Pacific Graphics 2017
J. Barbic, W.-C. Lin, and O. Sorkine-Hornung
(Guest Editors)

Volume 36 (2017), Number 7

High-resolution 360 Video Foveated Stitching for Real-time VR

*Wei-Tse Lee[1] *Hsin-I Chen[1,2] Ming-Shiuan Chen[1] I-Chao Shen[1] Bing-Yu Chen[1]

[1]National Taiwan University [2]Tesla, Inc
* equal contribution

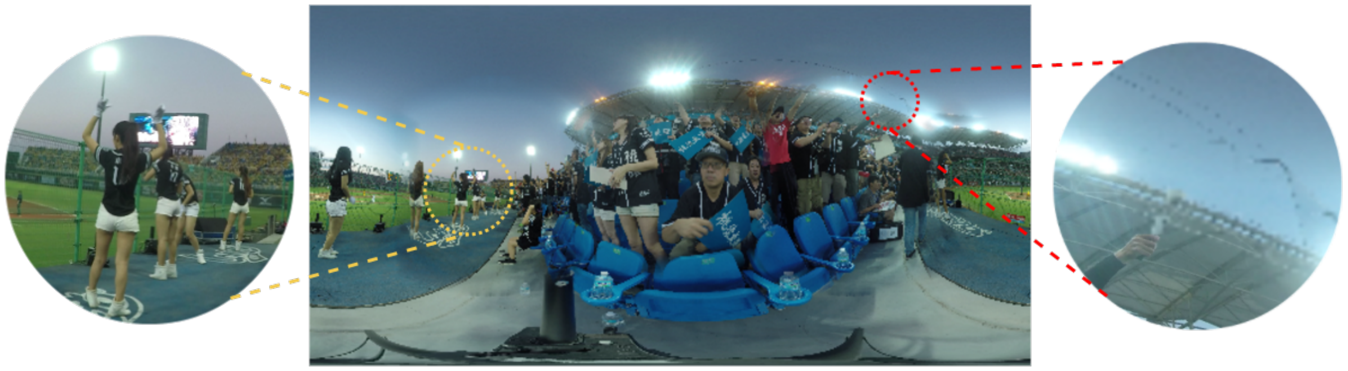

Figure 1: *In this paper, we present a novel high-resolution video stitching framework for real-time VR. We adapt image resolution and level-of-detail by estimating foveal region. The yellow circle indicates a highly-salient region thus rendered in high resolution, while red circle indicates less important region thus rendered in low resolution. The proposed foveated stitching greatly reduces the number of pixels to be processed and overall graphics computation.*

Abstract
In virtual reality (VR) applications, the contents are usually generated by creating a 360° video panorama of a real-world scene. Although many capture devices are being released, getting high-resolution panoramas and displaying a virtual world in real-time remains challenging due to its computationally demanding nature. In this paper, we propose a real-time 360° video foveated stitching framework, that renders the entire scene in different level of detail, aiming to create a high-resolution panoramic video in real-time that can be streamed directly to the client. Our foveated stitching algorithm takes videos from multiple cameras as input, combined with measurements of human visual attention (i.e. the acuity map and the saliency map), can greatly reduce the number of pixels to be processed. We further parallelize the algorithm using GPU to achieve a responsive interface and validate our results via a user study. Our system accelerates graphics computation by a factor of 6 *on a Google Cardboard display.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Due to maturity of virtual reality (VR) in recent years, various devices and applications are released. Meanwhile, content creation for VR becomes more and more important. For example, with VR goggles, basketball fans can feel like they are attending games courtside. The specialized 360°technologies offer views that most audiences can not afford before by placing cameras in locations like under the basketball hoop, VIP area in a live concert, etc. Among these applications, panoramic images are required to be captured in a real scene and delivered in real-time, aiming to provide the ultimate level of immersion and create a sense of physical presence for the users. For this reason, fast and high resolution stitching and rendering are vital for providing a more realistic, engaging and satisfying VR experience.

To create a full HD 360°video , users can use a camera with two lenses and two sensors pointing in opposite directions (e.g. RICOH Theta S), to capture the full 360°. However, most of those cameras cannot provide ultra high resolution (4K), which is extremely important for providing immersive virtual reality experience. Alternatively, one can use several cameras (e.g. GoPro) and a professional rig (e.g. Freedom360) (see upper-left Figure 2) to capture multiple videos. These videos can be aligned and stitched together to generate ultra high resolution 360°video. However, the time and computation complexities for stitching and displaying this ultra high resolution (4K) video is still too high that prevents live video streaming.

In this work, our goal is to deliver 4K resolution live 360°panoramic videos for real-time VR. Inspired by (i) the falloff of acuity in visual periphery in human eyes and (ii) the visual attention cognitive process, we propose a perceptual modulated gaze-contingent framework called foveated stitching, to optimize content delivery. By estimating region of focus and adapting image resolution in level of detail (LOD) fashion, our method can omit unperceived detail, thus stitch and render far fewer pixels. Exploiting foveation is used for efficient graphics processing before [MDT09, CM16, PKS*16, Lue16]. However, unlike the controllable latency environment in [GFD*12], our setting in VR has the following challenges: (i) unstable latencies from different parts of the system, (ii) high computational cost of rendering high resolutional videos. Specifically, we set the proper foveal rendering diameters according to the estimated system latency [SCM15]. As the foveal regions of the users are estimated by tracking their gazes, when we lost track of that due to system delay, we assume the users direct their gazes to salient regions. We construct video saliency map by a fast feature extraction and tracking method. Apart from this, we utilize GPU to greatly speed up the stitching process.

This paper makes several contributions. First, we propose a gaze-contingent video stitching approach to generate high-resolution 360°panoramic videos. The performance benefits can be accrued through the use of LOD in the time-critical domain of VR. We demonstrate significant performance advantages using foveation during stitching and rendering the panoramic videos. Second, we carefully analyze asynchronous communication between our system components (VR headset, server and network) to determine overall system latency. Latencies critically affect how foveated rendering is perceived and how much computational time it can save. Third, we experimentally verify our method through user studies. The study results suggest that our perceptual modulated stitching is free of objectionable artifacts and achieves comparable quality to non-foveated stitching. We demonstrate panoramic video results captured with 6 cameras, allowing the generation of panoramic video in ultr-high resolution (4K).

2. Related Works

Panoramic Video. The idea of combining content from multiple cameras into a wide field of view panorama attracts much attention recently. Lee et al. [LKK*16] proposed deformable spherical projection to minimize parallax from multiple cameras. Perazzi et al. [PSZ*15] presented a pipeline to generate panoramic video from an unstructured camera array. While it successfully minimized distortion in the output, its high computational complexity hinders its use in real-time scenario. Jiang et al. [JG15] proposed a content-preserving warping technique to better remove parallaxes across views. However, it utilized computationally expensive 3D graph cut on spatial-temporal volumes, and thus is not suitable for online video stitching task. Huang et al. [HCC*14] proposed a dynamic seam adjustment scheme to avoid the moving objects in the overlapped area. Yet, their method only covers 360°horizontal views but does not include all the vertical views, which thus cannot be used in VR.

Fast Video Stitching. El-Saban *et al.* [ESRKAH09] proposed a real-time video stitching system on mobile devices. It receives video streams from different mobile phones, and produces a single composite mosaic video using synchronized streams. But the video resolution of mobile devices is currently too low (less than 2K) for high quality VR application. Chen et al. [CWL10] also presented a video stitching method and is able to refine projection transform dynamically. However, similar to [ESRKAH09], the output resolution can not support high resolution display. To speed up the stitching process, Liao et al. [LHC12] transplanted the algorithm onto GPU to enhance the performance. However, exposure compensation and blending process are not included in their framework, which makes it inappropriate for high-quality VR. In contrast, we exploit the falloff of acuity in the visual periphery, combined with GPU optimization, to approach real-time performance.

Video Saliency Detection. Visual saliency is a concept that has been derived from human perception and describes how likely a simulus attracts the attention [GZ10, MPWB15, SS12, BI13]. Mahadevan et al. [MV10] introduced an unsupervised spatiotemporal saliency extraction algorithm on dynamic scenes and moving camera scenario. Although they provide a robust way to extract the saliency under challenging situations, the computation time is too high to be applied under real-time constraint. On the other hand, Cui [CLM09] proposed a fast saliency extraction method for video based on fourier spectrum analysis. However, they only focus on extracting foreground objects from the background.

Virtual Reality Panoramas. Recently, there are many works focusing on capturing scenes and materials from real world to provide user immersion experience in VR [AGB*16]. Fibbi et al. [FSSS15] introduced a prototype that implements the idea of traveling experience sharing in VR using Oculus Rift. The six recorded videos are merged into a single panoramic video using a commercial software (Kolor Autopano Video). Yet, it is still an offline solution to providing virtual travel experience. Similarly, Kasahara et al. [KNR14] designed an experience sharing system with wearable camera headgear that provides 360°spherical images of the user's surrounding environment. They focused on the stabilization problem and can only achieve real-time performance under limited resolution. In contrast, we investigate the stitching pipeline, aiming to provide a high-resolution stitching solution for real-time panoramic videos.

Commercial Software. Producing 360°videos from multiple cameras is still challenging, but it didn't slow down the development of related editing softwares. One can use a popular solution, e.g. VideoStitch [Vid], Kolor [Kol], to stitch several videos into one panoramic video. Yet, the performance achieved by these softwares are far from real-time. For example, it takes 0.07 second

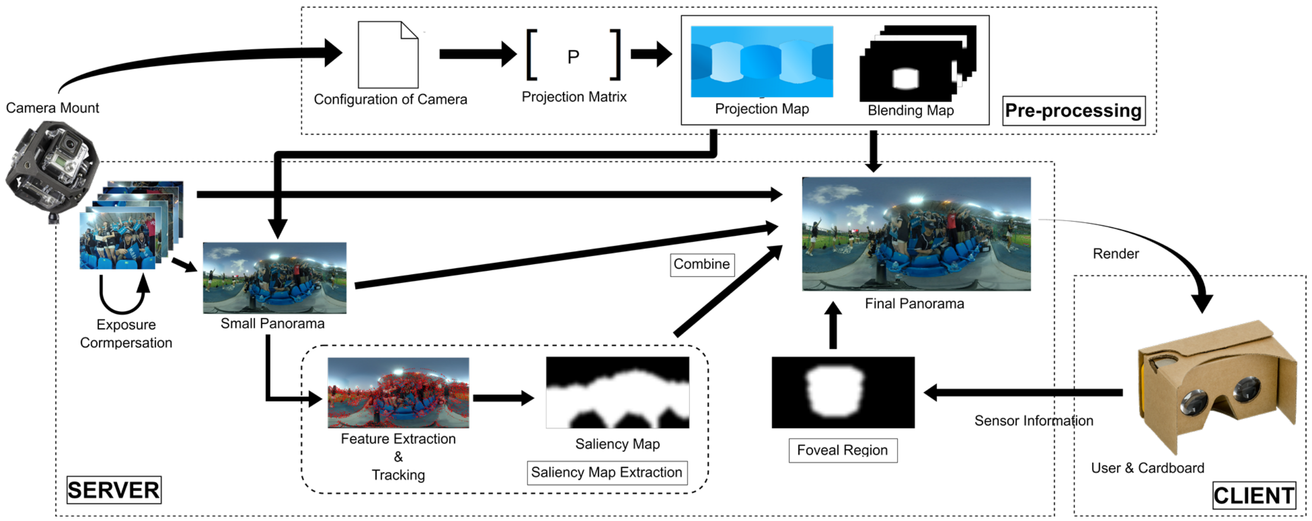

Figure 2: *The workflow of our 360° VR streaming system. Given source videos captured from multiple cameras, we estimate the reference projection of each camera by common calibration process on the server side. The corresponding projection and blending maps are constructed for offline preprocessing in this stage. On the client side, the user's head orientations are collected for estimating the acuity maps. The saliency maps are constructed by extracting and tracking salient features in the video. The estimated acuity map and saliency map are served as input for our foveated stitching algorithm. Our foveated stitching method can generate the final warped projections of each view in different level-of-details. Exposure compensation is also applied for each frame in this step. Our method reduces the computation complexity by removing extra (or unnecessary) details, while introduces least amounts of perceptual artifacts. Finally, the output panoramic video is encoded and streamed to the client via wireless network.*

per frame to generate a 1k panoramic video from six separated cameras of 2k resolution. Even the panoramic resolution is reduced to 1k from it's input resolution (i.e., 2k), the processing speed can only reach 12 fps. Apparently, it cannot be used in real-time scenarios, and such a low resolution output even makes user feel unreal in the virtual world.

Perceptually Lossless Rendering. Prior psychological and perceptual rendering literatures [HL97] [LW07] [MMG11] [LMS10] suggest that perceptually lossless rendering, in which lossy renders are indistinguishable from their non-degraded counterparts, is possible. Guenter et al. [GFD*12] improved graphics performance by employing foveation, in which the region corresponding to the central field of view is rendered with higher fidelity than the region corresponding to the periphery. Fujita et al. [FH14] proposed a foveated rendering technique based on ray-tracing to facilitate a new immersive experience for head-mount display VR applications. However, the effectiveness of previous foveated rendering approaches are still hampered by long latency in current uses of VR environments. Our work builds upon these ideas by extending the reduction of peripheral resolution in a virtual reality scene to a perceptually modulated level-of-detail stitching to compensate severe system latencies.

3. Overview

Our system is designed under a client-server architecture. The input are 6 videos captured by GoPro Hero4 at the frame of 30 fps, where the relative positions of the cameras are fixed. In the preprocessing stage, our system estimates (i) a projection map and (ii) a blending map for each camera on server side. The projection map is used to warp each video onto a common surface, and the blending map is used to enhance final video composite quality. A user wears a Google cardboard at the client side, where the sensor data from cardboard device are collected and transmitted to the server. The user's eye gaze is predicted using the obtained head's orientation. Our system generated different level-of-details by estimating the foveal region using estimated gaze and the focusing visual attention (Section 4). Finally, the output panoramic video are streamed to user's cardboard display through wireless network. Figure 2 is an overview of our real-time video foveated stitching algorithm.

3.1. Preprocessing

First, we get N input high resolution videos (each with T frames), $I_{i,t}, i = 1,\cdots,N, t = 1,\cdots,T$, and we assume the videos are synchronized and the configurations of cameras are static over the entire input sequences. In this stage, the goal is to find a projection map and a blending map for each video so that all the video frames could be aligned to a common reference canvas without noticeable artifacts.

Generation of Reference Projection A common first step for combining views from multiple cameras is to estimate their mutual poses. We take one camera as reference, and estimate relative rotations and translations of the rest cameras to the reference one. With the camera pose information, we estimate the projection matrix that maps each video frame to the user-defined projection surface (spherical and equirectangular projection). Specifically, for every pixel position (x,y) in the projection surface, we get its corresponding pixel in $I_i(x',y')$ by:

$$M_i(x,y) = (x',y'),\ where\ i \in N, (x',y') \in I_i, \tag{1}$$

where M_i is the reference projection mapping function obtained by the calibration process. Note that the size of M_i is the same size as our output panoramic video (4320×2160 in our experiments). This initial global alignment defines the common reference frame for the remaining video frames. The target panoramic stitching result could be generated by blending all the corresponding pixels for each pixel in M_i.

Generation of Blending Map To handle the exposure difference between cameras and get a smooth output, we estimate the blending function for each camera. To further speed-up the blending process, we pre-compute a fixed blending matrix for each video to reduce the per-frame effort while stitching. We adopt alpha blending with weighting maps calculated for all six views where the weight for each pixel is calculated according to its $L1$ distance to the center. Finally, we merge all weighting maps and normalize it to generate the final blending matrix B_i for each video.

Panoramas Projective Geometry With the reference projection map M_i and blending map B_i, our goal is to stitch $I_{i,t}$ together to generate a output panoramic video F_t. For each pixel (x,y) in output frame F_t at time t, we look-up the mapping matrix M_i to get its corresponding pixel in video I_i. Next, we apply the blending function B_i to each pixel (x,y). In addition, we also handle exposure compensation to get a smooth panoramic video. We analyze the input videos and compute exposure adjustments by the method proposed by Brown *et al.* [BL07]. The compensation matrix is denoted by E_i where $E_{i,t}$ represents the intensity compensation for i-th video at time t. We summarize our projective geometry in matrix form as followed:

$$F_t(x,y) = \sum_{i \in N} M_i(x,y) * E_{i,t}(x,y) * B_i(x,y), \tag{2}$$

where $M_i(x,y) = I_i(x',y')$ represents the projected video frame of pixel (x,y) according to the reference projection, and i denotes the video index and N denotes the total number of videos. As shown in Eq. 2, the stitching process is a per-pixel operation, which grows linearly with output size. We illustrate our acceleration technique in Section 4.

3.2. Transmitting and Rendering

After we stitch six input videos, the output is a panoramic video which covers 360°field of view. Next, we encode video file to H.264 video codec, and then stream it to the client side through wireless network. On the client side, the received data stream is decoded and is displayed in VR. Specifically, we create a sphere geometry in the VR world and put our virtual camera into the center of the sphere. After that, we display our panoramic video as the texture of the sphere. The movement of user's head will be synchronized with movement of virtual camera in order to give user immersion experience in VR.

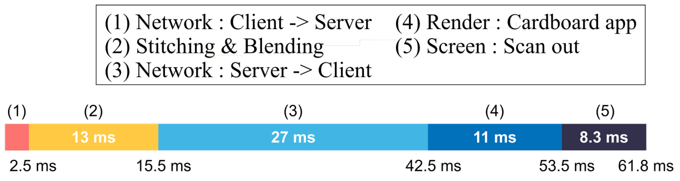

Figure 3: *System latency analysis. The network latency of transmitting sensor data from client to server is* 2.5 *ms. With sensor data, the average processing time for stitching and blending is about* 13 *ms. This part does not includes saliency map estimation as it does not depend on the input sensor data. Once the output panorama is generated, it takes* 27 *ms to transmit from the server to the client through wireless network. On the client side, it takes* 11 *ms to finish the rendering process. Since the screen update frequency of our test device is* 60 *Hz, the average latency is half of the cycle, that is* 8.3 *ms. The total latency of our system is* 61.8 *ms in average.*

4. Gaze-contigent framework

In this section, we develop our technique to accelerate the stitching process by constructing foveal region and saliency map for the input videos. While the foveal region is applicable to save great amounts of graphics computation, this advantage can be hindered by severe the system latency. We first analyze the system latency in Section 4.1. Then, we introduce our acuity map and saliency map construction in Section 4.2 and 4.3. Finally, blending are used to generate final result.

4.1. System Latency

Ideally, we can render image layers around tracked gaze points at progressively high angular size but lower sampling rate to reduce the full cost of rendering [GFD*12]. Yet, the performance achieved by this method is far from real-time. For example, it takes 0.07 second per frame to generate a 1k panoramic video from six separated cameras of 2k resolution. Even the panoramic resolution is reduced to 1k, which is fairly low compared to the input resolution, the processing speed can only reach 12 fps. Apparently, it cannot be used in real-time scenarios, and such a low resolution output even makes user feel unreal in the virtual world.

Perceptually Lossless Rendering. Prior psychological and perceptual rendering literatures [HL97] [LW07] [LMS10] suggest that perceptually lossless rendering, in which lossy renders are indistinguishable from their non-degraded counterpart. According to this, we can reduce the full cost of rendering by only rendering image layers around user's gaze at increasing higher angular size but lower sampling rate. However, its usage depends critically on having extremely low response times of the virtual environment (including panorama computation and rendering) to user interaction (head movement). The lag problem between user's head pose (position and orientation) changing and the updating of the displayed panoramic image, is most difficult to handle in the visual domain. The main reason is because current VR system doesn't have the ability to transmit and render such huge amount

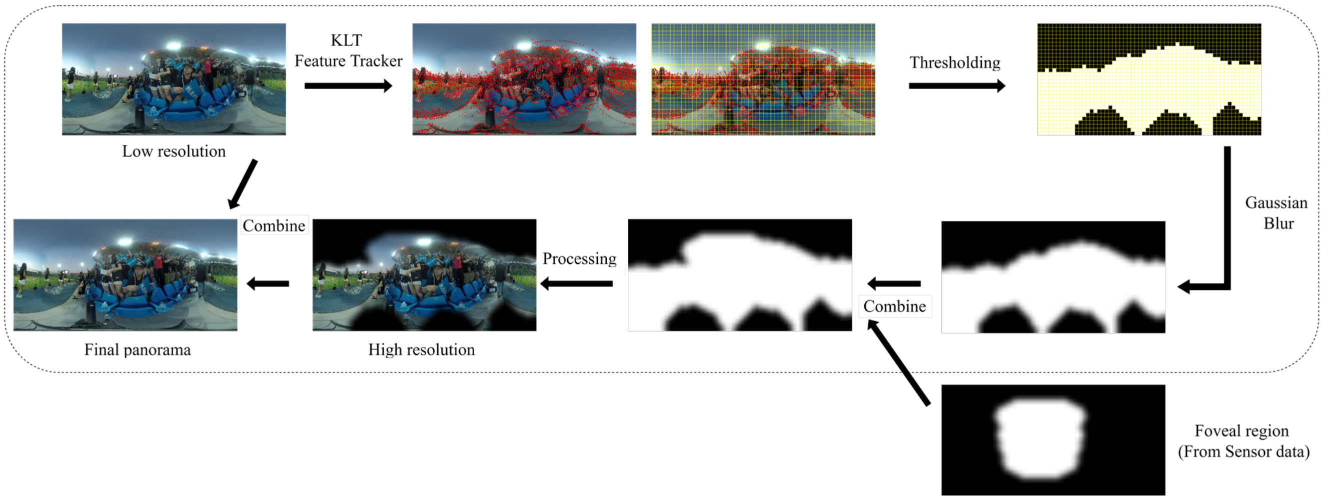

Figure 4: *The pipeline of our video saliency estimation and final rendering. We first generate a low-resolution panorama as the input of our saliency detection algorithm. Next, we divide the panoramic image into uniform grids and track salient features of it using KLT tracker. We generate a binary mask by thresholding the input image according to the feature density inside each grid and apply Gaussian filter to smooth it. The mask is considered as our saliency map, and it is used to estimate where is the user's gaze, in order to compensate the system delay. And we combined with the estimated saliency map and the acuity map derived from the foveal region as the final mask. We render the final panorama in high-resolution for the bright area and low-resolution for the dark area in the mask.*

of data in a responsive frame rate. In addition, for wireless VR headset (e.g. Google Cardboard), it usually transmits data through wireless network with slower transmitting speed and higher latency compared to wired network. Furthermore, the stitching process introduces extra latency which leads to a mismatch between image quality and the actual gaze location . This will introduce noticeable downgrade in viewing quality.

We measure the average latency of each component and analyze asynchronous device interaction to estimate the overall latency. There are five main latencies in our system: network delay (client to server), panorama stitching, network delay (server to client), scene render time, and monitor scan out delay. We analyze individual latency, the result is shown in Figure 3. It only takes 2.5 ms to transmit sensor data from the client to the sever since data is small. The panorama computation takes 13 ms. It takes 27ms to transmit a 4K output video via wireless network. Scene rendering latency is 11 ms, and the cardboard display latency is 8.3 ms. The overall latency is 61.8 ms in average. To achieve real-time performance, we must develop novel algorithmic solutions to combat the high latency.

4.2. Foveated Stitching

Real-time stitching and rendering still operate under the assumption that a single non-degraded render will be fully performed at any single point in time. Yet, with the necessary jump to 4K displays and their application in VR, delivering a 4K 360°video in real-time through standard pipeline remains difficult. The main reason is due to the computational cost required by the pixel-wise operation in stitching process, and the delay caused by asynchronous communication between system components. To accelerate graphics computation, we propose a foveal stitching framework based on the gaze-contingency paradigm, resulting in reduced variable pixel densities to meet the real-time requirement.

As only a finite number of nerve fibers can emerge from the eye, the eye trades off sparse sampling in the periphery for sharp, high resolution foveal vision [BBR09]. Inspired by the fact that peripheral vision is substantially worse than foveal vision, we employ foveation, in which the region corresponding to the central field-of-view is rendered with higher fidelity than the region corresponding to the periphery. Foveal region of human vision spans approximately $2^\circ \sim 5^\circ$of visual angle. The foveal region can be computed as the intersection of the foveal cone and the screen plane, which depends on the eye's viewing angle, and would vary with gaze position [Can13]. Since users' head movements are usually consistent with their gaze directions, we use the sensor orientation (yaw, pitch and raw) on the VR device to predict eye gaze direction.

To realize the performance benefit from foveal region, we need to manage the system latency as the system is tracking the eye's point of focus. To deal with that, an intuitive solution is to enlarge the size of the foveal region. However, increasing the foveated size would also increase the computational resources. Thus, to compensate the delay, the foveal region diameter can be increased such that the true foveal field-of-view is always contained within the rendered foveal region, for some estimated overall system latency. We calculated the adjusted foveal diameter as described in [SCM15]:

$$F_\phi = 2\rho_{pixel} d_u tan(L_{tot} S_{max} + \frac{\alpha}{2}) + 2b_w + c, \tag{3}$$

where ρ_{pixel} is the pixel density of the screen, d_u is the distance between user and the screen, α is the angle subtended by the fovea which is around 5°. L_{tot} is the total average latency of the system, S_{max} is the estimated maximum saccadic speed (we set it as 0.2

degree/ms), b_w is the width of blending border and c is an error constant. Here we assume the user remains at a constant distance from the screen between each tracking frame.

Nevertheless, our internal experiment confirms that the variance of system latency in the environment could be very large due to unstable network condition. In the worst case, the foveal diameter derived from the above equation would still be too large, and such a large size will significantly reduce the system efficiency and hamper the effectiveness of foveated stitching. In the following section, we propose a saliency-aware level-of-detail approach to handle the situation when the user doesn't focus on foveal region.

4.3. Saliency-aware Level of Detail

Visual saliency is a subjective and perceptual quality which makes certain areas of an image stand out during visual observation. The saliency map to accelerate computation can help us attribute a value to each object in the scene according to how salient and how noticeable it is. Such objects gaining a low saliency value will be rendered with a low level-of-detail. A majority of the previous saliency models use centersurround filters or image statistics to identify salient patches that are complex (local complexity/contrast) or rare in their appearance (rarity). To generate a continuous saliency map in videos, we track salient features on a down-sampled panorama by KLT tracking technique [TK91]. To further reduce the efforts to maintain the saliency map S, we divide the panorama into uniform grids with a fixed size λ as shown in Figure 4. We set $\lambda = 8$ in our experiment. After that, we examine the number of features located in each grid and assign a binary number to this grid if the number of features is beyond the threshold. The function THRESH(., ε_f) assigns 1 to a pixel if the feature density of a grid g is greater than a pre-defined threshold ε_f. For each grid g, it's saliency is determined as:

$$S_t(g) = \begin{cases} 1, & \text{if THRESH}(f_t(g)/S_g, \varepsilon_f) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $f_t(g)$ represents number of features in the grid g at time t. We use S_g to denote the grid size. To blend the high resolution regions and low resolution regions together, we apply a Gaussian filter to make the border of the binary map looks smoother. In addition, Gaussian filter is applied in temporal domain to encourage the temporal coherence of the estimated saliency map. The final panoramic image is rendered in full resolution at high salient region and smoothly decrease resolution outward from that. Figure 4 illustrates how the computation of saliency map and it's usage in final rendering.

5. Experiments

We have conducted several experiments to show the effectiveness of our stitching framework in this section. We also discuss a number of results created with our method. Panoramic video results are provided in the accompanying videos and supplemental materials.

Data Setups and Implementation Details. The video data were captured using 6 GoPro Hero4 cameras, each operating at 2704×1520 resolution. We collected sequences at a baseball court. After capturing, the videos are streamed to the media server through wireless network. In our experiment, we used a computer with quad-core Intel i7-3770 CPU @3.40 GHz, 24 GB RAM and a GTX 980 GPU as server. On the server side, we downsampled each video to 480×240 to compute the saliency map as illustrated in Section 4.3. We set $\lambda = 8$ and $\varepsilon_f = 0.04$ through all experiments in this paper. We used Sony XPeria Z as client side, and displayed 360°video on Google Cardboard. Since there is no android phone with 4K resolution that can be installed into Google cardboard,

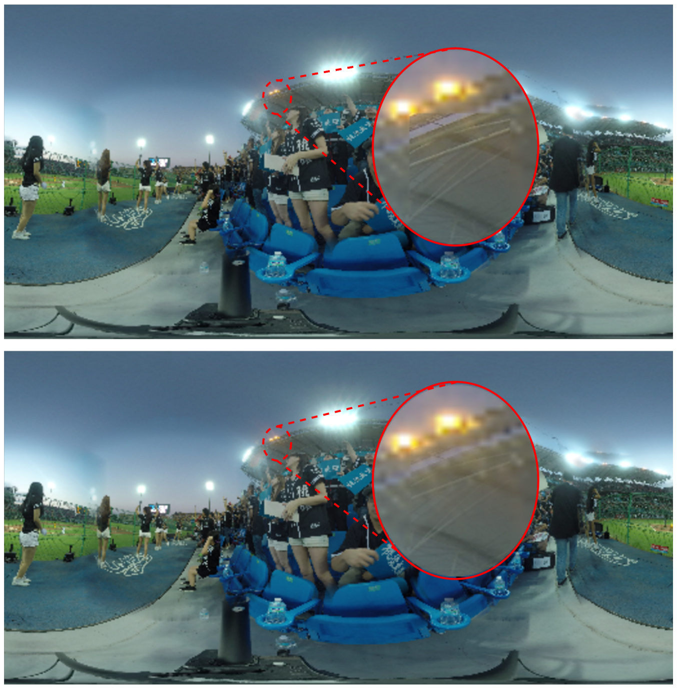

Figure 5: *In this figure, we show the difference between blending and non-blending results. The upper image is generated without performing blending between low resolution and high resolution boundaries. We can see the noticeable structural discontinuity. In contrast, the lower image demonstrates a more smooth result.*

Table 1: *The result of our user study. For both sequences, our methods obtain comparable scores with full resolution ones. Specifically, for sequence 1, our method scores* 6.26 *while full resolution rendering scores* 6.68*. For sequence 2, our method scores* 5.95 *while full resolution rendering scores* 6.11*.*

Seq.	Case	Rendering type	Avg. score
1	1	High resolution 1920×960	6.68
	2	Low resolution 960×480	2.68
	3	Ours ($\varepsilon_f = 0.03$) 1920×960	5.95
	4	Ours ($\varepsilon_f = 0.04$) 1920×960	6.26
	5	Ours ($\varepsilon_f = 0.06$) 1920×960	5.05
2	6	High resolution 1920×960	6.11
	7	Low resolution 960×480	3.05
	8	Ours ($\varepsilon_f = 0.03$) 1920×960	5.90
	9	Ours ($\varepsilon_f = 0.04$) 1920×960	5.95
	10	Ours ($\varepsilon_f = 0.06$) 1920×960	5.58

Table 2: *A statistical result of our user study. The p-values suggest that the participants confused the full resolution rendering results with ours.*

Rendering type 1	Rendering type 2	p-value	Sig. Diff.
Full (1920)	Ours ($\varepsilon_f = 0.03$)	0.05	x
	Ours ($\varepsilon_f = 0.04$)	0.11	x
	Ours ($\varepsilon_f = 0.06$)	8×10^{-4}	o
Full (960)	Ours ($\varepsilon_f = 0.03$)	2×10^{-12}	o
	Ours ($\varepsilon_f = 0.04$)	3×10^{-13}	o
	Ours ($\varepsilon_f = 0.06$)	4×10^{-10}	o

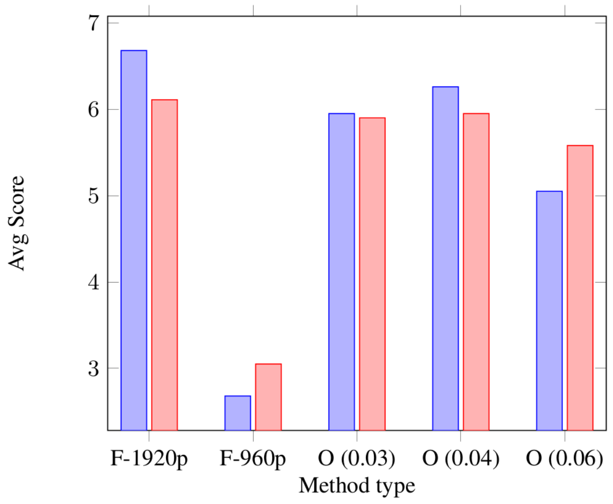

Figure 6: *Performance comparison between each setting. Blue bars indicate the scores of each case from seq. 1, while red bars denote those from seq. 2. 'F' denotes full resolution rendering while 'O' denotes our foveated stitching approach.*

Table 3: *Measured performance at selected configuration of our CPU implementation.*

Seq.	Case	Rendering type	Avg. FPS
1	1	High resolution 4320×2160	3.05
	2	Low resolution 2160×1080	9.44
	3	Ours ($\varepsilon_f = 0.03$) 4320×2160	3.08
	4	Ours ($\varepsilon_f = 0.04$) 4320×2160	3.47
	5	Ours ($\varepsilon_f = 0.06$) 4320×2160	4.20
2	6	High resolution 4320×2160	3.05
	7	Low resolution 2160×1080	9.35
	8	Ours ($\varepsilon_f = 0.03$) 4320×2160	3.12
	9	Ours ($\varepsilon_f = 0.04$) 4320×2160	3.48
	10	Ours ($\varepsilon_f = 0.06$) 4320×2160	4.20

the video we used in the user study are displayed in 1920×960 provided by SONY XPeria Z display resolution. Please note that we still use 4k resolution as target resolution in other experiments for performance evaluation.

User Study. The user study consists of 10 cases. Cases $1 \sim 5$ are panoramic videos generated from sequence 1, while cases $6 \sim 10$ are generated by sequence 2. Please note that we do not assume there is a foveal region in this study. We evaluated the effectiveness of different parameter values (ε_f) in our approach under full resolution rendering (1920×960). We recruited 21 users (13 males and 8 females) in this study. The 10 cases were shown in random order, and the participants were asked to score quality from 1 to 10 for each panoramic video. A score of 10 indicates participant thinks the video provides high quality viewing experience, while a score of 1 indicates a poor quality. Simultaneously, we collect users' head movement data from the sensors on Android phone inside the Google Cardboard.

The result of our user study is shown in Table 1. In this table, the average score of low resolution (960p in case 2) is 2.68, while its high resolution version (1920p in case 1) has a score of 6.68. By applying multi-resolution gaze-contingent stitching (case $3,4,5$), our method can achieve comparable quality (6.26) with less computational resources. We observe the similar performances in case $6 \sim 10$. The results suggest that users cannot distinguish the quality between the full resolution one and ours.

At 5% significant level, the p-value by performing t-test for the user study result is shown in Table 2. The p-value suggests that the participants confused the results generated by full resolution rendering with the results generated by foveated stitching (ours). Note that using multi-resolution technique does not necessarily suggest that the output quality of panorama is worse than the full resolution one. Since in the optimal situation, the system has low delay such that foveal regions can be estimated correctly using the sensor data. Users will still see the same high quality result as ones by full-resolution manner.

We also investigate in which configuration produces most the acceptable quality equivalent to standard stitching approach as shown in Figure 6. $\varepsilon_f = 0.04$ gives the best results for both sequences.

GPU Acceleration and System Performance In Table 3, we showed the averaged fps for each case. In our method, the foveal region is estimated according to the collected sensor data in the user study. For case 1, the fps rendered by full resolution is 3.05. In contrast, our method achieved 3.47 fps in comparable quality. Please note that the results here are reported under CPU implementation. The other test cases also validate that our performance is better than the full-resolution method. In addition, our test sequences are quite challenging since there are lots of subjects, e.g. the audiences in the baseball stadium, resulting in more salient regions to render. For the other scene with less foreground objects, our performance might be able to demonstrate higher performance gains compared to full resolution rendering. To further boost the performance, we implemented our solution on GPU and showed the performance in Table 4. As the table shown, we obtain significant performance gains increase by using GPU optimization and make the stitching process in real-time under ultra high (4K) resolution.

6. Discussion and Limitation

In our implementation, the reference projection is fixed over time and estimated based on ideal configuration of the camera positions. However, in the real case, panorama stitching is still impeded by parallax between input views. To better improve the alignment

Table 4: *Running time comparison between using CPU and GPU implementation. It shows that we achieve real-time performance to deliver a 4K resolution panoramic video.*

Seq.	Output size	Architecture	Average fps
1	2160×1080	CPU	7.30
1	4320×2160	CPU	3.47
1	2160×1080	GPU	23.76
1	4320×2160	GPU	20.25
2	2160×1080	CPU	7.42
2	4320×2160	CPU	3.48
2	2160×1080	GPU	23.87
2	4320×2160	GPU	20.32

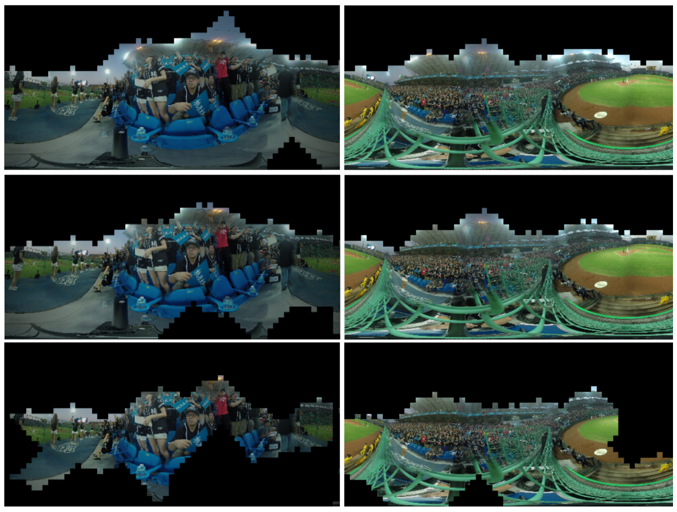

Figure 7: *In this figure, we illustrate how the threshold ε_f influences the detected salient region. From top to bottom, results are shown by setting $\varepsilon_f = (0.03, 0.04, 0.06)$ respectively. The parameter λ is set as 8 under 4320×2160 resolution.*

quality, we would like to extend the basic concept of local warping [ZCBS13, LLCZ16] for parallax removal. Furthermore, we intend to support dynamic reference frame in order to stitch videos from multiple moving cameras. Besides, we would like to implement our solution on high-performance VR devices such as HTC Vive. Several changes to hardware and software elements of our system would make foveation more effective. Two of these are tracking users' eye gazes using more sophisticated methods and adding more additional sensor on VR headset to increase the tracking accuracy.

7. Conclusion

In this work, we proposed a foveated stitching and saliency-aware level-of-detail approach to accelerate the panoramic video stitching in virtual reality system. An perceptual modulated stitching algorithm, powered by an efficient GPU implementation of rendering process, allows users to obtain real-time response from their VR devices without sacrificing the viewing quality. Such techniques could be used in several VR applications such as live game streaming and view sharing, which could bring more realistic immersion VR experiences to users.

Acknowledgements

This work was supported in part by the Ministry of Science and Technology under MOST106-2923-E-002-013-MY3 and MOST106-2221-E-002-211-MY2.

References

[AGB*16] ANDERSON R., GALLUP D., BARRON J. T., KONTKANEN J., SNAVELY N., ESTEBAN C. H., AGARWAL S., SEITZ S. M.: Jump: virtual reality video. *ACM Trans. Graph. 35*, 6 (2016), 198:1–198:13. 2

[BBR09] BENJAMIN BALAS L. N., ROSENHOLTZ R.: A summary-statistic representation in peripheral vision explains visual crowding. *Journal of vision 12*, 4 (2009). 5

[BI13] BORJI A., ITTI L.: State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 1 (2013), 185–207. 2

[BL07] BROWN M., LOWE D. G.: Automatic panoramic image stitching using invariant features. *International Journal on Computer Vision 74*, 1 (2007), 59–73. 4

[Can13] CANOSSA M. S. E.-N. D.: *Game Analytics: Maximizing the Value of Player Data*. Springer, 2013. 5

[CLM09] CUI X., LIU Q., METAXAS D.: Temporal spectral residual: Fast motion saliency detection. In *Proceedings of the 17th ACM International Conference on Multimedia* (2009), pp. 617–620. 2

[CM16] CHWESIUK M., MANTIUK R.: Acceptable System Latency for Gaze-Dependent Level of Detail Rendering. In *EG 2016 - Posters* (2016), Magalhaes L. G., Mantiuk R., (Eds.). 2

[CWL10] CHEN L., WANG X., LIANG X.: An effective video stitching method. In *International Conference on Computer Design and Applications (ICCDA)* (2010), pp. V1–297. 2

[ESRKAH09] EL-SABAN M. A., REFAAT M., KAHEEL A., ABDUL-HAMID A.: Stitching videos streamed by mobile phones in real-time. In *Proceedings of the 17th ACM International Conference on Multimedia* (2009), MM '09, pp. 1009–1010. 2

[FH14] FUJITA M., HARADA T.: *Foveated Real-Time Ray Tracing for Virtual Reality Headset*. Tech. rep., 2014. 3

[FSSS15] FIBBI S., SPANO L. D., SORRENTINO F., SCATENI R.: Wobo: Multisensorial travels through oculus rift. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (2015), pp. 299–302. 2

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3d graphics. *ACM Transactions on Graphics 31*, 6 (2012), 164:1 – 164:10. 2, 3, 4

[GZ10] GUO C., ZHANG L.: A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE Transaction on Image Processing 19*, 1 (2010), 185–198. 2

[HCC*14] HUANG K.-C., CHIEN P.-Y., CHIEN C.-A., CHANG H.-C., GUO J.-I.: A 360-degree panoramic video system design. In *VLSI Design, Automation and Test (VLSI-DAT), 2014 International Symposium on* (2014), pp. 1–4. 2

[HL97] HORVITZ E., LENGYEL J.: Perception, attention, and resources: A decision-theoretic approach to graphics rendering. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (1997), pp. 238–249. 3, 4

[JG15] JIANG W., GU J.: Video stitching with spatial-temporal content-preserving warping. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2015), pp. 42–48. 2

[KNR14] KASAHARA S., NAGAI S., REKIMOTO J.: Livesphere: immersive experience sharing with 360 degrees head-mounted cameras. In *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology* (2014), pp. 61–62. 2

[Kol] Kolor. http://www.kolor.com/. 2

[LHC12] LIAO W.-S., HSIEH T.-J., CHANG Y.-L.: Gpu parallel computing of spherical panorama video stitching. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on* (2012), IEEE, pp. 890–895. 2

[LKK*16] LEE J., KIM B., KIM K., KIM Y., NOH J.: Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transaction on Graphics 35*, 4 (2016), 63. 2

[LLCZ16] LIN K., LIU S., CHEONG L.-F., ZENG B.: Seamless Video Stitching from Hand-held Camera Inputs. *Computer Graphics Forum 35*, 2 (2016), 479–487. 8

[LMS10] LOPEZ F., MOLLA R., SUNDSTEDT V.: Exploring peripheral lod change detections during interactive gaming tasks. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization* (2010), pp. 73–80. 3, 4

[Lue16] LUEBKE D.: Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph. 35*, 6 (2016), 179:1–179:12. 2

[LW07] LOSCHKY L. C., WOLVERTON G. S.: How late can you update gaze-contingent multiresolutional displays without detection? *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 3*, 4 (2007), 7. 3, 4

[MDT09] MURPHY H. A., DUCHOWSKI A. T., TYRRELL R. A.: Hybrid image/model-based gaze-contingent rendering. *ACM Transactions on Applied Perception 5*, 4 (2009). 2

[MMG11] MCNAMARA A., MANIA K., GUTIERREZ D.: Perception in graphics, visualization, virtual environments and animation. *SIGGRAPH Asia 2011 (Course)* (2011). 3

[MPWB15] MAUTHNER T., POSSEGGER H., WALTNER G., BISCHOF H.: Encoding based saliency detection for videos and images. In *IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 2494–2502. 2

[MV10] MAHADEVAN V., VASCONCELOS N.: Spatiotemporal saliency in dynamic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 1 (2010), 171–177. 2

[PKS*16] PATNEY A., KIM J., SALVI M., KAPLANYAN A., WYMAN C., BENTY N., LEFOHN A. E., LUEBKE D.: Perceptually-based foveated virtual reality. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, Emerging Technologies* (2016), pp. 17:1–17:2. 2

[PSZ*15] PERAZZI F., SORKINE-HORNUNG A., ZIMMER H., KAUFMANN P., WANG O., WATSON S., GROSS M. H.: Panoramic video from unstructured camera arrays. *Computer Graphic Forum 34*, 2 (2015), 57–68. 2

[SCM15] SWAFFORD N. T., COSKER D., MITCHELL K.: Latency aware foveated rendering in unreal engine 4. In *Proceedings of the 12th European Conference on Visual Media Production* (2015), pp. 17:1–17:1. 2, 5

[SS12] SCHAUERTE B., STIEFELHAGEN R.: Quaternion-based spectral saliency detection for eye fixation prediction. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)* (2012), pp. 116–129. 2

[TK91] TOMASI C., KANADE T.: *Detection and Tracking of Point Features*. Tech. rep., International Journal of Computer Vision, 1991. 6

[Vid] Videostitch. http://www.video-stitch.com. 2

[ZCBS13] ZARAGOZA J., CHIN T.-J., BROWN M. S., SUTER D.: As-projective-as-possible image stitching with moving dlt. In *IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 2339–2346. 8