MULTIVIEW VIDEO COMPRESSION AND STREAMING BASED ON PREDICTED VIEWER POSITION

Dinei Florêncio and Cha Zhang

Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{*dinei,chazhang*}*@microsoft.com*

ABSTRACT

Recent technological advances have made possible a number of new applications in the area of 3D video. One of the enabling technologies for many of these 3D applications is multiview video coding, which has received significant attention in the last several years. However, the fundamental need of multiview coding for applications like immersive tele-conferencing has not been addressed. In this paper we define the boundaries of the problem, and show how a simple algorithm can yield gains of up to 2X reduction in bitrate with similar PSNR in the synthesized view. Our algorithm is based on using an estimate of the viewer position to compute the expected contribution of each pixel to the synthesized view, and encoding each macroblock of each camera views with quality proportional to the likelihood that the pixel will be used in the synthetic image.

***Index Terms*—** Multiview compression, streaming, view-dependent, immersive tele-conferencing

1. INTRODUCTION

Advances in camera, display and networking technology have enabled a new set of applications for three dimensional (3D) scene communication, among which are 3D TV/free viewpoint TV (FTV) [1], tele-immersive environment [2], immersive tele-conferencing [3], etc. In these applications, multiple video cameras are often used to simultaneously acquire the visual scene from different viewpoints. These videos are then transmitted to the remote end for rendering, providing the user an immersive experience.

Due to the high raw data rate of multiview video, multiview video compression is regarded as an essential piece of technology to enable 3D communication, and has attracted a lot of attention recently [4]. In particular, researchers have studied extensively on predictive coding for multiview video compression, since there is apparently huge redundancy across the videos from different viewpoints. Work has also been done to impose additional constraints for compression, such as delay constraints and random accessibility [5].

Since 3D TV is one of the most dominant driving forces of multiview video, most multiview video compression schemes assumes a two-stage process – an offline stage for compression and an online stage for streaming. Namely, the videos are first compressed with advanced predictive coding schemes and stored. When broadcasting to the remote end, all streams are sent across the network for decoding and rendering. In the 3D TV scenario, the data are transmitted through multicast channels, which can be very efficient if there are thousands of remote viewers.

The need to generate images from an arbitrary point may arise for two reasons: viewpoint selection, and parallax simulation. In viewpoint selection, the user is allowed to select a desired point of view, and video is generated/presented from that viewpoint. In this case (unless we are using a broadcast medium) the server may simply generate the desired view, then encode it and send that to the user. In other words, we either need current multiview coding (to encode all images and broadcast), or we can send just one/two, and traditional video coding or stereo coding is enough.

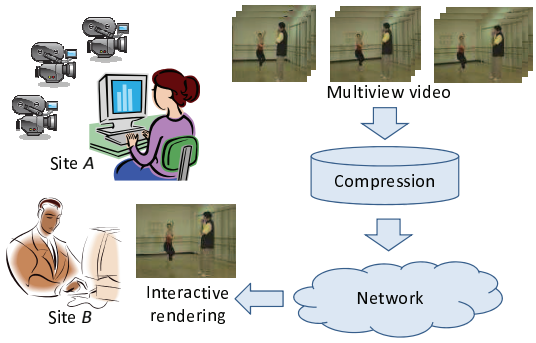

Fig. 1. Immersive tele-conferencing scenario. The system could be symmetrical, i.e. site *B* could also sent its multiview video to site *A*.

A main claim in our paper is that none of the current coding solutions addresses a very important scenario: providing the sensation of parallax in a one-to-one (or one-to-few) scenario. In that case, the round trip delay between the user and the server is too high to provide adequate parallax feedback, and traditional multiview coding is a significant waste of bandwidth.

A typical application is immersive tele-conferencing. Without loss of generality, let us consider a two-party conferencing session between site *A* and site *B* (Fig. 1). At site *A* a set of cameras is used to capture the scene. The videos are compressed with predictive coding, and sent to site *B*. The videos are then decompressed and rendered to the user in an interactive fashion, where the user may change their virtual viewpoint from time to time. Such a scenario, while simple, presents a number of challenges. For instance, the coding efficiency must be high, and the amount of data transmitted to the remote site must be kept low. To support interactivity , the videos must be compressed and sent to the remote site with little delay. To support parallax, it is imperative that the view be generated at the receiver side.

In immersive tele-conferencing, the traditional approach described above for 3D TV is not efficient any more. The remote user may only need to synthesize one or two images from the virtual viewpoints. It is very likely that only two or three nearby captured videos will be necessary to perform the rendering, and all the other

video streams that are encoded and transmitted become useless.

An alternative solution that has the best possible coding and transmission efficiency is to render the virtual view at site *A* and send only the compressed rendered image to site *B*. This requires site *B* to send its current virtual viewpoint and look direction to site *A* in an instantaneous fashion. Unfortunately, any network can have delays, which will introduce a lag between the time the user changes his/her viewpoint and the time the rendered image makes the change. This is not acceptable in interactive applications.

In interactive static scene browsing and streaming, researchers have proposed additional random access requirements on multiview compression [6]. Namely, by limiting the inter-frame dependencies between neighboring images, the decoding of any image block can be successful if a small set of other dependent image blocks are decoded. Index tables can be built to ensure that the compressed stream of image blocks is randomly accessible. Only the image blocks that are needed to render the virtual view is requested from the remote site during interactive browsing. Unfortunately, although the extension of the above approach to multiview video is straightforward, such a scheme is questionable. First, there is a delay between the stream request and the arrival of packages. Second, in multiview video compression, limiting the predictive coding for lower inter-block dependency can significantly reduce the coding efficiency. Third, since the scene is constantly changing, local caching techniques as in [6] may not be very effective. If a predicted block needs to trace back 3-4 other predicted blocks and finally a few intra-coded blocks in order to be decoded, it may be much more efficient to simply intra-code that image block and sent to the user.

In this paper, we propose a novel multiview compression and streaming scheme for immersive tele-conferencing (or other single-user view including parallax effects). Take the scenario in Fig. 1 as an example. The user at site *B* first sends his/her current viewpoint/look direction and motion information to site *A*. Site *A* renders the view (one or multiple) and generates weight maps for all the video frames. The weight maps indicates the quality requirement of each pixel or macroblock. If the weight is high, the pixel is used for rendering and needs to be encoded at high quality. Otherwise, the pixel can be heavily compressed. Site *A* then adaptively encode the multiview video based on these weight maps, and stream the video to site *B* for regular decompression and rendering. Delay effects can be alleviated by motion prediction and weight map smoothing. We show that such a scheme can dramatically improve the coding and streaming efficiency compared with traditional approaches.

The rest of the paper is organized as follows. Multiview video rendering and weight map generation are discussed in Sec. 2. Adaptive multiview video compression with the weight maps is described in Sec. 3. Experimental results are given in Sec. 4. Conclusions and future work are presented in Sec. 5.

2. MULTIVIEW VIDEO RENDERING

Multiview video rendering belongs to the broad research field of image-based rendering [7], and has been studied extensively in the literature. In this paper, we focus on one particular form of multiview video – multi-stereoscopic video with depth maps. We assume we are given a number of video sequences captured from different viewpoints. Meanwhile, a *single* depth map is available to facilitate the virtual view rendering. An example data set is shown in Fig. 2 [8]. It is worth pointing out that although our compression and streaming algorithm is introduced on this data format, the method can be applicable to other multiview video formats.

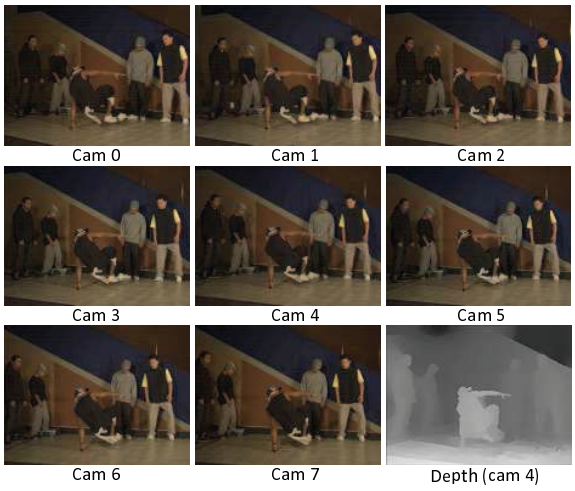

Fig. 2. An example multiview data set.

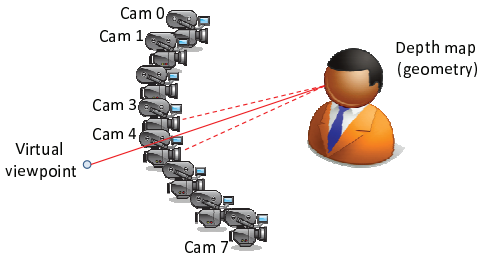

Fig. 3. The rendering process from multiview video.

In the following, we briefly describe the process of rendering an image from a virtual viewpoint given the image set and the depth map. As shown in Fig. 3, given a virtual viewpoint, we first split the to be rendered view into light rays. For each light ray, we trace the light ray to the surface of the depth map, obtain the intersection, and reproject the intersection into nearby cameras (cam 3 and 4 as shown in Fig. 3). The intensity of the light ray is thus the weighted average of the projected light rays in Cam 3 and Cam 4. The weight can be determined by many factors [9]. In the simplest form, we can use the angular difference between the light ray to be rendered and the light ray being projected, assuming the capturing cameras are at roughly the same distance to the scene objects.

Care must be taken to perform such rendering, as when the virtual viewpoint moves away from Cam 4 (where the depth map is given), there will be occlusions and holes when computing the light ray/geometry intersection. In our algorithm, we first convert the given depth map into a 3D mesh surface, where each vertex corresponds to one pixel in the depth map. The mesh surface is then projected to the capturing cameras to compute any potential occlusions in the captured images. Finally, the mesh is projected to the virtual rendering point with multi-texture blending, similar to that in [9]. For each vertex being rendered, it is projected to the nearby captured images to locate the corresponding texture coordinate. This process takes into consideration the occlusion computed earlier. That is, if a vertex is occluded in a nearby view, its weight for that camera will be set to zero.

Fig. 4 shows one of the images rendered from a virtual view point nearby Cam 2 (slightly rotated view direction). The weight maps clearly demonstrate whether a pixel in a captured video frame will be useful for rendering this particular virtual view. In Fig. 4,

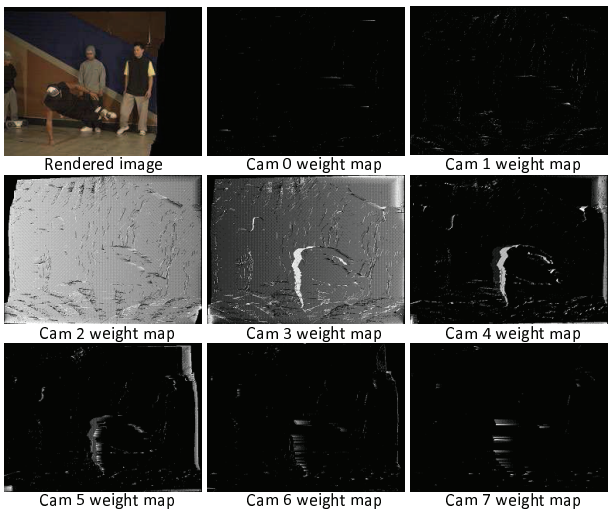

Fig. 4. The weight maps generated by the rendering process.

brighter pixels are the ones with larger weights. Note even for Cam 7, which is the farthest from the virtual viewpoint, there are still pixels being used due to occlusions. Naturally, during compression of the multiview video, the pixels with high weights shall be encoded with high quality, while the rest pixels can be encoded with low quality. Such an adaptive compression scheme is the key to our algorithm, and will be explained in detail in Sec. 3.

The weight maps in Fig. 4 are computed from a single virtual viewpoint. If there are multiple remote participants, we can compute such weight maps individually for each participant and average them to obtain a combined weight map. A similar scheme can be used to combat the delay issue that may be introduced by network transmission. For instance, assume there is a time τ delay between the two conferencing sites. When site *A* receives the virtual viewpoint request from site *B* (sent at time t), it has already been $t+\tau$. However, if site *A* also sends the user's motion information, it is possible to estimate the user's viewpoint at time $t+\tau$ by the user's moving speed to be within a certain range. With that information, multiple virtual rendering within the estimated range can be conducted to compute a combined weight map for compression. In addition, if the user's viewpoint does not change significantly, we may achieve a similar effect by simply smoothing the computed weight maps. As shown in Sec. 3, during adaptive multiview video compression, the weight map will be converted into a coarser one for macroblock based encoding, which effectively smoothes the weight map too.

3. ADAPTIVE MULTIVIEW VIDEO COMPRESSION

The objective is to maximize the quality of the synthesized image corresponding to the virtual viewpoint. (The synthesized image is, of course, based on the camera images, but not all pixels are equally used.) The point we exploit is to encode with higher fidelity portions of the images that are more likely to be used by the receiver to synthesize the final view. For that purpose, we use a modified version of the H.264 codec which allows us to specify the quantization parameter QP for each macroblock.

Deriving the optimum strategy for assigning the QPs requires detailed knowledge of the rate-distortion curve for the given images. Instead, we compute the quantization parameter for each macroblock QP_{mb} by using a $BaseQp$ and a simple ad-hoc mapping:

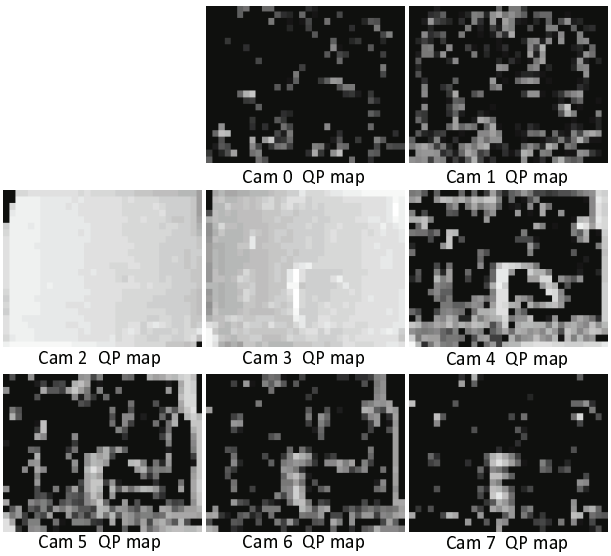

Fig. 5. QP values for weight maps of Fig. 4.

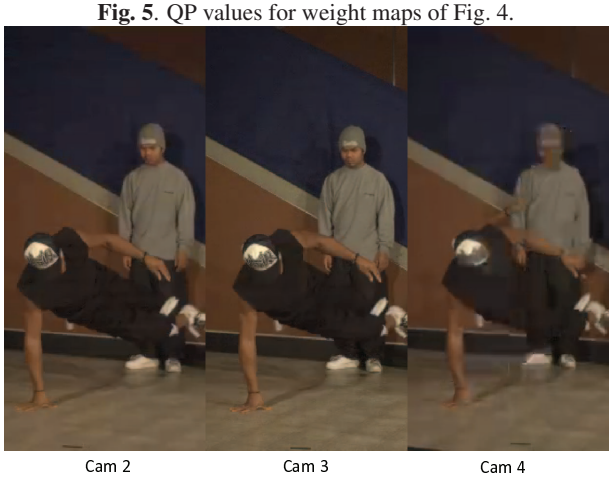

Fig. 6. A segment of encoded images from cameras 2, 3, and 4.

$$QP_{mb} = BaseQp + 6log_2\sqrt{1/256\sum_{mb} w_i^2} \qquad (1)$$

where w_i is the predicted weight for each pixel in the macroblock. Note that we take into consideration the logarithmic scale of PQ in H.264, which doubles the quantization step for each increment of 6 in QP. Based on the above rule, we compute QP for each macroblock in each of the camera images, and run our modified H.264. Note that we also limit the QP to the maximum H.264 value of 51. Figure 5 illustrates the final QP values for the same frames as Figure 4. Figure 6 shows a segment of the corresponding encoded images for cameras 2, 3, and 4. Note how portions of the image that have low weights (and thus high QP) are much more coarsely quantized.

4. EXPERIMENTAL RESULTS

As it can be expected, the savings are significant. Knowing in advance which portions of the image are likely to be (or not) used to synthesize the view provides a significant advantage, as we can allocate bits in proportion to the likelihood that the pixels will be used

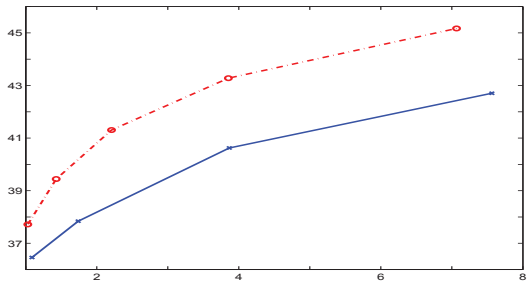

Fig. 7. PSNR for the synthesized image (in dB) as a function of bitrate (in Mbps) for H.264 (x) and proposed (o).

Fig. 8. Zoom of synthesized images: H.264 @ 3.4 Mbps (left), no coding (center) and proposed @ 1.7 Mbps (left).

in the synthetic view. Experimental results show savings of around half the rate for same PSNR in the synthesized image. Of course specific results will depend heavily on system elements, including the motion dynamics of the viewer, and the total transmission delay (the higher the delay, the less reliable the position prediction will be). Figure 7 shows the PSNR figures for the Break Dance sequence, using standard H.264 versus our codec (note that in both cases each camera is encoded independently). It can be seen that our proposed method achieves the same PSNR at around half the rate as H.264. Figure 8 shows the synthesized image from each algorithm.

Of course, another big question is what happens when the prediction of the viewer position is not 100% accurate. This is extremely important, after all, if the prediction was 100% accurate we could simply render the synthetic view at the encoder, and send only that view. We simulated this uncertainty by rendering the view from a view point deviating from the position used to compute the weight maps. Figure 9 shows the results, it can be that for a deviation of around 10 cm from the estimated position, the loss is about 1dB. At an uncertainty of 20 cm, the loss is about 2dB, erasing the gains obtained with the proposed scheme. Of course, even with a 200ms delay, we expect to be able to have a precision better than 10cm of the current position of the viewer's head.

5. CONCLUSIONS AND FUTURE WORK

We have presented a novel view-dependent multiview video compression and streaming framework for immersive tele-conferencing. The key idea is to let the remote participant send the capturing site its current viewpoint and motion information. The capturing site then renders the virtual view and obtains a set of weight maps for the

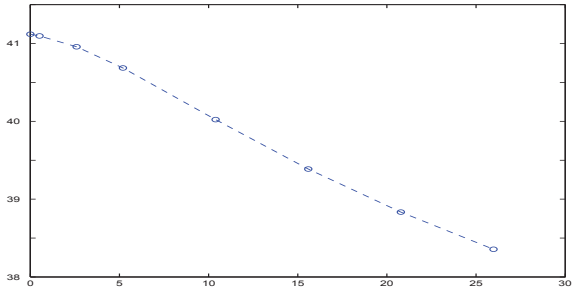

Fig. 9. PSNR of the reconstruction (in dB) as a function of standard deviation of predicted viewer position (in cm).

control of an adaptive compression scheme to maximize coding efficiency and minimize network bandwidth. We have shown that such a scheme dramatically outperforms the traditional non-adaptive coding schemes.

Although the proposed coding strategy does outperform the existing solution by around 2X, there is still a lot of space for improvement. With proper optimization and adequate design, it should be possible to reach achieve another 2X in rate reduction, possibly more. The main contribution of this paper is actually to point out a new research sub-area: Real-time, single user, parallax-based free viewpoint video coding. Of course, additional contributions are the initial ideas and algorithm presented, as well as the proposed metric to compare results.

Immersive tele-conferencing creates a new exciting research venue for multiview video processing, compression and streaming. While significant improvement has already been shown with our simple approach, there are many future research opportunities. Particularly associated with this paper would be extensions to explore interview-coding, joint user tracking and compression, rate distortion optimization of the overall codec, and channel coding for multiview compression. Other related research areas that will help in making an immersive tele-conferencing practical include immersive display technologies, effective user/eye tracking, and many others.

6. REFERENCES

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multi-view imaging and 3dtv," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, 2007.

[2] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy, "Viewcast: View dissemination and management for multi-party 3d tele-immersive environments," in *ACM Multimedia*, 2007.

[3] H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender, "The coliseum immersive teleconferencing system," Tech. Rep., HP Labs, 2002.

[4] M. Flierl and B. Girod, "Multiview video compression," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 66–76, 2007.

[5] A. Smolic and P. Kauff, "Interactive 3-d video representation and coding technologies," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 98–110, 2005.

[6] C. Zhang and J. Li, "Interactive browsing of 3D environment over the internet," in *Proc. SPIE VCIP*, 2001.

[7] C. Zhang and T. Chen, "A survey on image-based rendering – representation, sampling and compression," *EURASIP Signal Processing: Image Communication*, vol. 19, no. 1, pp. 1–28, 2004.

[8] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *ACM SIGGRAPH*, 2004.

[9] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," in *ACM SIGGRAPH*, 2001.