JOINT CODING OF MULTI-VIEW VIDEO AND CORRESPONDING DEPTH MAP

Sang-Tae Na, Kwan-Jung Oh, and Yo-Sung Ho

Gwangju Institute of Science and Technology (GIST)
1 Oryong-dong Buk-gu, 500-712, Gwangju, Korea

ABSTRACT

In this paper, we propose a joint coding scheme for both multi-view video and its corresponding depth map. After we synthesize a virtual image for the target view using adjacent view images and their depth information, we apply a view interpolation prediction (VIP) method for both multi-view video coding and its depth data coding. In order to improve the synthesized virtual view, we also propose a hole filling method that can compensate for empty regions caused by the 3D warping operation. With the proposed algorithm, we have obtained approximately 0.65 dB of the PSNR gain on average for the multi-view depth data, and 0.17 dB of the PSNR gain for the multi-view video data, compared to JMVM 1.0.

Index Terms— Multi-view video, Depth data, 3D warping, Hole filling, View interpolation prediction.

1. INTRODUCTION

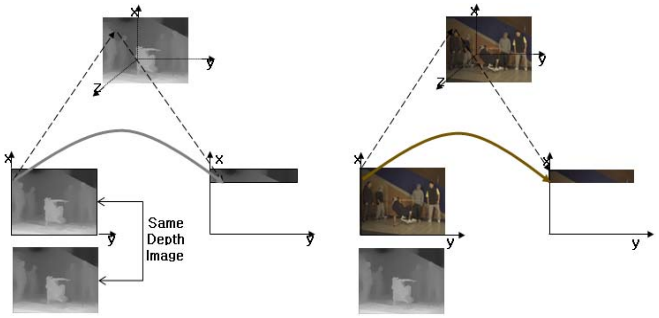
In most traditional multimedia systems, we can see images and videos at only one pre-determined viewpoint. However, multi-view video systems enable us to choose an arbitrary viewpoint and to feel more natural and photo-realistic three-dimensional (3D) effects. The multi-view video system can be applied to free-viewpoint television (FTV) and 3DTV. FTV enables users to choose an unrestricted viewpoint that is captured by multiple cameras or generated using depth data. Users can enjoy more realistic 3D scenes generated from conventional 2D video and its depth data [1]. In those systems, we need depth data as well as multi-view video. However, the more cameras are employed, the more coding bits for multi-view video and depth data are needed. Therefore, it is necessary to develop efficient coding methods to compress both multi-view video and its depth data.

Among various video compression schemes, the latest H.264/AVC outperforms previous coding standards, such as MPEG-2 and MPEG-4 Visual. In 2001, a new activity was launched in MPEG by Ad-Hoc Group (AHG) on 3D audio and visual (3DAV) to investigate additional video coding tools for multi-view video. In October 2004, they conducted a comparison test between the simulcast H.264/AVC coding method and a new coding method that allows inter-view prediction. It was reported that multi-view video coding tools gave better results than the simulcast video coding approach based on H.264/AVC [2]. Consequently, a joint multi-view video coding model (JMVM) based on the joint scalable video coding model (JSVM) [3] was proposed, and a reference software was also designed for multi-view video coding (MVC) in October 2006. Furthermore, in April 2007, a multi-view video-plus-depth (MVD) format was proposed to be investigated for future video systems, such as 3DTV and FTV [4].

Although independent coding of multi-view video and depth data has been studied for a while, joint coding of both multi-view video and its depth data is a new approach. In this paper, we propose a joint coding method of multi-view video and its corresponding depth data. After we apply the view interpolation prediction (VIP) scheme [5] to encode multi-view depth data using only available depth date itself, we apply the VIP scheme for multi-view video coding using multi-view video and reconstructed depth information.

2. CODING OF MULTI-VIEW DEPTH DATA

Depth data can be represented in a sampled video format where each sample value indicates the distance between the camera and each object point. One of the advantages of representing depth data in the video format is that we can apply video coding methods directly to compress depth data. In this paper, we treat the depth data as a video sequence. Since multi-view depth consists of multiple depth sequences of the same scene, there exists high correlation among neighboring depth data. Thus, we can develop a coding method based on JMVM for multi-view depth data.



(a) Depth image (b) Texture image

Figure 1. View Synthesis by 3D Warping.

Furthermore, we can borrow the idea of view synthesis prediction [6] to exploit the characteristics of depth data. We synthesize a virtual depth data for the current view using the 3D warping operation before we encode the actual depth data of the current view. Then, we use the synthesized depth data as an additional reference. In this special situation, the same depth data are used as depth information as well as a reference data for the 3D warping operation. Therefore, only camera parameters need to be sent as side information for depth data coding. However, as shown in Fig. 1, we need a reference texture image and its depth information in addition to camera parameters for texture image coding.

2.1. 3D warping operation

We apply a 3D warping operation to the synthesized views. A perspective projection matrix for 3D warping can be represented by

$$PM = A\left[R \mid t\right]. \tag{1}$$

where A, R, and t are camera parameters and denote the intrinsic matrix, rotation matrix, and translation vector, respectively. Eq. (2) is the projection equation which consists of depth data and Eq. (1). Eq. (2) can be transformed into Eq. (3). By Eq. (3), we can project pixel positions from the image coordinate to the 3D world coordinate.

$$P_{ref}(x, y, 1) \cdot D = A\left[R \mid t\right] \cdot \widetilde{P}_{WC}(x, y, z, 1) \tag{2}$$

$$P_{WC}(x, y, z) = R^{-1} \cdot A^{-1} \cdot P_{ref}(x, y, 1) \cdot D - R^{-1} \cdot t \tag{3}$$

where D denotes the depth information, P is a homogenous coordinate in the reference image coordinate system or the pixel position in the 3D world coordinate, and $\widetilde{P}$ indicates a homogenous coordinate in the 3D world coordinate system. After the projection, the pixel positions in the 3D world coordinate are mapped to positions in a desired target image by Eq. (4), which is the inverse form of Eq. (3).

$$P_{target}(x, y, 1) = A \cdot R \cdot (P_{WC}(x, y, z) + R^{-1} \cdot t) \tag{4}$$

Finally, we can find the corresponding pixel positions in the target image to the pixel positions in the reference image. We then obtain the synthesized image by sharing the pixel values between the pixel positions in the reference image and in the target image.

2.2. Hole filling and compensating

Figure 2 illustrates the 3D warping process at a pixel level. As shown in Fig. 2, during the 3D warping operation, none or more than one pixel positions in the reference view can be projected into one pixel position in the target view due to occlusion and disocclusion situations. Occlusion regions are defined as areas that exist at the reference view, but cannot be visible at the target view. On the other hand, disocclusion

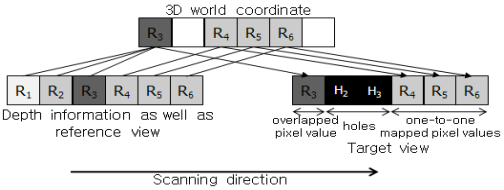


Figure 2. Hole Generation during 3D Warping.

regions are the areas that cannot be seen at the reference view, but exist at the target view.

In occlusion regions, since more than one pixel positions in the reference view are projected into one position in the target view, pixel values are overlapped and foreground regions can be mixed up with background regions. However, since we are dealing with depth data, we can easily resolve the problem. Among the pixel values in the overlapped pixel position, the largest value represents the most foreground object. Therefore, we can solve the occlusion problem by choosing the largest value among all candidate values.

In disocclusion regions of the target view, none of pixel position is projected. Thus, those regions are remained as initial values and indicated as empty holes. In order to fill those holes, we mainly divide the problem into two cases: (1) the case when there is only one directional reference view, such as P-view, and (2) the case when both directional reference views are available, such as B-view.

For the case of P-view, we should fill the holes from the synthesized view itself because there is no other available information. In this case, most holes are generated in the regions that are hidden by foreground regions in the reference view, but are visible in the target view. Thus, we can assume that they are mostly in the background regions. We scan the synthesized view to distinguish background regions from foreground regions. If depth values on the left side of the holes are larger than the values on the right side, holes are on the right side of foreground; otherwise, holes are on the left side of foreground.

Since the depth data in the video format does not have any texture information, but it contains only simple distance information, depth values are very similar to one another. Thus, we can fill the holes effectively using depth values from background regions, and again this process is quite well-defined when we are dealing with depth data.

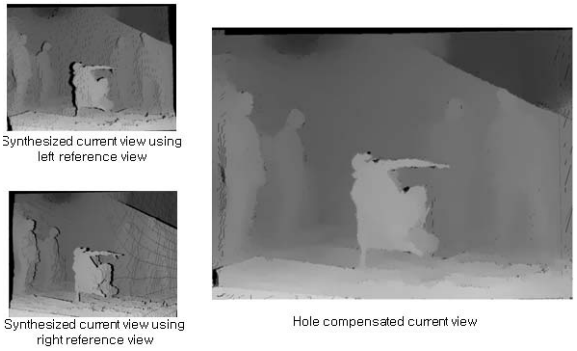


Figure 3. Hole Compensation in B-view.

For the case of B-view, the hole filling process is more obvious. When the reference view is located on the left, holes are generated on the left side of foreground regions in the synthesized view; otherwise, they are on the right side. Thus, as shown in Fig. 3, we can compensate the holes using the correct information from both synthesized views.

Furthermore, relatively small holes can be generated due to inaccurate camera parameters and depth data. For those holes, we can apply simple median filtering. Besides, holes existing around edge regions are filled by adjacent pixel values that are the nearest nonzero values from the holes.

2.3. Encoding process

After the hole filling process, we obtain a synthesized depth image. When we encode P- or B-view, we can consider the synthesized image as additional information and apply the VIP coding method [5]. Figure 4 shows a coding procedure.

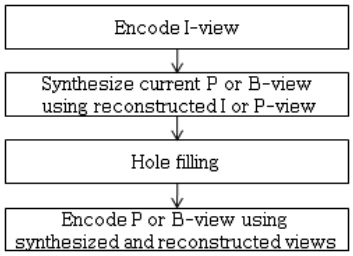


Figure 4. Encoding Process.

3. MULTI-VIEW VIDEO DATA CODING USING RECONSTRUCTED DEPTH DATA

For multi-view texture video coding, we can apply a similar coding method employed in multi-view depth video coding, provided that multi-view depth data are available. In order to synthesize a virtual texture image for the current view, we also apply the 3D warping operation that was explained before. During the view synthesis process, holes can be generated because of the same reasons as discussed before.

For multi-view texture images, we apply similar hole filling and compensation methods with multi-view depth data. However, when only one reference image is available, we use the corresponding depth data in the hole filling process. Figure 5 illustrates the synthesized image of the first frame from View-2, the corresponding depth image, the enlarged image of the dotted square area, and the overlapped image, clockwise. As shown in Fig. 5, we examine whether holes are generated in foreground or background regions using the corresponding depth data of the current view. Thus, we fill holes more precisely and results usually look better when background regions are planar. Furthermore, depth data are used to distinguish foreground regions from background regions in occlusion regions.

For multi-view texture video coding, we use the same coding procedure as discussed in Section 2.3.

Figure 5. Hole Filling using Depth Data, 'Breakdancers'.

4. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithm, we have implemented the proposed method based on JMVM 1.0 reference software [3]. For the test sequence, we have used 8 views and 46 frames of 'Breakdancers' and 'Ballet' video, and depth data sequences in XGA (1024 x 768) format provided by Microsoft Research [7].

After we encoded multi-view depth data, we utilized the reconstructed depth data for multi-view texture video coding with the same QP value. We selected two different search ranges (SR) to show that our proposed method is more effective when the search range is small because pixel values in the synthesized view are collocated with pixel values of the current view. Table 1 to Table 6 list PSNR vales and bit rates.

In this experiment, we did not include camera parameters in calculation of multi-view depth data coding. Moreover, coding bits for camera parameters and depth data are not considered for multi-view texture video data coding.

Table 1: Results for multi-view depth, SR ±96. (Breakdancers)

	JMVM 1.0		**Proposed**		**Gain**	
QP	***PSNR (dB)***	***Bitrate (kbps)***	***PNSR (dB)***	***Bitrate (kbps)***	***PSNR (dB)***	***Bitrate (%)***
22	46.68	586.60	46.99	539.71	+0.73	-13.66
27	43.34	320.02	43.67	295.72		
32	40.27	166.15	40.61	153.31		
37	37.36	87.93	37.77	82.83		

Table 2: Results for multi-view depth, SR ±96. (Ballet)

	JMVM 1.0		**Proposed**		**Gain**	
QP	***PSNR (dB)***	***Bitrate (kbps)***	***PNSR (dB)***	***Bitrate (kbps)***	***PSNR (dB)***	***Bitrate (%)***
22	47.28	521.60	47.55	494.02	+0.57	-9.47
27	43.98	312.91	44.33	299.63		
32	40.51	172.38	40.90	167.92		
37	37.39	92.55	37.80	90.76		

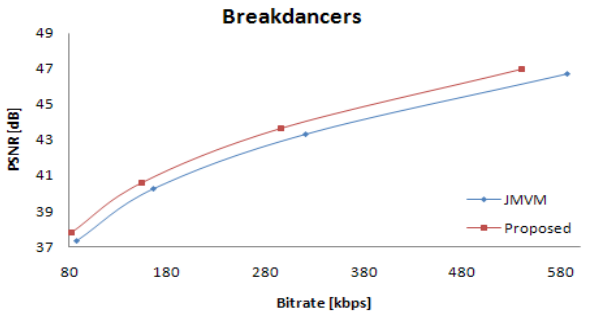


Figure 6. Rate distortion curves for multi-view depth, SR ±96.

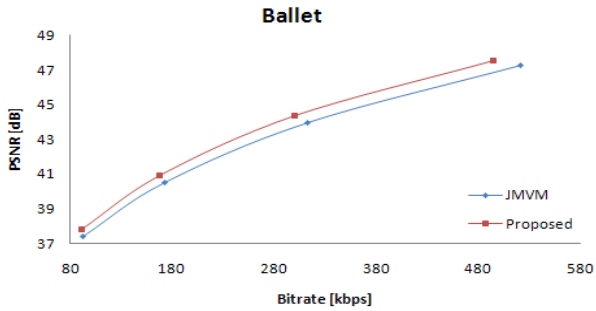


Figure 7. Rate distortion curves for multi-view depth, SR ±96.

Table 3: Results for multi-view video, SR ±96. (Breakdancers)

	JMVM 1.0		Proposed		Gain	
QP	*PSNR (dB)*	*Bitrate (kbps)*	*PNSR (dB)*	*Bitrate (kbps)*	*PSNR (dB)*	*Bitrate (%)*
22	40.01	979.29	40.00	942.00	+0.12	-4.75
27	38.78	419.74	38.75	401.01		
32	37.17	218.84	37.12	205.12		
37	35.08	127.63	35.07	116.39		

Table 4: Results for multi-view video data, SR ±96. (Ballet)

	JMVM 1.0		Proposed		Gain	
QP	*PSNR (dB)*	*Bitrate (kbps)*	*PNSR (dB)*	*Bitrate (kbps)*	*PSNR (dB)*	*Bitrate (%)*
22	42.22	403.91	42.20	386.34	+0.21	-6.26
27	40.91	203.36	40.89	192.86		
32	39.00	116.64	38.99	108.88		
37	36.64	71.61	36.73	65.58		

Table 5: Results for multi-view video, SR ±8. (Breakdancers)

	JMVM 1.0		Proposed		Gain	
QP	*PSNR (dB)*	*Bitrate (kbps)*	*PNSR (dB)*	*Bitrate (kbps)*	*PSNR (dB)*	*Bitrate (%)*
22	40.00	1038.22	39.98	991.61	+0.22	-7.79
27	38.75	456.57	38.70	423.34		
32	37.11	243.30	37.01	216.63		
37	34.94	143.17	34.90	122.19		

Table 6: Results for multi-view video, SR ±8. (Ballet)

	JMVM 1.0		Proposed		Gain	
QP	*PSNR (dB)*	*Bitrate (kbps)*	*PNSR (dB)*	*Bitrate (kbps)*	*PSNR (dB)*	*Bitrate (%)*
22	42.19	433.81	42.18	416.94	+0.28	-7.58
27	40.85	223.98	40.82	210.42		
32	38.88	129.74	38.85	118.56		
37	36.39	79.14	36.42	69.66		

From the above results, we note that coding efficiency varies according to the number of scenes that contain large holes and complexity of the texture in the video data.

5. CONCLUSIONS

In this paper, we have proposed an efficient coding method for both multi-view video and its depth data using virtual view synthesis. In order to synthesize a virtual image for the current view, we have employed the 3D warping operation. During the process of synthesizing a virtual image, holes are generated due to the occlusion and disocclusion situations. Those holes are filled by considering the characteristics of depth data. We have also applied the VIP coding scheme to encode both multi-view video and its depth data. In the VIP scheme, the synthesized image was used as an additional reference frame and new coding modes were added. By experiments, we have shown that the proposed scheme outperforms JMVM 1.0 by 0.17 dB and 0.65 dB in PSNR for the multi-view video and its depth data, respectively.

ACKNOWLEDGMENT

This work was supported in part by ITRC through RBRC at GIST (IITA-2008-C1090-0801-0017).

REFERENCES

[1] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand, "3D Video and Free Viewpoint Video – Technologies, Applications and MPEG Standards," Proc. of ICME 2006, pp. 2161-2164, July 2006.

[2] ISO/IEC JTC1/SC29/WG11 N6720: Call for Evidence on Multi-view Video Coding, October 2004.

[3] ISO/IEC MPEG & ITU-T VCEG JVT-U207: Joint Multiview Video Model (JMVM), October 2006.

[4] ISO/IEC MPEG & ITU-T VCEG JVT-W100: Multi-view Video plus Depth (MVD) Format for Advanced 3D Video Systems, April 2007.

[5] C. Lee, K.J. Oh, S. H. Kim, and Y.S. Ho, "An Efficient View Interpolation Scheme and Coding Method for Multi-view Video Coding," Proc. of IWSSIP, pp. 107-110, June 2007.

[6] S. Ince, E. Martinian, S. Yea, and A. Vetor, "Depth Estimation for View Synthesis in Multiview Video Coding," Proc. of IEEE 3DTV Conference, May 2007.

[7] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-qulity Video View Interpolation using a Layered Representation," Proc. of ACM SIGGRAPH, pp. 600-608, August 2004.