即時生成360度虛擬實境全景影片之中央窩影像串接法

李維哲* 陳心怡* 陳明軒* 陳炳宇†
國立臺灣大學
*{wlee, fensi, intere2960}@cmlab.csie.ntu.edu.tw †robin@ntu.edu.tw

ABSTRACT

近年來，虛擬實境(VR)成為時下最迷人的技術, 尤其是沈浸式虛擬實境更是成為眾所矚目的焦點。而要生成這樣的沈浸式虛擬實境的內容，通常必須在現實的場景中利用360度全景拍攝的方式來產生。儘管現在已經有許多拍攝裝置可以使用，但若是在高畫質的狀態下，由於運算量非常龐大，要即時地拍攝360度全景影像並以高畫質顯示仍是非常有挑戰性的。在此我們提出了名為中央窩影像串接法的框架，定義了如何決定影像中的各個部份需要以多高的畫質去處理的方法。在這框架中主要可以分為兩個部份，其一是以人眼視覺的理論基礎去定義的敏銳程度映射函數，其二是基於影像內容對人類視覺的顯著程度來定義的顯著程度映射函數。我們的方法可以以多臺相機拍攝的內容作為輸入，即時地串接成高畫質的全景影片並串流到客戶端的裝置上。速度方面，我們使用了圖形處理器來平行化演算法已達到即時運算的層級。畫質方面，我們做了使用者經驗調查來證明我們產生出的全景影像的畫質並不因為加速而有顯著的下降。我們最終實做了我們的系統於Google Cardboard上，並在速度上相較於原方法有六倍以上的提昇。

CCS Concepts

•**Computing methodologies** → **Image manipulation;** *Computational photography;*

Keywords

即時、環景、360、虛擬實境、VR

1. INTRODUCTION

Due to maturity of virtual reality (VR) in the recent years, various devices and applications are released. Meanwhile, contents creation for VR becomes more and more important.

For example, with virtual reality technology, more fans can have that front row experience. The specialized 360-degree technology offers a view that being in the audience could never buy – placing cameras in locations beyond a front row experience (i.e. under the basketball hoop, VIP area in a live concert, etc.) – and gives the user the feeling of being in a special place. Among those applications, the 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 high resolution 360° video content, users can bring a single piece of camera (e.g. RICOH Theta S), which has two lenses and two sensors pointing in opposite directions, to capture the full 360 degrees. However, most of those cameras cannot provide ultra high resolution, 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) to capture a full panoramic video. But, stitching, combined with display technology, is still time-consuming and not fast enough to provide live video stream under ultra high resolution.

In this work, our goal is to deliver high resolution live 360° panoramic videos for real-time VR. Inspired by the falloff of acuity in visual periphery in human eyes and the visual attention cognitive process, we design a perceptual modulated gaze-contingent framework: foveated stitching, to optimize content delivery. Current VR practice ignores user gaze, thus stitches and renders high-resolution images over the whole display. By estimating region of focus and adapting image resolution and geometric level of detail (LOD), our method can omit unperceived detail, thus stitch and render far fewer pixels. Exploiting foveation in graphics to reduce the processing time is not a new idea. But unlike the previous work [2] that assumed a controllable latency environment, latency in current VR is stll the most critical factors in providing a high quality experience. In addition, video stitching induces extra computational cost. Our work deals with the more critical system latency and computational intensive challenges to successfully realize the performance benefit. Specifically, we set the proper foveal rendering diameters for the estimated system latency [4]. For the non-foveated region, 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° video panorama. The performance benefits can be accrued through the use of LOD in the time-critical domain of VR. We exploit foveation by stitching and rendering the salienct region and the region centered around the current predicted gaze point and for the first time demonstrate a significant performance advantage. Secondly, we carefully analyze asynchronous communication between our system components (VR headset, server and network) to determine overall system latency. Latency critically affects how foveated rendering is perceived and how much it can save. Third, we experimentally verify our method through user studies that suggests our

CGW '16, Taipei, Taiwan
ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Figure 1: In this paper, we present a novel high-resolution video stitching technique for real-time VR. We adapt image resolution (yellow circle = high resolution, red circle = low resolution) and level of detail (LOD) by estimating foveal region. The proposed foveated stitching technique greatly reduces the number of pixels to be processed and overall graphics computation.

perceptual modulated stitching avoids objectionable artifacts and achieves quality comparable to non-foveated stitching. We demonstrate panoramic video results captured with 6 cameras, allowing the generation of panoramic video in high resolution (4K) .

2. OUR APPROACH

Figure 3 is an overview of our real-time high-resolution video foveated stitching algorithm. The input videos are captured from 6 high resolution cameras (GoPro Hero4) at the frame rate of 30 fps. They are assumed to be synchronized and have fixed relative positions. Our video-stitching and live-streaming system is designed under a client-server architecture. In the prepossessing stage, the system first estimates the pair-wise relationship between each camera and calculates the corresponding projection and the blending map at the server side (Section 3.1). The projection map allows to project video frame onto a common projection surface, corresponding to the desired projection model for the panorama. And the blending map is used to enhance final video composite quality as described in Section 3.2. At the client side, a user wears a Google cardboard smart-phone virtual reality headset. The sensor data from cardboard device are collected and transmitted to the server via internet connection and user's eye gaze is predicted by his head orientation. To reduce the sampling density in the peripheral layers and deliver 4K live 360°video streaming, the system generated different levels of detail by estimating the foveal region and the focusing visual attention (Section 4). Finally, the output panorama video are streamed to the client side using Internet. Users can see the final panorama via their google cardboard.

2.1 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 configuration of cameras to be static over the input sequences. In this stage, the goal is to find a warping 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, for all other cameras we estimate their relative rotations and translations to the reference. This process can be done by running a feature-based method or manual alignment. With the camera poses information, we estimate the projection matrix that map 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:

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

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 panorama video. We set it as 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..

For smoothing the output and handling the exposure difference between cameras, we estimate the blending function for each camera. To further speed-up the blending process, we first construct a fixed blending matrix for each video to reduce the per-frame effort while stitching. We adopt feathering blending technique that a weighting map calculated for each video 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.

2.2 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 panorama 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 computes exposure adjustments by the method proposed by Brown *et. al* [1]. 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:

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

where $M_i(x, y) = I_i(x', y')$ represents the projected video frame of pixel (x, y) according to the reference projection. i is the video index and N is the total number of videos. As shown in the above equation, the stitching process is a per-pixel operation, which grows

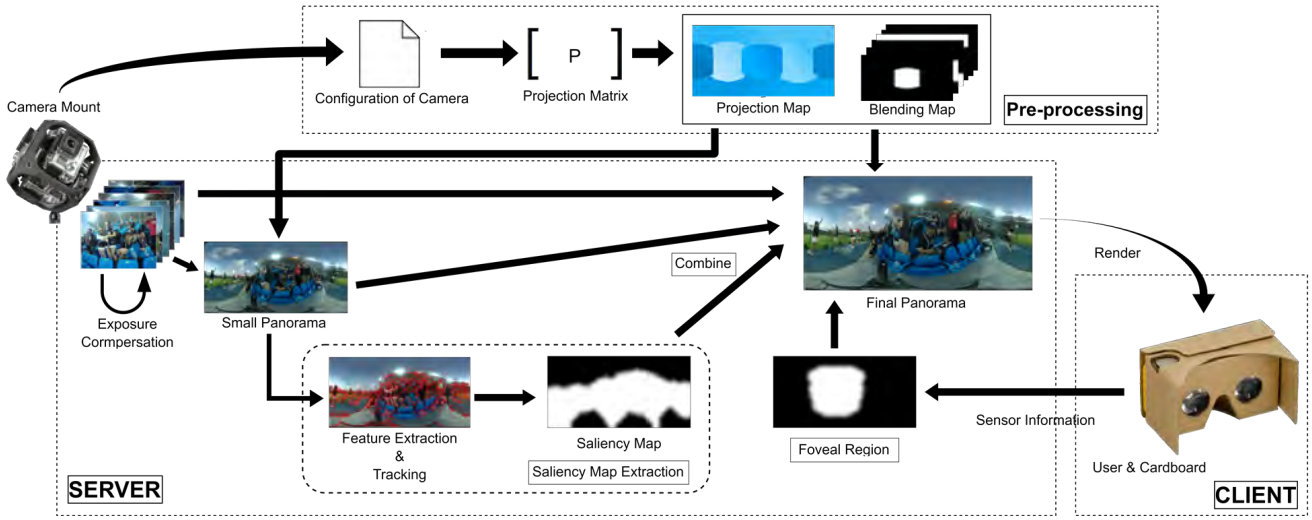

Figure 2: The workflow of our Virtual Reality and 360°Streaming system. Given source videos captured from multiple cameras, on the server side, we generate the reference projection of each camera by common calibration process. The corresponding projection and blending maps are constructed for offline preprocessing in this stage. On the client side, the user's head orientation is collected and sent to estimate the acuity map. The saliency map is 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. With our foveated stitching, we can generate the final warped projections of each views in different levels of details. Exposure compensation is also applied for each frame in this step. In this way, we can remove extraneous detail from an environment which the user cannot perceive, and thus modulate the graphical complexity of a video stitching process with little or no perceptual artifacts. Finally, the output panorama video is encoded and streamed to the client via internet.

linearly with output size. We illustrate our acceleration technique in section 4.

2.3 Transmitting and Rendering

After we construct a single panoramic video by stitching N 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. We use Internet video data transmission. On the client side, the received data stream is decoded and then display the output 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 panorama 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.

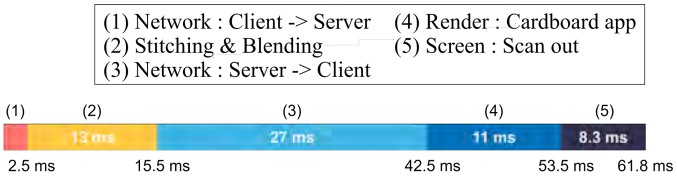

Figure 3: Average System latency. The network latency to send sensor data from client to server is 2.5ms. With sensor data, the average processing time for stitching and blending is about 13ms. This part does not includes saliency map estimation as it does not depend on the input sensor data. Once the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

3. PERCEPTUAL MODULATED STITCHING

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 accelerate many graphics computation, latency critically affects how foveated stitching is perceived and how much it save. We start by 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 as described in Section 4.4.

3.1 System Latency

Ideally, by tracking user's gaze point, we can render image layers around it at progressively high angular size but lower sampling rate to reduce the full cost of rendering [2]. 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 problem, while described as the lag between user's head position or orientation changing and

the updating of the displayed panoramic image, is most difficult to handle in the visual domain. The main reason is because of the sheer quantity of data to transmit and processing required to update the view delivered to the user is beyond the capabilities of current VR system. In addition, for VR framework that use wireless headset (e.g. Google Cardboard), it usually transmits data through Internet, which could not perform as well as Ethernet that is able to transmit in faster speed with lower latency. Furthermore, the stitching process itself introduces extra latency which may raise the risk of processing and displaying an area that user is no longer looking at and rendering the actual area that user focus on in low resolution. This will result noticeable downgrade in viewing quality.

We measure the average latency of each components and analyze asynchronous device interaction to find the overall latency. Five main sources contribute: network delay (client to server), panorama stitching, network delay (server to client), scene render time, and monitor scan out delay. We anlyze 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. To transmit a 4K output video via Internet, it takes 27ms. Scene rendering latency is 11 ms. And the cardboard display latency is 8.3ms. The overall latency is 61.8ms in average. To make real-time performance possible, work is required into the development of algorithmic solutions to combat the high latency.

3.2 Foveated Stitching

Real-time stitching and rendering still operates on the assumption that a single 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 high resolution (4K) 360 video in real-time via standard stitching algorithms remains difficult due to the computational cost required by the pixel-wise operation in stitching process and the delay caused by asynchronous communication of system components. Our goal is to build a fast panorama stitcher that is able to offer 4K live 360 videos generated from multiple cameras. We propose a perceptual modulated framework based on the standard stitching pipeline but with reduced sampling density to fit the real-time requirement.

Our method is motivated by perceptually lossless rendering technique adopted in [2]. It is well established that peripheral vision is significantly worse than foveal vision in many ways, 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. Since user's 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.

Successfully realizing the performance benefit requires managing the system latency. To deal with the high latency, 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 followed [4]:

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

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 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 experimentation 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 saliency-aware level of detail approach to handle the situation that user's focus is outside the estimated foveal region. Here, instead of using a worst case of system latency, we choose to use an average system latency.

3.3 Saliency-aware Level of Detail

Based on the observation that most users direct their gaze to relevant locations rather than looking arbitrarily when watching videos on their VR display, and these locations are usually known to be relevant to the deployment of visual attention. We would like to model the perceptual phenomena with visual saliency to automatically compute the sampling rate.

Visual saliency is a subjective and perceptual quality which makes certain areas of an image stand out during visual observation. The saliency map 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 [5]. 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 predefined threshold ϵ_f. For each grid g, it saliency is determined as:

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

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 boolean map looks more smooth. In addition, Gaussian filter is applied in temporal domain to encourage the temporal coherence of the predicted saliency map. The final panoramic image is rendered in full resolution where saliency is high and continuously decrease resolution outward from that. Figure 4 illustrates our approach.

4. EXPERIMENTS

In this section, we have conduct several experiments to show the effectiveness of our stitching framework. We also discuss a number of results created with our method. Panoramic video results are providing in the accompanying video and supplemental material.

Data Setups and Implementation Details. The video data were captured using 6 GoPro Hero4 cameras, each operating at 2704×1520 resolution. We collected 2 sequences at a baseball court. After capturing, the videos are streamed to the media server through internet. On server side, we down-sample each video to 480×240 to compute the saliency map. We set $\lambda = 8$ and $\epsilon_f = 0.04$ for the experiment. On the client side, we use Google Cardboard with Sony XPeria Z as VR platform so that users can view a 360 degree video

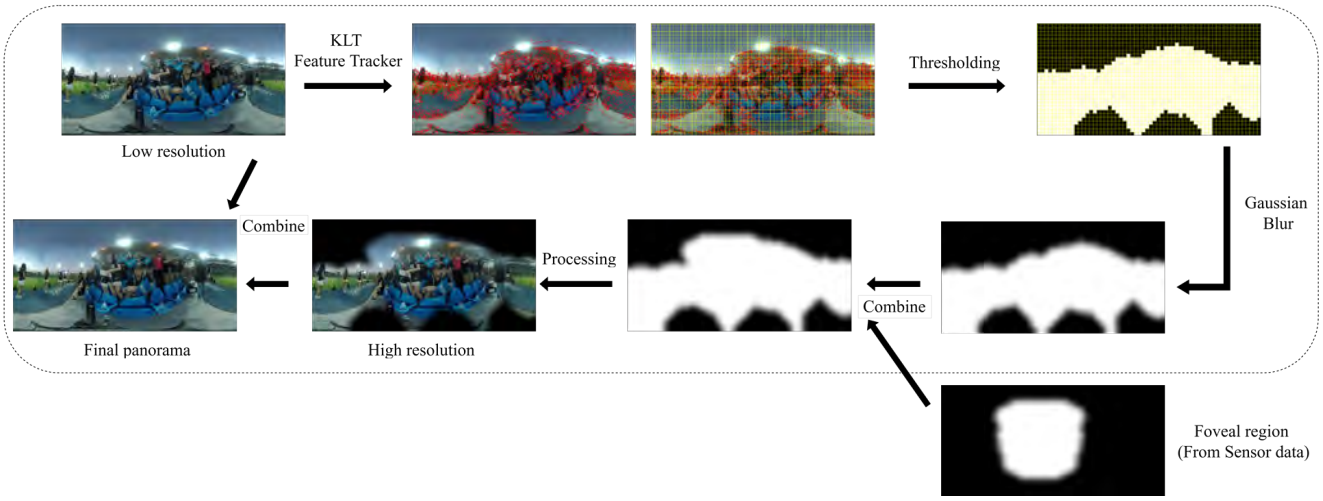

Figure 4: The pipeline of our video saliency estimation. We first generate a low-resolution panorama as the input of our saliency detection algorithm. Next, we divide the panoramic image into uniform grids and track salient features of it using KLT tracker. We generate a boolean mask by thresholding the input image according to the feature density inside each grid. We also apply Gaussian filter on the mask to make it smooth. The mask is considered as our saliency map, then combined with the acuity map derived from the foveal region as the final mask. According to the mask, we can generate the final panorama by rendering in high-resolution for the bright area and low-resolution for the dark area.

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

Table 1: The result of our user study.

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

Table 2: A statistical result in user study. The p-values suggest that the participants confused the result obtained by full resolution rendering and ours.

through a head-mounted display. All our experiments are running on a consumer desktop computer with quad-core Intel $i7$-3770 CPU @3.40GHz, 24GB RAM and a GTX 980 GPU. Since there is no android phone with 4k resolution that can be popped into Google cardboard, the video we used in the user study are displayed in 1920×960. Please note that we still use 4k resolution as target resolution in other experiments for performance evaluation.

User Study. In this user study, we aim to validate that our saliency map can avoids objectionable artifacts and achieves quality comparable to when users look at non-foveated region. 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. They are processed to achieved target output resolutions. Please note that we do not assume there is a foveal region in this study. The full resolution rendering and our approach (with different ϵ_f) are performed to compare their performance. We recruited 21 users (13 males, 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 user's head movement data from the sensors on Android phone inside the google cardboard.

The result of our user study is shown in Table1. 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 performance in the case $6 \sim 10$. Thus our finding suggests 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 suggests that the output quality of panorama is worse than the full

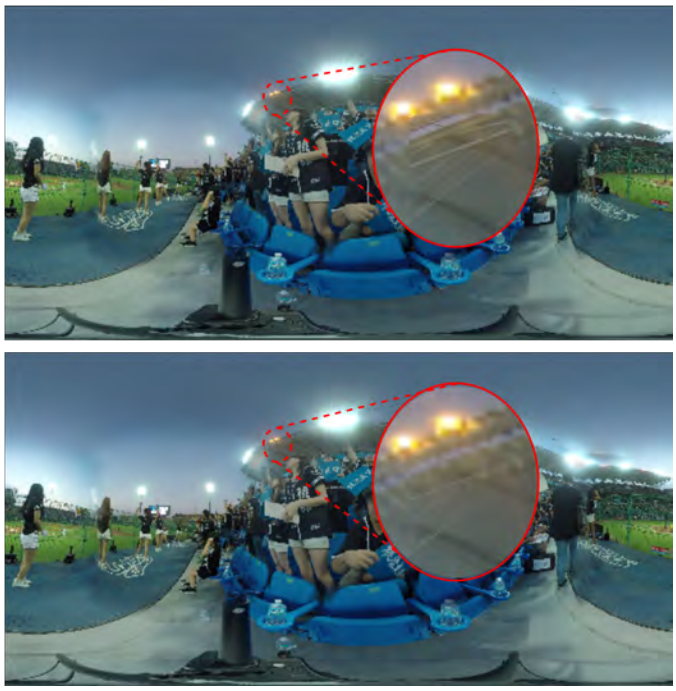

Figure 5: In this figure, we show the difference between blending and non-blending results. The upper image is generated without performing blending between low resolution and high resolution boundaries. We can see the noticeable structural discontinuity. In contrast, the lower image demonstrates a more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

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 acceptable quality equivalent to standard stitching approach as shown in Figure 6. $\epsilon_f = 0.46$ gives the best results for both sequences.

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

Table 3: Measured performance at selected setting in CPU.

System Performance. Measured performance in average fps for each case is shown in Table 3. In our method, the foveal region is estimated according to the collected sensor data in the user study. For the case 1, the fps rendered by full resolution is 3.05. In contrast, our method achieved 3.47 fps in the same 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

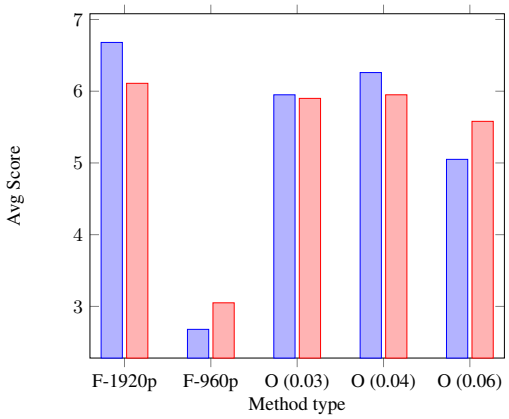

Figure 6: Performance comparison between each setting. Blue bars indicate the scores of each case from seq. 1, while red bars denote those from seq. 2. 'F' denotes full resolution rendering while 'O' denotes our foveated stitching approach.

challenging since there are lot 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 better.

GPU Acceleration. We also implement our solution on GPU. The result is shown in Table 4. As the table shown, we obtain a significant performance increase by using GPU optimization and make the stitching process in real-time under ultra high resolution.

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**

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

5. 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 quality, we would like to extend the basic concept of local warping [3,6] for parallax removal. Furthermore, stitching multiple moving cameras would require a dynamic definition of the reference frame. Besides, we would like to implement our solution on high-performance VR devices such as HTC Vive. By splitting the tasks from the server side to the client side, we could reduce the latency of our system and get a better user experience in VR. 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.

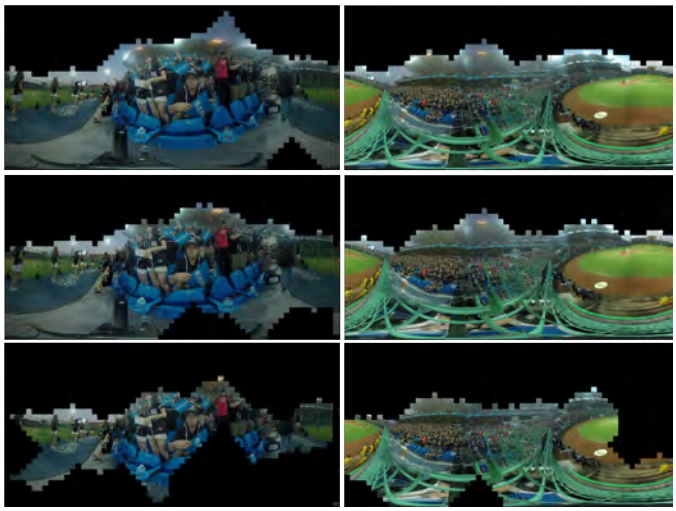

Figure 7: In this figure, we illustrate how the threshold ϵ_f influences the detected salient region. From top to bottom, results are shown by setting $\epsilon_f = (0.03, 0.04, 0.06)$ respectively. The parameter λ is set as 8 under 4320×2160 resolution.

6. CONCLUSION

In this work, we proposed a foveated stitching and saliency-aware level of details 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 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 experience to users.

7. 致謝

本論文感謝科技部經費補助，計畫編號：MOST103-2218-E-002-024-MY3 與 MOST105-2633-E-002-001。

8. REFERENCES

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

[2] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics*, 31(6):164:1 – 164:10, 2012.

[3] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng. Seamless Video Stitching from Hand-held Camera Inputs. *Computer Graphics Forum*, 35, 2016.

[4] N. T. Swafford, D. Cosker, and K. Mitchell. Latency aware foveated rendering in unreal engine 4. In *Proceedings of the 12th European Conference on Visual Media Production*, pages 17:1–17:1, 2015.

[5] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[6] J. Zaragoza, T.-J. Chin, M. S. Brown, and D. Suter. As-projective-as-possible image stitching with moving dlt. In *CVPR*, pages 2339–2346. IEEE Computer Society, 2013.