A NEW SCALABLE FREE VIEWPOINT VIDEO STREAMING SYSTEM OVER IP NETWORK

Zhun Han[1], Qionghai Dai[1,2] Senior member IEEE

[1]Graduate School in Shenzhen, Tsinghua University
[2]Broadband Networks & Digital Media Lab, Department of Automation, Tsinghua University

ABSTRACT

Free Viewpoint Video (FVV), as a new type of multimedia, is a very challenging research area. In order to achieve free viewpoint navigation in realistic scenes, many Image Based Rendering (IBR) technologies are introduced into FVV systems. Tow most important IBR technologies are Light Field Rendering (LFR) and Depth Image Based Rendering (DIBR). This paper will show a new LFR based FVV streaming system over broadband IP networks. Our system uses jump frame method to support scalable quality of service (QoS) to users. An I frame retransmission method in application layer collaborates with RTP/RTCP technology in video streaming ensures different level of Qos.

Index Terms— free viewpoint video, video streaming, light field rendering

1. INTRODUCTION

Free Viewpoint Video (FVV) service can let users do free navigation in 3D scenes through interaction between users and server. A free roam in realistic 3D scene can give users exciting experiences that cannot be given by traditional video stream. In order to render realistic scenes, Image Based Rendering (IBR) technologies are introduced. Two of the most broadly used methods are using Light Field Rendering (LFR) [1, 2] and Depth Image Based Rendering (DIBR) [3] in dynamic scenes. LFR interpolates from multi-camera images to achieve free viewpoint image, while DIBR uses fewer images and a depth map to establish new views.

One of the advantages of LFR based FVV system is it can be used for live 3D show because camera-captured frames can be encoded and transmitted directly to users. However, frames from many cameras may be used in LFR, so FVV based on LFR can be only transmitted in broadband IP network even the streams are compressed. On the other hand, DIBR based FVV transmits much fewer streams and a depth video stream, and can be used in lower bandwidth networks. However, for depth images must be generated offline, DIBR based FVV can not support live streaming.

Transmitting of FVV is different from traditional video streaming in the following points. Firstly, FVV usually include several video streams captured by different cameras, synchronization in streaming process among all cameras must be done. Secondly, video streams required by users may change randomly because of free navigation, so jitter of visual quality due to view switching must be solved. Thirdly, because FVV costs more bandwidth than single video stream, scalable quality of service is more important.

Although many efforts have been done to compress LFR based FVV [4-7], transmitting issues have not been deeply researched. Some relative works appeared stream light field data in static scenes. Bernd Girod et al [8] stream static light field images using R-D optimization. Zagorodnovd et al [9] gives an effective way to smooth data jitter due to view switching using 2D pre-fetching method.

Streaming system for DIBR based FVV is also not fully researched. Researchers in [11] showed a system includes compression, transmitting and display for DIBR based FVV, but they only considered transmitting FVV through DVB network. Goran et al [10] shows a near future streaming system for DIBR based FVV over IP networks. They divide video streams into depth video, texture video and common video, and transmit them in RTP/RTSP individually. However, their method has no solution for view switching and synchronization problems.

For the problems above, research of FVV streaming over IP network is highly desired. This paper put forward a new streaming system for LFR based FVV which designed for solving the problems we mentioned. We will focus on our I-frame retransmission and jump frame techniques in application layer. Collaborated with RTP/RTCP, it can support different level of QoS for users.

The remaining sections of our paper will be organized as follows: in section 2 we will have a snapshot of encoding methods of FVV. We will illustrate physical and logical structure of our system in section 3. In section 4, we will focus on our application layer design. Section 5 shows our experimental results and in sections 6 we reach our conclusion.

Thanks for: The National Science Fund for Distinguished Young Scholars (No.60525111) and Key project of NSFC (No.60432030)
Also Thanks to Yebin Liu's discussion that enlighten us

2. A SNAPSHOT OF ENCODING METHODS

FVV usually includes several video streams (or depth stream), frames in single video stream have their temporal correlations, while frames at the same time among all streams have spatial correlations. Encoding method usually use both of these correlations to compress FVV streams [3, 4, 7, 11]. Hence, streaming system must consider the random access properties when send frames to users.

Shing-Chow Chan et al [4] used an adaptive MPEG-2 encoding method to compress LFR based FVV. They divide all streams into several groups. Every group has a referenced stream used for spatial disparity compensation. However, when users require streams for rendering, the "referenced stream" must also be transmitted. This will not only waste the bandwidth, but also make clients decode more frames. Note that clients of FVV must decode multiple streams, render views and display them. So we do not want FVV clients waste their CPU on decoding many frames not useful.

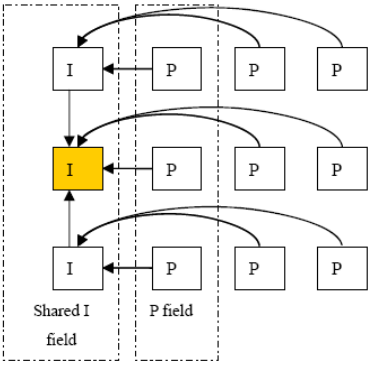

Figure 1 SIF compression method for our streaming system

Many researchers use Multi-view Video Coding (MVC) methods to compress FVV streams [3, 11]. Multi-view video coding scheme like [13] usually depend on multi-reference chain that makes the reference structure more complicated. System in [13] encodes video from each camera into a stream and also encodes two kinds of cross camera streams to support multi-view navigation. Note that different from multi-view video, FVV clients need several streams rather than only one stream at one time, and view switching behavior is more random than multi-view cases. If we use a multi-reference chain structure like [13], we must generate huge number of cross camera streams most of which will not be useful but only a waste of memory. Further more, delivery a new stream (a single camera stream or a cross camera stream) to client will lead to re-buffer in client side.

To give a solution of the problem we mentioned above, Liu et al [7] shows a Shared I Field (SIF) compression technique (shown in Figure 1). This method break the reference structure of original streams and make all frames referenced to frames in SIF. Spatial correlations are only used in SIF. Frames that captured in the same time not in a SIF are called a P field.

The most important advantage of SIF design is decoder need not to decode many referenced frames when a single frame is desired. In this paper, we use [7] as our compression method.

3. STUCTURE OF OUR SYSTEM

Figure 2 shows physical and logical structure of our system. Every 4 1394-port video cameras are linked with a capture PC. Capture PCs do SIF compression [7] to frames and then transmit encoded streams to transmitting serve. Frames are buffered in SIF buffer and P field buffer individually in transmitting server. Synchronization among different video cameras on the same capture PC is done by software, and synchronization among capture PCs is done by RTSP protocol between transmitting server and capture PCs.

When a user access transmitting server for FVV service, the server will first do "pre-send" to users through reliable TCP/IP protocol. The data need to be pre-sent are geometry data of the scene including intrinsic and extrinsic camera parameters.

After pre-sending geometry data, clients will require streams through RTSP protocol based on user's view point. Server will first send frames from a SIF, then send P frames from P fields in order.

Frames in a SIF or P field will be packed in Group of Frame (GOF). All GOF will be further packed by RTP and UDP protocol to smaller packets. RTCP protocol is used for reporting packet loss. Packet retransmission will not be done in RTCP layer. We apply an application layer retransmission scheduler to our system. In fact, we will not do retransmission to all lost GOF packets, but only to ones that include SIF frames. Detailed retransmission design will be illustrated in next section.

When view switch happens due to user's free navigation, client side will send RTSP message informing server new streams user need. Server will pack all SIF frames of new streams into a GOF and send. We support multiple frame rates to users using jump frame technique which will be illustrate in the following section.

To solve the problem of data jam in server side, transmitting server has a single thread design that can always send packets to most urgent user. Our system has an optimized sorting algorithm for scheduling packet sending for users from multiple frame rates. Complication analysis shows our sorting method can select the user in $O(k)$ time rather than $O(n)$ time. Where k is the number of levels which is much smaller than number of users n.

Although our system is designed for LFR based FVV, consider depth map as special kind of video stream, our system can extent to DIBR based FVV streaming.

4. APPLICATION LAYER DESIGN

4.1. I Frame Retransmission & Adaptive Display Client

For we use UDP based RTP protocol for transmitting FVV, packet loss may happen due to bandwidth jitter and other reasons in unreliable IP networks.

If we do retransmission to all lost GOF, it will lead to a waste of bandwidth. Further more, retransmission of all lost packets will force clients create long buffers (usually several seconds) to wait the retransmitting frames arrive. However, due to random view switching in FVV, a long buffer will lead to further waste of bandwidth and strong visual quality jitter. When view switching happens, client need re-buffer new streams and dump old frames which are transmitted but not useful.

In our design, we do not use long buffer on client side. Buffers that have a length of only 0.1-0.5s due to network condition are used. When server get a report of packet loss through RTCP protocol, application layer program will first check which GOF the packet is in. If the packet is in a P field GOF, nothing will be done. For we use compression method in [7], packet loss happens in P field GOF will only lead to decreasing of visual quality on a certain view, and will not affect the following views. If we do some simple error resilience technologies on rendering side, for example, using frames from last field, the user can rarely be aware of the decreasing of visual quality.

If packet loss is happened to a SIF GOF, the frames that lost (usually, one RTP packet include no more than two frames) will be packed into next GOF and sent to the user. The visual quality will decrease a little before the retransmitted packet arrives. Our client is designed that if one stream required by rendering program cannot be decoded, it will use remaining streams for rendering. So a single stream packet loss will not stop the display, but only make quality decreased a little.

Our design of I frame retransmission and adaptive display on client side can save bandwidth and solves view switching problem effectively. Assume delay for a single packet in network is t, and jitter is Δt, usually, we have $t >> \Delta t$. For traditional streaming method, the buffer length must be $k \times (2t), k > 1$, while our client side only need a buffer length of $m \times \Delta t, m > 1$. This will reduce the waste of bandwidth greatly.

4.2. Adaptive Frame Rate

Bandwidth of IP networks will jitter randomly due to many reasons. If bandwidth decreases when doing FVV service to a user and the actual bandwidth becomes smaller than FVV service takes, a large number of packets will be lost due to data jamming. In this case, retransmission method mentioned in last sub-section will not be helpful. We must reduce and bandwidth we use adaptively and dynamically.

For we use SIF compression method, reduce P fields sent to users will not lead to decoding failure of the streams. Our system is designed that if a large rate of packets is lost in a short time, we will reduce the P field frame rate in order to reduce bandwidth.

To illustrate that in detail, we assume packet loss rate for last $k \times N$ GOF is $Rloss(k)$, we define our threshold to reduce the frame rate to be:

$$Rloss(k) > P/k, k = 1...m$$

that is, if packet loss happen in a long term at a lower rate or happen in short term at a higher rate, we will reduce our frame rates.

For the case bandwidth becomes wider again, users can set a period T try increasing frame rate, failure of attempting higher bit rate will make $T = 2T$.

5. EXPERIMENTAL RESULTS

Table 1 shows comparison between our system and "Near Future" one in [10]. It can be seen that our system support view switching better and have more scalable properties.

The hardware environment of our experiments is system shown in section 3. 64 video cameras link with 16 capture PCs do encoding in [7]. Transmitting server buffers the frames and then sends them to users directly to simulate FVV live streaming case. Every camera has a resolution of 320×240 and a frame rate of 30 fps, while transmitting server can support 30fps and 15fps frame rates. Rendering clients use a LRF algorithm presented in [12].

A network module is introduced to delay and drop packets in our experiment. We assume the packet arrive delay by:

$$t = packet\ length / bandwidth$$

with a random jitter $\Delta t = 0.1t$. And assume that if there are more than 5 GOF data jammed in the network sending queue, packet loss will happen.

In our experiment, 9 streams are required by every user. We simulated a FVV streaming process of 100 seconds. With a bandwidth of $2.0 Mbit/s$ at first, a view switching happens on 33rd second and bandwidth decreases to $1.5 Mbit/s$ on 67th second, which is lower than 30fps FVV service need.

Figure 3 shows our experimental results of our streaming process. We value visual quality by number of frames can be used for rendering per second. Frame buffer of varied length is tested in our system from 0.1s to 0.5s.

It can be seen that with 0.5 second buffer, our system can serve users with a smooth visual quality, while a 0.1 second buffer make a little jitter. View switching affects quality little in our design, while a network bandwidth jitter may lead to several second's of quality decreasing if a 0.1s or 0.2s buffer is used. However, quality switches smoothly when apply a 0.5 second buffer on client side.

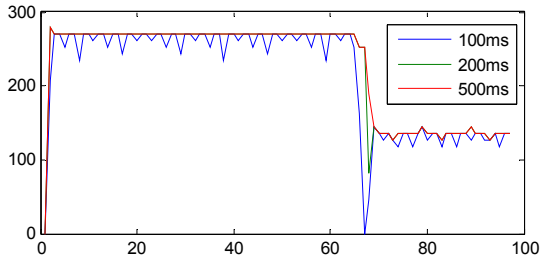

Figure 3 Visual qualities in account of frames per second in our system

6. CONCLUTION

This paper presents our system for streaming LFR based FVV over broadband IP network. We use application layer I-frame retransmission collaborate with RTP/RTCP streaming protocol ensuring video quality in our FVV service. Jump frame technology is used for support scalable quality of service (QoS) to users. Although this system is designed for LFR based FVV, it can also be extent to DIBR based FVV streaming. Experimental results show that our system design solve the view switching and net work jitter problem effectively.

7. REFERENCES

[1] M.Levoy and P.Hanrahan., "Light field rendering", Computer Graphics (Proceedings. SIGGRAPH96), pages 31 42, August 1996

[2] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph", Computer Graphics (Proceedings SIGGRAPH-96), pages 31 42, Aug.1996

[3] Christoph Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV", Proceedings of SPIE -- Volume 5291 Stereoscopic Displays and Virtual Reality Systems XI, May 2004, pp. 93-104

[4] Shing-Chow Chan, King-To Ng, Zhi-Feng Gan. Kin-Lok Chun and Heung-Yeung Shum, "The Compression of Simplified Dynamic Light Field", ICASSP 2003

[5] M. Magnor and B.Girod, "Hierarchical Coding of Light Fields with Disparity Maps", Proc. Int. Conf. Image Processing ICIP-99, Kobe, Japan, pp. 334-338, Oct. 1999

[6] Xin Tong and Robert M. Gray, "Interactive Rendering from Compressed Light Fields", IEEE Trans. Circuits Syst. Video Techn. 13(11): 1080-1091 (2003)

[7] Yebin Liu, Qionghai Dai, and Wenli Xu, "A Real Time Interactive Dynamic Light Field Transmission System", IEEE international conference on multimedia expo.2006, Toronto, Canada ICME 2006

[8] Chuo-Ling Chang and Bernd Girod, "Rate-distortion optimized interactive streaming for scalable bitstreams of light fields", Proc. SPIE Visual Comm. and Image Processing VCIP-2004, January 2004.

[9] Zagorodnov, V and Ramadge, P.J, "Data rate smoothing in interactive walkthrough applications using 2D prefetching", Volume 3, 24-28 June 2002 Page(s):III-201 - III-204 ICIP.2002

[10] Goran Petrovicand Peter H. N. de With, "Near-future Streaming Framework for 3D-TV Applications", ICME2006

[11] Aljoscha Smolic, Karsten Mueller et al, "3D Video and Free Viewpoint Video – Technologies, Applications and MPEG Standards", ICME2006

[12] Aaron Isaksen, Leonard McMillan and Steven J. Gortler, "Dynamically Reparameterized Light Fields" SIGGARPH 2000,

[13] Jianguang Lou, Hua Cai and Jiang Li, "A Real-time Interactive Multi-view Video System", proceedings of the 13th annual ACM international conference in Multimedia, 2005

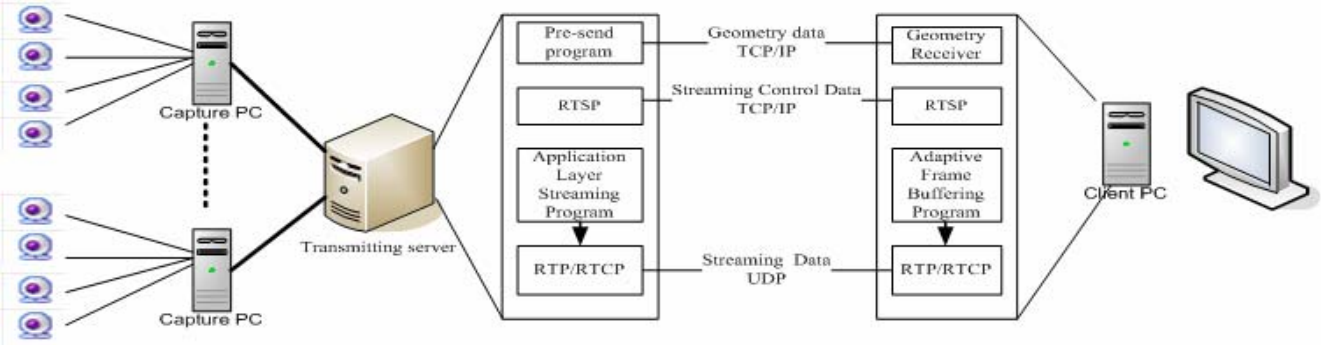

Figure 2: Physical and Logical structure of our transmitting system

	Our method	"Near Future" method in [10]
rendering method support	LFR support, can be extended to DIBR	DIBR only
Scalable property	Both frame rate, and number of streams	Only number of streams
view swithing support	Have support	No special support
Stream classify	Divided to SIF and P field	Due to rendering need

Table 1: Comparison between our system and "Near Future" one in [10]