarno.vanne}@tuni.fi

***Abstract*—This paper presents an open-source software implementation for real-time 360-degree video stitching. To ensure a seamless stitching result, cylindrical and content-preserving warping are implemented to dynamically correct image alignment and parallax, which may drift due to scene changes, moving objects, or camera movement. Depth variation, color changes, and lighting differences between adjacent frames are also smoothed out to improve visual quality of the panoramic video. The system is benchmarked with six 1080p videos, which are stitched into 4096×732 pixel output format. The proposed algorithm attains an output rate of 18 frames per second on GeForce GTX 1070 GPU and real-time speed can be met with a high-end GPU.**

Keywords—stitching, 360 video, parallax-tolerant, content-preserving warping (CPW), multiband blending

I. INTRODUCTION

Video stitching is a key process in combining multiple videos seamlessly together in order to create a panoramic video with a higher *field of view* (*FoV*) and resolution. Recently, video stitching has become an integral processing tool in many fields such as in surveillance [1] and entertainment. It has particularly gained traction with the emergence of immersive 360-degree videos in *virtual and augmented reality* (*VR/AR*) applications [2] whose global IP traffic is estimated to grow 65% annually [3]. However, existing open and free solutions, such as Facebook 360 [4] and Radeon Loom [5], call for expensive equipment, thus putting live video stitching out of reach for wider public.

Parallax, color tune, and exposure errors are common visual artifacts in panoramic photography [6]. Parallax misalignment [7] is caused by a shift in apparent object position when observed from different angles or view points. Combining images together also leads to color tone and exposure differences between them [8]. Video stitching has to also take care of temporal quality and react to the changing scenery, moving objects, and camera movement.

In the previous works, stitching artifacts have been addressed by a sequence of independently optimized steps. There are two common approaches for aligning and warping images and videos: 1) feature point detection and 2) direct comparison of image pixels. In the rest of the paper, we focus on the first approach. Global alignment based stitching methods [1] can be used to adjust images which have been taken from different angles but they become inefficient when the distance between cameras increases. Both global alignment and parallax error were minimized in [7], [9]. In addition, several feature matching techniques [2], [8], [10] have been proposed for image stitching whereas the problem of depth variation is tackled in [11], [12], [13], [14] with dynamic video content.

In this paper, we propose an algorithm for stitching a 360-degree live panoramic video from multiple input video feeds. The input can be either raw or compressed video. Our proposal uses cylindrical warping as a rough warping technique to ensure a continuous structure of the adjacent frames. *Content-preserving warping* (*CPW*) is then applied on top of cylindrical warping to improve warping quality through feature point matching. To compensate changing scenery, moving objects, and camera movement, recalibration is dynamically performed to determine the matching features and apply the CPW in parallel with stitching. The exposure compensation is also implemented to reduce lighting differences. Multiband blending is finally used to smooth out color differences between frames and soften frame edges on the stitching seams.

The proposed algorithm is based on the *Open source computer vision* (*OpenCV*) library [15] and its computation is partitioned between the GPU and CPU. To the best of our knowledge, our solution is the first open-source parallax-tolerant algorithm for live 360-degree video stitching.

This paper is structured as follows. Section II presents the state-of-the-art algorithms for video stitching. The proposed live video stitching algorithm is described at high level in Section III and its three main stages in Sections IV-VI. Performance analysis and comparison with prior-art is provided in Section VII. Section VIII concludes the paper.

II. RELATED WORK

In the last five years, several approaches have been introduced for live video stitching. S. H. Yeh et al. [10] proposed a GPU-accelerated solution that used temporally interpolated homography and cylindrical warping. They sped up feature point detection with masks. K. C. Huang et al. [8] improved cylindrical warping by introducing feature pairs for more accurate image alignment. Dynamic seams were also used to make them less obtrusive.

P. P. Shete et al. [1] separated stitching process into offline and online stages. The offline stage utilized OpenCV library [15] to calculate inverse look up maps and feather weight masks. The maps were needed on the online stage for warping the image with OpenGL. F. Xu et al. [2] created a live 360-degree 3D stitching technique for VR devices.

Scene depth is frequently varying for dynamic video content. C. Herrmann et al. [11] addressed the depth variation in images by creating multiple transformations to different parts of the image. Similarly, J. Gao et al. [13] separated input images into two planes for better alignment. W. Zeng et al. [14] tackled temporal depth variation in videos with an algorithm that calculates changing projection transforms with a depth compensation method. K. Chen et al. [12] mapped movement of feature points into a mesh to stitch videos smoothly both temporally and spatially.

F. Zhang et al. [7] introduced the CPW algorithm to minimize stitching artifacts caused by parallax and global alignment. It addressed both image content and geometric alignment to find an optimal local region for stitching. W. Jiang et al. [9] expanded upon the idea of CPW for video stitching by also taking into account temporal information.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

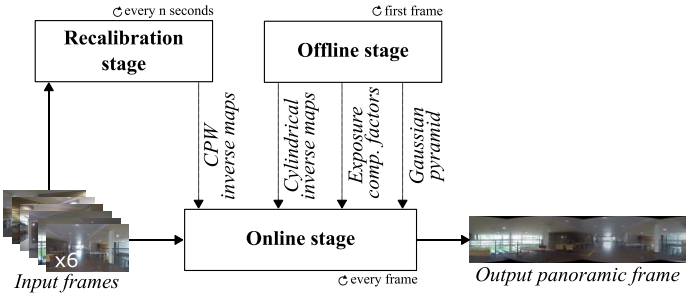

Fig. 1. Overview of the stitching algorithm.

III. Proposed Solution

The objective of the proposed work is to develop an algorithm that is capable of stitching 360-degree panoramic video from multiple input videos in real time. Seamless stitching result is sought by tackling all common stitching artifacts, i.e., the parallax effect, changing scenery, exposure differences, and frame edge blending. To create the conditions for live stitching, the camera setups are subject to the following constraints, i.e., the cameras are required to

1) Have coplanar projection centers.
2) Have a known and equal amount of rotation around the vertical axis between them.
3) Have uniform circular distribution ensuring sufficient overlap between adjacent cameras.
4) Have a linear FoV due to the absence of distortion correction.

In this work, our algorithm is mapped to a system with six video cameras, each having 15 degrees of overlap on both vertical edges of the frame.

The proposed stitching process is divided into 1) offline; 2) recalibration; and 3) online stages, as illustrated in Fig.1. The offline stage is performed only once at the startup. It generates the cylindrical inverse maps, the exposure compensation factors, and the Gaussian pyramid for the online stage. The recalibration and online stages are executed in parallel for all frames. The purpose of the recalibration stage is to periodically compute the CPW inverse maps for the online stage. The online stage is the most time-critical part of the algorithm. It uses the inputs from the offline and recalibration stages to warp, compensate exposure, and finally blend the input frames into 360-degree panoramic frame. The Sections IV-VI describe these three stages in more detail.

IV. Offline Stage

The offline stage is executed once at the start of the stitching process. It is composed of the following three operations which are all implemented on a GPU.

A. Exposure Compensation

In multi-camera setups, adjacent cameras may have different exposure times depending on the scene. This tends to cause differences in brightness levels between individual video feeds. To compensate this phenomenon, exposure compensation factors are determined for the first frame of each input video feed. These factors are individually calculated per frame by minimizing pixel-wise difference in the overlapping regions of adjacent frames. A prior term is also used to keep the factor values close to one. The factors are taken in use on the online stage (cf. Section VI).

B. Cylindrical Inverse Map Generation

The cylindrical warping is based on a cylindrical projection, where the pixels of the source frames are projected on virtual cylindrical coordinates with the help of cylindrical inverse maps. The purpose of these maps is to make stitching seams look visually more continuous. In practice, the cylindrical inverse maps are 2D arrays, which map locations in the final frame to a point in a source frame. They are generated for each source frame as

$$x = f \times \left(\tan\left(\frac{x'}{s}\right) + \tan(\theta)\right) \text{ and } y = f \times \left(\frac{y'}{s}\right) \times \sec\left(\frac{x'}{s}\right),$$

where f is the focal length of the camera, (x', y') are the coordinates in a source frame, θ is the horizontal rotation of the camera, and s is a scaling factor [16].

C. Gaussian Pyramid Generation

The Gaussian pyramid is generated for the multiband blending performed on the online stage (cf. Section VI). The blending is calibrated by first creating a Voronoi diagram [17] of all input frames so that a point set for the diagram is composed of frame centers. Then, a Gaussian pyramid of depth 6 is created from the Voronoi diagram.

V. Recalibration Stage

The cylindrical inverse map provides a rough estimate of frame alignment. As described in Fig. 2, CPW [7] is applied on the recalibration stage to refine the rough alignment and remove the parallax caused by the depth of the scene and position of the cameras.

The CPW process divides every input frame into a mesh of $M \times N$; 10×10 mesh is applied in this work. The mesh is recalibrated in two steps: 1) the matching features determination (in blue in Fig. 2) and 2) the CPW mesh recalibration (in red in Fig. 2). These two steps are executed every n seconds in parallel with the online stage.

A. Matching Feature Determination

This step aims to find relevant feature points on the input frame edges. The feature points are then used to recalibrate the corresponding meshes. Our solution uses the *Oriented FAST and rotated BRIEF* (*ORB*) feature detector [15] of OpenCV. For each input frame, ORB searches for features and creates a mask that only includes the overlapping regions of the frames in order to limit the computational complexity and reduce feature mismatches. The features are then matched to those of the adjacent frames using the BruteForce-Hamming descriptor matcher [15]. The outliers are finally filtered out with *Random sample consensus* (*RANSAC*) [15].

A small amount of overlap and uniform scenery reduce the number of matching features and increase the portion of mismatched features, even after filtering out outliers with RANSAC. Since the cameras are fixed on a horizontal plane, they can only rotate on a vertical axis. This characteristic is used to further filter out matching features that have y-distance or x-distance greater than the corresponding thresholds. With our camera rig, the y-distance and x-distance thresholds were set to 40 and 300 pixels, respectively.

Mesh recalibration causes unwanted temporal movement on the seam, which can be minimized by

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

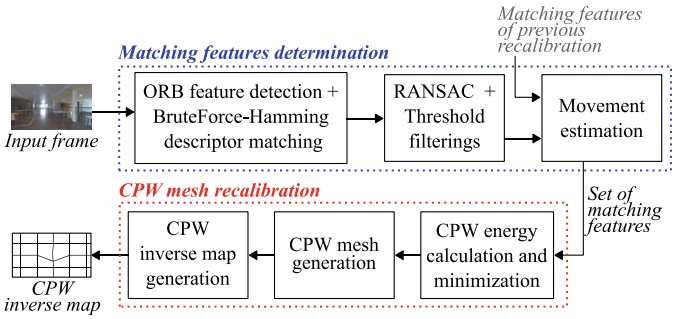

Fig. 2. Recalibration stage.

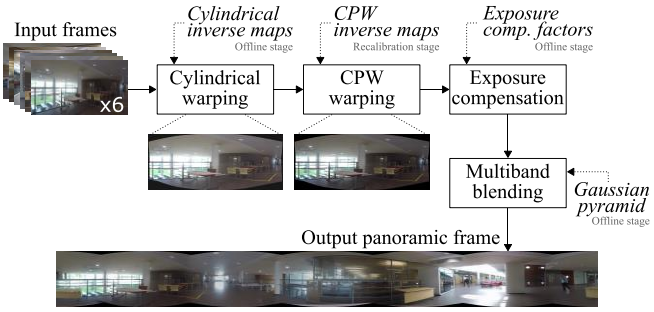

Fig. 3. Online stage.

recalibrating only seams featuring backward or forward movement. The movement is roughly derived from the average spatial distance of the matched feature points across two adjacent frames. This distance is compared with that of the last recalibration. If the difference between them is higher than a fixed threshold of 15 pixels, the matching features of the previous frame are used for the recalibration, as illustrated in Fig. 2.

B. CPW Mesh Recalibration

This step implements CPW mesh recalibration using the matching features determined in the previous step. If no matching features have been found in the overlapping regions after filtering, CPW mesh recalibration is omitted. In that case, only cylindrical warping is applied on the seam of interest on the online stage.

CPW mesh recalibration starts by computing three energy terms on each vertex of the uniform mesh: local, global, and smoothness energies. The local term is computed based on feature points and it is used to align the adjacent mesh vertices of the frame. The global term constrains the mesh vertices without matching feature points to keep their positions. The smoothness term minimizes the local distortion, making the warping less noticeable. These three terms are combined and then minimized using least squares conjugate gradient solver in Eigen library [18]. CPW is described in more detail in [7].

The generated mesh is then converted to inverse map format based on an intermediate downscaled forward map. The forward map is calculated by a bilinear interpolation of the values between the mesh vertices. The downscaled forward map is then inverted by mapping pixels from forward map to inverse map. Values which correspond to the same pixel in the inverse map are averaged. The inverse map is finally upscaled to full frame size using bilinear interpolation.

To smooth the transition between calibrations of successive frames, a linear interpolation is applied between the consecutive meshes. The idle time between two recalibrations is used to compute interpolated meshes for intermediate frames.

The ORB feature detection and CPW inverse map generation are executed on a GPU whereas the rest of this stage is implemented on a CPU.

VI. Online Stage

Fig. 3 illustrates the online stage, which is repeated for every group of frames. First, the online stage warps the input frames from the cameras with the help of the corresponding cylindrical inverse map received from the offline stage. Secondly, it applies the latest CPW inverse map got from the recalibration stage. Thirdly, the exposure compensation factors are used to smooth the lighting differences between the frames. Finally, a multiband blender, optimized for video stitching, blends the six frames together into 360-degree panoramic frame. The multiband blending algorithm makes Laplacian pyramid out of each warped frame. These pyramids are masked with the corresponding Gaussian pyramids generated on the offline stage and summed together. The sum of pyramids is restored to a single frame resulting in the output panoramic frame.

The high complexity of this stage is addressed by fully porting it to a GPU, i.e., CUDA-accelerated OpenCV implementations are applied for warping, exposure compensation, and multiband blending. Furthermore, the OpenCV implementation for multiband blending is optimized with pre-calculations.

VII. Performance Analysis

The input video feeds for our experiments were taken with a rig of six GoPro Hero 6 action cameras. Each of them output H.264/AVC video at 30 fps and 1080p resolution. The stitching was run on a desktop computer equipped with GeForce GTX 1070 GPU and Intel Xeon W-2145 CPU (32 GB of RAM) running the Ubuntu 18.04 operating system. The videos were decoded in parallel with the stitching.

A. Performance Evaluation

According to our results, the execution of the offline stage takes around 880 ms. However, this stage is needed only once at the start of the stitching process, so its overhead is negligible with respect to the overall algorithm.

The recalibration stage is executed in around 1000 ms, which includes the determination of matching features and the CPW mesh recalibration. Finding the best trade-off between quality improvement and computational overhead requires that the execution interval of this stage is tied to scenery changes and camera movement. In our experiments, a recalibration interval was set to 5 s.

The online stage takes 47 ms when the CPW mesh recalibration is off and 55 ms when enabled. This stage is executed for every frame, making it the most critical stage in terms of the algorithm speed. With the recalibration interval of 5 s, our system is capable of producing live 360-degree panoramic 4096×732 video at 18 fps.

B. Quality Assessment

Fig. 4 shows the quality comparison of the proposed algorithm features on two consecutive frames: *robot* (with a close object) and *corridor* (without any close objects). *Corridor* is the frame just after the robot is removed.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

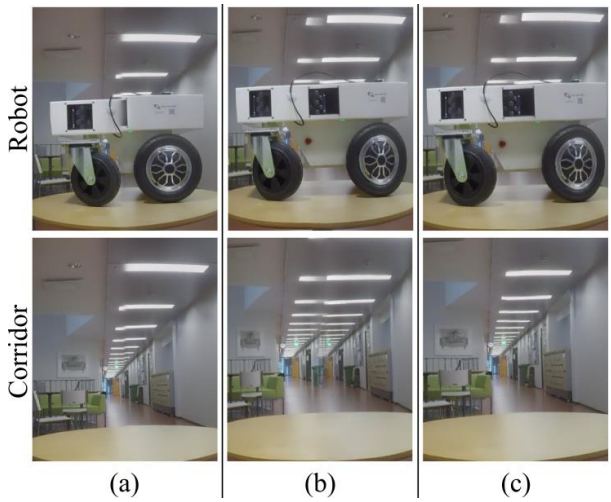

Fig 4. Quality comparisons of the stitching algorithm features. (a) CPW mesh recalibration disabled. (b) CPW mesh recalibration enabled without recalibration between frames. (c) CPW mesh recalibration enabled with recalibration between frames.

Fig. 4(a) illustrates the stitching results when only cylindrical warping is enabled. This setup produces decent quality with the *corridor* frame, but misalignment from parallax is visible when the object is close to the cameras, as shown on the *robot* frame. The CPW mesh recalibration is enabled in Fig. 4(b). It fixes the previous misalignment on the *robot* frame but the error becomes visible on the *corridor* frame after the robot is removed. The recalibration interval is adjusted in Fig. 4(c) to take into account the scenery changes. This adjustment removes misalignment from both frames and validates depth-adaptive and parallax-tolerant properties of our algorithm.

C. Comparison with the State-of-the-Art

Table I presents a feature comparison between our proposal and prior stitching algorithms in the literature. The second half of the related works is limited to offline video stitching [9] or still image stitching [11], [13], [7]. The first half [1], [8], [10], [14] is capable of live video stitching, but only two of them [10], [14] support recalibration and none of them is parallax-tolerant. Moreover, only one of these prior works supports 360-degree videos.

To the best of our knowledge, our solution is the first open-source parallax-tolerant algorithm implementation for live 360-degree video stitching. It is available at *https://github.com/ultravideo/video-stitcher*.

VIII. Conclusion

This paper presented an open-source stitching algorithm to produce panoramic videos in real time. Our proposal uses feature point matching to ensure a continuous structure of the adjacent frames. Recalibration is dynamically performed to compensate depth variation induced by scenery changes, moving objects, and camera movement. The color differences and frame edges on the stitching seams are smoothed out to improve quality of the stitched video. Our proposal is able to stitch six 1080p videos to 4096×732 format at 18 fps on a consumer-grade GPU and it is foreseen to attain real-time speed on a high-end GPU.

Acknowledgment

This work was supported in part by the European ECSEL project PRYSTINE (under the grant agreement 783190) and the Academy of Finland (decision no. 301820).

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

TABLE I. Feature Comparison with the State-of-the-Art

		Performance	Recalibration	Parallax-tolerant	360-degree	Open source
Proposed Solution		live	yes	yes	yes	yes
P. P. Shete et al.	[1]	live	no	no	no	no
K. C. Huang et al.	[8]	live	no	no	yes	no
S. H. Yeh et al.	[10]	live	yes	no	no	no
W. Zeng et al.	[14]	live	yes	no	no	no
W. Jiang et al.	[9]	video	yes	yes	no	n/a
C. Herrmann et al.	[11]	image	n/a	yes	no	n/a
J. Gao et al.	[13]	image	n/a	yes	no	n/a
F. Zhang et al.	[7]	image	n/a	yes	no	n/a

References

[1] P. P. Shete, D. M. Sarode, and S. K. Bose, "Real-time panorama composition for video surveillance using GPU," *in Proc. Int. Conf. Advances in Computing, Communications and Informatics*, Jaipur, India, Sept. 2016.

[2] F. Xu, T. Zhao, B. Luo, and Q. Dai, "Generating VR live videos with tripod panoramic rig," *in Proc. IEEE Conf. Virtual Reality and 3D User Interfaces*, Reutlingen, Germany, Mar. 2018.

[3] Cisco, Cisco Visual Networking Index: Forecast and Trends, 2017-2022, Feb. 2019.

[4] Surround360 System [online]. Available: https://github.com/facebook/Surround360.

[5] Radeon Loom Stitching Library [online]. Available: https://gpuopen.com/compute-product/radeon-loom/.

[6] D. B. Goldman, "Vignette and exposure calibration and compensation," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2276-2288, Dec. 2010.

[7] F. Zhang and F. Liu, "Parallax-tolerant image stitching," *in Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Columbus, Ohio, USA, June 2014.

[8] K. C. Huang, P. Y. Chien, C. A. Chien, H. C. Chang, and J. I. Guo, "A 360-degree panoramic video system design," *in Proc. IEEE Int. Symp. VLSI design, Automation and Test*, Hsinchu, Taiwan, Apr. 2014.

[9] W. Jiang and J. Gu, "Video stitching with spatial-temporal content-preserving warping," *in Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, Boston, Massachusetts, USA, June 2015.

[10] S. H. Yeh and S. H. Lai, "Real-time video stitching," *in Proc. IEEE Int. Conf. Image Processing*, Beijing, China, Sept. 2017.

[11] C. Herrmann, C. Wang, R. S. Bowen, E. Keyder, M. Krainin, C. Liu, and R. Zabih, "Robust image stitching with multiple registrations," *in Proc. European Conf. Computer Vision*, Munich, Germany, Sept. 2018.

[12] K. Chen, J. Yao, B. Xiang, and J. Tu, "Video stitching with Extended-MeshFlow," *in Proc. Int. Conf. Pattern Recognition*, Beijing, China, Aug. 2018.

[13] J. Gao, S. J. Kim, and M. S. Brown, "Constructing image panoramas using dual-homography warping," *in Proc. IEEE Computer Vision and Pattern Recognition*, Colorado Springs, Colorado, USA, June 2011.

[14] W. Zeng and H. Zhang, "Depth adaptive video stitching," *in Proc. IEEE Int. Conf. Computers and Information Science*, Shanghai, China, Aug. 2009.

[15] OpenCV library [online]. Available: http://code.opencv.org.

[16] R. Szeliski, "Image Alignment and Stitching: A Tutorial," *Foundations and Trends in Computer Graphics and Vision*, Vol. 2: No. 1, pp 1-104, Jan. 2006.

[17] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345-405, 1991.

[18] Eigen linear algebra library [online]. Available: http://eigen.tuxfamily.org.