A LOW LATENCY CLOUD GAMING SYSTEM USING EDGE PRESERVED IMAGE HOMOGRAPHY

Lingfeng Xu[1,*], *Xun Guo*[2], *Yan Lu*[2], *Shipeng Li*[2], *Oscar C. Au*[1], *Lu Fang*[3]

[1]The Hong Kong University of Science and Technology
[2]Microsoft Research, Beijing, China
[3]University of Science and Technology of China
[1]{lingfengxu, eeau}@ust.hk, [2]{xunguo, yanlu, spli}@microsoft.com, [3]fanglu@ustc.edu.cn

ABSTRACT

The emerging cloud gaming technology has been growing fast, driving up huge mobile consumer demands. The video streaming based cloud gaming scenario renders the game scenes in the cloud servers, and streams the encoded sequences to the thin clints where the game scenes are decoded and displayed to the players. However, current existing clouding gaming services have some problems, such as the latency and bandwidth limitation. The size of the video stream is usually quite large which requires heavy transmission. Worse still, the frame data rate will burst when the game scenes contain fast translation or rotation, resulting in strong latency problem. In this paper, we propose a novel video streaming based cloud gaming algorithm which reduces the burst of the frame rate significantly. There are mainly two innovations in this paper. Firstly, based on the analysis of the motion estimation strategy in the video codec, we introduce image homography technique for better motion prediction. Meanwhile, according to the rasterization rules of the game engine, we present a special designed interpolation algorithm named Edge Preserved Interpolation (EPI), for more accurate edge interpolation and further reduce the residues in the edge regions. The proposed algorithm is implemented on the x264 platform. Experimental results show that our algorithm has 18.0% BD-rate reduction compared with x264.

Index Terms— Cloud gaming, video streaming, image homography, edge preserved interpolation

1. INTRODUCTION

Many believe the future of the gaming lies in the cloud. The cloud gaming services become more and more desirable with the popularity of the smart phones and mobile devices. As shown in Fig. 1, cloud gaming systems render the game scenes on cloud servers and stream the encoded sequences to clients over the network. The control events from mice, keyboards, joysticks, and touch screens are transmitted from the clients back to the cloud servers. Compare with traditional game scenario, cloud gaming allows the users to play high-end games with a thin clint without expensive hardwares, such as Television, tablet, set top box, smartphone, etc. Meanwhile, cloud gaming does not require local installation of the games, which reduces the installation time for the users and simplifies the publishing process for the game companies.

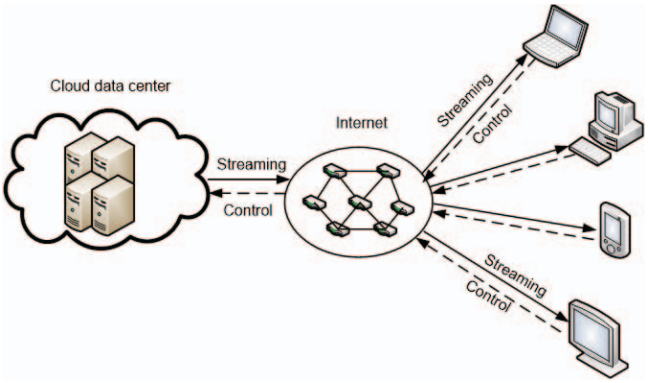

Fig. 1. Cloud gaming systems

The state-of-the-art commercial cloud gaming services are represented by OnLive. Onlive [1] has expanded its game offerings to Vizio TVs as well as mobile platforms. Meanwhile, Gaikai partners with Intel, EA and Bigfoot to deliver free game demos straight to consumers. In the research aspect, Huang et. al provided an open source cloud gaming system named GamingAnywhere [2], for the purpose of extensive research and game development. Nave et al [3] also developed a cloud gaming system: Game@Large, which has the functions of game execution distribution and game streaming.

Generally speaking, there are three different approaches in the current existing cloud gaming systems. The first approach is based on 3D graphics steaming, which transmits the interactive streams of graphical output to the client devices [3, 4]. In the server side, the graphical commands are intercepted. The mesh, vertex, texture, and graphical commands are then compressed and transmitted to the client. The client side will decode the received data and render the scenes lo-

*This work was done when the author was an intern in Microsoft Research, Beijing.

cally. The graphical streaming approach distributes the kernel computation and model rendering, which is suitable for the applications which need heavy computation and are insensitive to the latency. Since this approach requires client rendering, it is not suitable for thin clients such as the mobile devices.

The second approach is based on the video streaming techniques [1, 2]. This approach renders the game in the server, captures the game scenes as video frames, compresses and transmits them to the client. What the client side needs is just to decode the video frames and display them. Video streaming approach is a quite stable and simple method for cloud gaming applications. Therefore, it is well adopted in the commercial systems, like OnLive and Gaika. What the game players need are just good network conditions and thin clients. They can choose any game in the cloud servers, and play them directly without installation. However, pure video streaming needs heavy transmissions. For example, the most popular commercial cloud gaming system, OnLive, needs at least 2 Mbps bandwidth. And it is reported [2] that 130ms transmission latency is observed with 5 Mpbs bandwidth. Meanwhile, the video streaming approach has strong jitter requirements, and is sensitive to packet loss and delay.

In order to solve the latency problem of the video streaming approach, many researchers are focusing on the rendering aided video streaming approach [5, 6, 7, 8]. They utilize the prior information derived from the game engines, such like depth maps, motion vectors and perspective matrices of the view cameras and so on, for pre or post rendering to help reduce the codec's complexity or bit rate. Shi et al proposed an algorithm [6] that utilized the 3D rendering method to generate the reference views for inter view prediction. They introduced the idea of stereo images rendering to handle the occlusion problems. Nevertheless, additional 3D information such as depth maps and perspective matrices are required to be transmitted to the clients, which increases the bandwidth. Another problem of their method is it needs a large buffer to store several previous frames, which is not suitable for thin client applications. Giesen et al [5] proposed an augmented compression method for cloud gaming sequences. They utilized the prior 3D information and the warping equations to derive a candidate motion vector for the encoder. Their method has slight PSNR gains in some sequences, but will have problems when the game scenes contain large translation or rotation. Laikari et al [7] proposed an algorithm which is special designed for sky box technology based games. In their algorithm, the sky region is extracted and rendered to construct the reference frame. Their method reduced the encoder complexity for about 35%, but with an average 0.8dB PSNR loss. And their method can not be well applied nowadays because most of the modern games do not perform the sky box technique.

Our proposed algorithm belongs to the third approach of cloud gaming. Different from existing methods which transmit the depth maps for rendering, our method just transmit the rotation matrices of the view camera. We analyze the property of the motion estimation and find that the camera translation can be well compensated by the motion vector but the camera rotation cannot. Therefore, we intercept the rotation matrix of each view from the game engine, and utilize the information to adaptively perform image homography according to the relative rotation between adjacent views. Moreover, in order to generate more accurate edge for the game scenes, we present an edge preserved interpolation algorithm. The rest of the paper is organized as follows: in Section 2, we will introduce the proposed algorithms. Experimental results are shown in Section 3 and conclusions are given in Section 4.

2. PROPOSED ALGORITHMS

In the video streaming based cloud gaming system, the game scenes are treated as consecutive video frames. Different from conventional video sequences captured by a real camera, the game sequences are computer generated graphs and contain some different properties. The first difference is that game sequences usually contain additional information such as the depth maps, rotation matrices and camera intrinsic matrices while the conventional videos do not. The second difference is that, for some types of games, e.g. first angle games such like *Counter-Strike*, the view angles of the game keep moving around, while the conventional video is more static. The third difference is about the frame properties: compare with conventional videos which usually capture the real world, the game scenes are computer generated graphs which are usually noiseless and contain sharp edges. Those differences compose the special properties of the game videos. In this section, the special designed algorithms are proposed according to these properties.

2.1. Image Homography based Reference Construction

The typical video codec utilizes inter prediction to explorer the inter-frame correlation and reduce the bit rates. When performing inter prediction, the current frame is divided into a few blocks. Motion estimation is applied to each block to find a similar block in the reference frames. After that, the residues of the two blocks are compressed and transmitted. In the case of game sequences, the view angles usually keep moving, especially in the first angle games. And motion estimation will be affected by the camera motions.

The motions of the camera can be decomposed into three types: horizontal & vertical displacement, perpendicular displacement and rotation. Different types of motions produce different affections to motion estimation. Most of the objects keeps static in the 3D space in general. When the camera only contains the horizontal and vertical displacement, it will produce horizontal and vertical translations for the static objects in the 3D space. According to the motion estimation strat-

egy, translations can be easily detected by the motion estimation, leading to small residues for compression. However, when perpendicular displacement for the camera happens, the scales of the objects are changed in the adjacent views. The objects will be enlarged in the adjacent views when the camera is moving forward. And it will be shrunk when moving backward. Since motion estimation is sensitive to the scaling change, the residue would be relatively larger. As for the case of camera rotation, for example, when the gammers want to watch around, they will rotate their view angles. The rotation will produce affine transforms to the objects. The square block will be rotated or transformed into a trapezoid after the camera rotations. Therefore, typical motion estimation would fail when the camera rotation happens.

Fig. 2 shows the relationship between the camera motion and bits used for compression. The test sequence is generated from a first angle game named 'PowerPlant'. We intentionally perform different types of motions in the game and then compress the game sequence by x264 in low delay configuration. It is observed that when the camera only contains the horizontal & vertical motion, the bits used are much smaller than the case of rotation. And the perpendicular motions: forward motion and backward motion, generate bits inbetween the previous two cases. This result verifies our analysis above. One thing worth mentioning is that, much more bits are required to compress the backward motion frame than the forward motion frame. That is because when move backward, large regions of new contents will appear. And it is difficult to find similar blocks for those newly appeared regions.

Some prior algorithms utilized Depth Image Based Rendering (DIBR) to generate better reference frames. But those methods need to pay for the bits for transmitting the depth maps, which usually occupy $1/5$ to $1/10$ bit rates of the streams. In order to save the bits for the depth maps while still generating good references frames for motion estimation, we proposed the image homography based reference construction

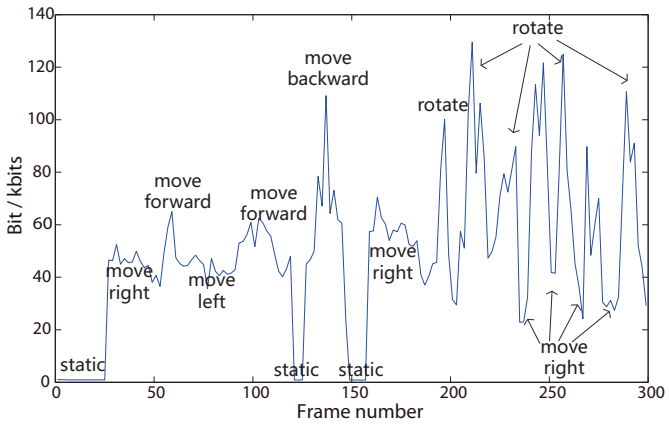

Fig. 2. Relationship between the camera motion and bit rate

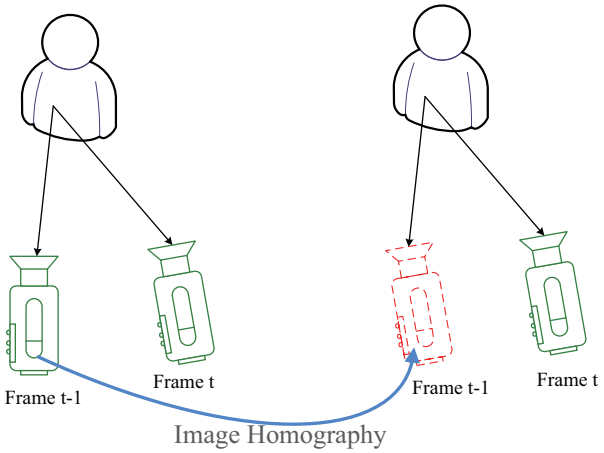

Fig. 3. Proposed image homography based reference view construction algorithm

Image homography is a non-singular linear relationship between points in two images captured by perspective cameras sharing the same optical center. The points are related by [9]:

$$x' = Hx \tag{1}$$

where x and x' are the corresponding points (in homogeneous coordinates) in the first and second view respectively. H is a 3x3 homography matrix computed by:

$$H = K'R'R^{-1}K^{-1} \tag{2}$$

where K and K' are the intrinsic matrices of the first and second camera respectively. And $R'R^{-1}$ represents the relative rotations of the two cameras.

As analyzed above, the camera motions between the adjacent views can be decomposed into three parts. And the motion estimation is most sensitive to the camera rotation. Therefore, in our proposed algorithm, we rectify the camera rotation by image homography to make the reference view be parallel to the current view. As shown in Fig. 3, the current view is frame t while the original reference view is frame t-1. Image homography is applied to the original reference view to make it be parallel to the current view. One advantage of the algorithm is that, after the rotation, there are only translations between the stereo views which is more motion estimation friendly. Much less bits are required to compressed the residues. And another advantage is that, depth maps are not required during the homography. Only the relative rotation between the stereo views and the camera intrinsic parameters are required. In our algorithm, the rotation and intrinsic matrices are intercepted from the game engines. The relative rotation contains three degrees of freedoms so only three float numbers are transmitted. The intrinsic parameters usually keep the same for the whole sequences and are only required once.

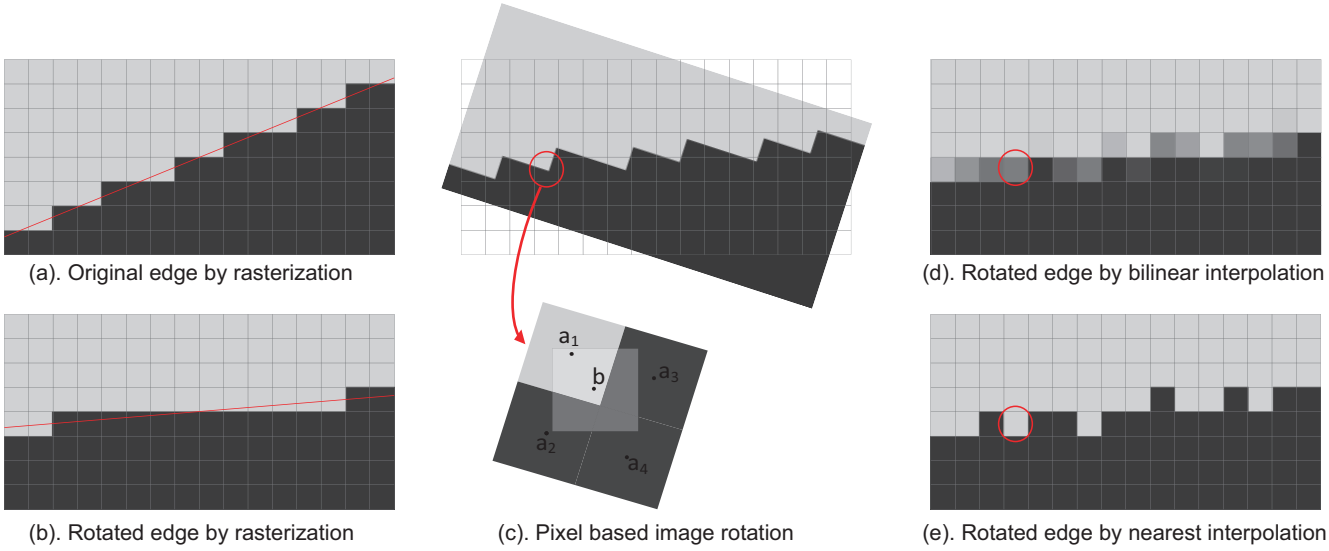

Fig. 4. Difference between rasterization and image interpolation

We apply image homography adaptively based on the relative rotations between adjacent views. The SAD of the rotation matrices between adjacent views is computed and compared with a pre-defined threshold. If the SAD is larger than the threshold, the proposed algorithm is performed to construct a novel reference frame. Otherwise typical video coding algorithms are utilized.

2.2. Edge Preserved Interpolation

One difference between conventional video sequences and game sequences is that, game sequences usually contain sharp edges due to the rasterization rules in the game engines while the conventional video sequences usually have transition regions in the edges of the adjacent objects. In the game engines, the rasterization rules define how vector data is mapped into the integer location of the pixels. Each pixel in the image is a tiny block which covers a small portion of the graph. When a pixel comes into the border of the objects, the rasterizer in most of the common used game engines would assign one of the objects' color intensity as the pixel value according to the pre-defined rules. This procedure is different from the real camera capture system in which the pixel value in the border is a weighted average of the each object's color intensity.

When image homography is performed to the game scenes, the edges will be interpolated in the new image. The conventional image interpolation algorithms cannot preserve the sharp and accurate edge. Fig. 4 is an example shows the difference between the rasterization and image interpolation when the image is rotated. There is an edge in Fig. 4(a) indicated by a line in the graph, where the regions above the line belong to a bright object and the regions below belong to another object with dark color. The graph is rasterized into an image according to the relative location of the pixel center and the line: when the pixel center is above the line, bright pixel value is assigned; when the pixel center is below the line, dark pixel value is assigned. Without loss of generality, suppose the graph is rotated into the location shown in (b). A new frame will be rasterized by the game engine according to the new location of the edge. However, the video steaming based approach does not have the exact location of the edge. Pixel interpolation algorithms are performed to generate the image after rotations. Fig. 4(c) shows the image interpolation procedure. The original image is first rotated. Afterwards, each pixel is re-sampled and interpolated. For pixel b in the new image, it is corresponding to four pixels a_1, a_2, a_3 and a_4 in the original image. If nearest interpolation algorithm is performed, pixel value of b is equal to the value of a_1 because the center of pixel b is inside of pixel a_1. When the conventional interpolation algorithm such as bilinear interpolation is performed, the pixel value of b is computed by the weighted average of neighbouring pixels with the weighting assigned by the relative distance from b to the neighbouring pixels. Other interpolation algorithms such cubic spline or truncated sinc algorithms have similar performance as bilinear interpolation. In our algorithm, we utilize bilinear algorithm for the ease of simplicity. The bilinear and nearest interpolation results are shown in Fig. 4(d) and (e) respectively. As observed in the figure, both of the interpolation results are quite different with the ground truth result shown in (b). In order to generate a sharp and accurate edge, we proposed the edge preserved interpolation (EPI) algorithm.

In the proposed EPI, we first detect the edge in the original image by edge detection algorithms. In our algorithm, Roberts cross [10] is adopted to detect the edges because of its low complexity and high accuracy in the noiseless game scenes. For the pixels not in the edge region, conventional

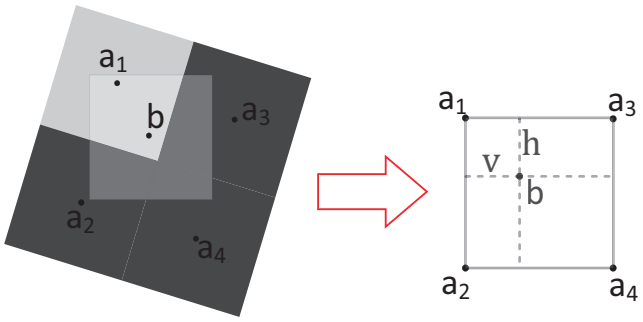

Fig. 5. Proposed edge preserved interpolation algorithm

bilinear interpolation algorithm is applied. For the pixels in the edge regions, we first classify the pixels into two objects in each 2x2 block:

$$\begin{cases} a_i \in C_1, & if\ I(a) \geq \bar{I} \\ a_i \in C_2, & if\ I(a) < \bar{I} \end{cases} \quad (3)$$

$$\text{where} \quad \bar{I} = \frac{1}{4}[I(a_1) + I(a_2) + I(a_3) + I(a_4)]$$

$I(a_i)$ is the pixel value of pixel a_i. By compare with the mean, the four pixels are classified into two sets: C_1 and C_2. In the example shown in Fig. 5, pixel a_1 is classified into C_1 while a_2, a_3 and a_4 belong to C_2. We calculate the probability that pixel b belongs to each class by:

$$p(b \in C_k) = \sum_{a_i \in C_k} p(b = a_i), \text{where} \quad k = 1, 2 \quad (4)$$

Here $p(b = a_i)$ indicates the probability that pixel p belongs to the pixel a_i and is defined as:

$$p(b = a_i) = f(D(b, a_i)), \text{where} \quad i = 1, 2, 3, 4 \quad (5)$$

$D(b, a_i)$ is the distance from pixel b to pixel a_i. And function f is a monotonically decreasing function ranging from 0 to 1. If pixel b is close to pixel a_i, then it will have a high probability to belong to a_i. However, it is complicated to calculate $D(b, a_i)$ and then derive the probability. In the implementation, we simplify Eqn. (5) to save the complexity by:

$$p(b = a_1) = p(b \in top) \cdot p(b \in left)$$
$$\text{where} \begin{cases} p(b \in top) = 1 - h \\ p(b \in left) = 1 - v \end{cases} \quad (6)$$

Here h and v are the horizontal and vertical distance from the left border and up border of the block respectively, as shown in Fig. 5. When b is close to a_1, h and v are small, resulting in a large probability. The probability of other three pixels are computed as:

$$p(b = a_2) = h \cdot (1 - v) \quad (7)$$
$$p(b = a_3) = (1 - h) \cdot v \quad (8)$$
$$p(b = a_4) = h \cdot v \quad (9)$$

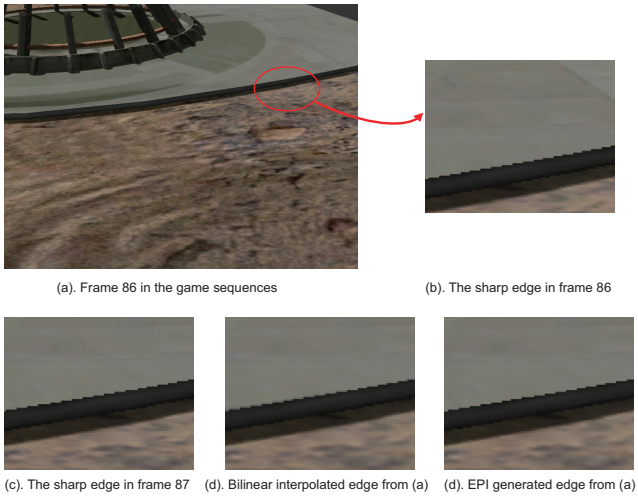

Fig. 6. Comparison of EPI and bilinear interpolation in the test sequence 'Power Plant'

The pixel value of b is computed as the mean value of the class with larger probability in Eqn. (4):

$$I(b) = \frac{1}{N_{\hat{k}}} \sum_{i \in C_{\hat{k}}} I(i) \quad (10)$$
$$\text{where} \quad \hat{k} = \arg\max_k p(b \in C_k)$$

Here N_k is the pixel number in class C_k.

3. EXPERIMENTAL RESULTS

We implement our algorithm based on the platform of x264. The encoder is set in low delay configuration because bi-directional (B) frames will lead to dependency on future frames and result in additional latency. The experiments are conducted to two games named 'Power Plant' and 'Column Scene', which are open source games based on Direct3D. We execute the games and intercept 200 consecutive frames with resolution 800x600. The proposed algorithm is simulated and compared with original x264. To show the advantage of the proposed EPI, we also compare with the algorithm which performs image homography but utilizes bilinear interpolation instead of the proposed EPI.

Fig. 6 compares the bilinear interpolation and the proposed EPI when image homography is performed to generate the new reference frame. Fig. 6 (a) is frame 86 of 'Power Plant' with the sharp edge shown in (b). And (c) is the edge in frame 87, while (d) and (e) are the homography results generated to predict (c) by bilinear interpolation and the proposed EPI respectively. As shown in the figures, bilinear interpolation blurs the edge and would produce large residues, while the edge derived by EPI is sharp and much more similar to (c).

Fig. 7 and Fig. 8 show the RD performance comparisons of the three algorithms on 'Power Plant' and 'Column Scenes'

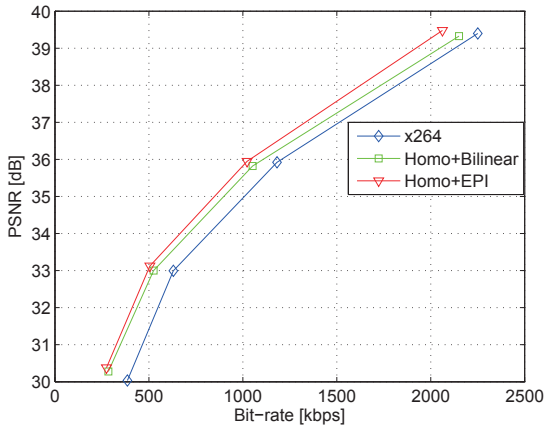

Fig. 7. RD performance comparison for 'Power Plant'

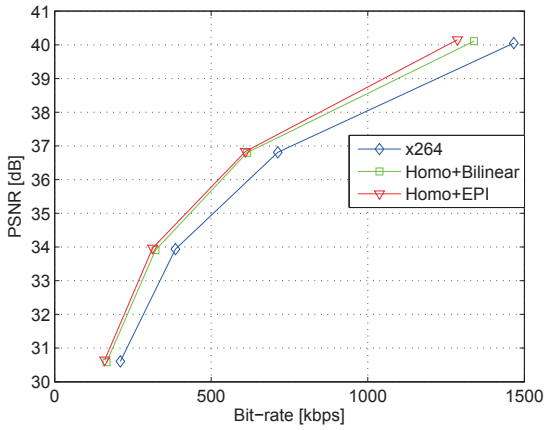

Fig. 8. RD performance comparison for 'Column Scenes'

respectively. The red curves are computed by our proposed image homography with EPI and the green curves are generated by image homography with bilinear interpolation. The results calculated from x264 are shown by the blue curves. Notice that we only show the bit rate of the frames whose SAD of the the rotation matrices between adjacent views is larger than the threshold. For the other frames that do not have relative rotation, just x264 is performed, which leads to the same bit rates.

As shown in the figures, the compression performance of the proposed algorithms with image homography have significant bit rate reduction comparing with the original x264 algorithm. This shows the effectiveness of the image homography. Meanwhile, proposed EPI based interpolation further reduces the bit rate, which corroborates our analysis above.

Numerically speaking, our method has 18.0% and 3.9% BD-rate reduction respectively compared with x264 and the algorithm without EPI. This result demonstrates the effectiveness of the two innovations proposed in our method: when only image homography is performed, 14.1% BD-rate reduction is achieved; when EPI is applied instead of the bilinear method, additional 3.9% BD-rate reduction is derived. As explained in Section 2, there are strong fluctuations and bursts when strong rotation happens. Our proposed algorithms reduce the bursts of the rate, resulting in significant transmission latency reduction.

4. CONCLUSION

In this paper, we propose a novel video streaming based cloud gaming algorithm. Image homography is introduced in our algorithm to rectify the reference frame for better motion estimation. Meanwhile, we also propose edge preserved interpolation for better edge prediction and further reduce the bit rate. Experimental results have verified the superiority of the proposed method with significant compression performance improvement.

5. REFERENCES

[1] OnLive, http://www.onlive.com/.

[2] Huang C Y, Hsu C H, and Chang Y C, "Gaminganywhere: an open cloud gaming system," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2014.

[3] I. Nave, H. David, A. Shani, and Y. Tzruya, "Games@ large graphics streaming architecture," in *IEEE International Symposium on Consumer Electronics*, 2008.

[4] Jurgelionis A, Fechteler P, and Eisert P, "Platform for distributed 3d gaming," in *International Journal of Computer Games Technology*, 2009.

[5] Giesen, Fabian, Ruwen Schnabel, and Reinhard Klein, "Augmented compression for server-side rendering," in *VMV*, 2008.

[6] S. Shi, C. H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *In Proceedings of the 19th ACM international conference on Multimedia*, 2011.

[7] A. Laikari, P. Fechteler, and B. Prestele, "Accelerated video streaming for gaming architecture," in *In 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2010.

[8] Fechteler, Philipp, and Peter Eisert, "Depth map enhanced macroblock partitioning for h. 264 video coding of computer graphics content," in *IEEE International Conference on Image Processing (ICIP)*, 2009.

[9] Hartley, Richard, and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge. Vol.2: 202-205, 2000.

[10] LS. Davis, "A survey of edge detection techniques," in *Computer Graphics and Image Processing*, 1975.