Coding of Layered Depth Images Representing Multiple Viewpoint Video

Seung-Uk Yoon[1], Eun-Kyung Lee[1], Sung-Yeol Kim[1], Yo-Sung Ho[1],
Kugjin Yun[2], Sukhee Cho[2], and Namho Hur[2]

[1]Department of Information and Communications
Gwangju Institute of Science and Technology (GIST)
1 Oryong-dong, Buk-gu, Gwangju, 500-712, Republic of Korea
{suyoon, eklee78, sykim75, hoyo}@gist.ac.kr

[2]Broadcasting System Research Group
Electronics and Telecommunications Research Institute (ETRI)
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, Republic of Korea
{kjyun, shee, namho}@etri.re.kr

Abstract. The multi-view video is a collection of multiple videos, capturing the same scene at different viewpoints. Since the data size of the multi-view video linearly increases as the number of cameras, it is necessary to develop an effective framework to compress, store, and transmit multi-view video data. Currently, most multi-view video coding algorithms are based on the H.264/AVC standard. However, we can exploit the three-dimensional depth information contained in multiple videos to efficiently encode multi-view video data. In this paper, we describe how to encode multi-view video data with depth information based on the framework using layered depth image (LDI).
Index Terms -MPEG, H.264/AVC, multi-view video coding (MVC), layered depth image (LDI)

1 Introduction

The multi-view video is a collection of multiple videos, capturing the same scene at different camera locations. If we acquire multi-view videos from multiple cameras, it is possible to generate scenes at arbitrary view positions. It means that users can change their viewpoints freely within the limited range of captured videos and can feel the visible depth with view interaction. The multi-view video can be used in a variety of applications including free viewpoint video (FVV), free viewpoint TV (FTV), three-dimensional TV (3DTV), surveillance, and home entertainment.

Although the multi-view video has much potential for a variety of applications, one big problem is a huge amount of data. In principle, the size of multi-view video data increases linearly as the number of cameras; therefore, we need to encode the multi-view video data for efficient storage and transmission. Hence, it has been perceived that multi-view video coding (MVC) is a key technology to realize those applications.

ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group (MPEG) has been recognized the importance of MVC technologies, and an ad hoc group (AHG) on 3-D audio and visual (3DAV) has been established since December 2001. Four main exploration experiments (EEs) on 3DAV were performed from 2002 to 2004: EE1 on omni-directional video, EE2 on FTV, EE3 on coding of stereoscopic video using multiple auxiliary components (MAC), and EE4 on depth/disparity coding for 3DTV and intermediate view interpolation. In response to the call for comments issued in October 2003, a number of companies have expressed their interests for a standard that enables FTV and 3DTV. After MPEG called interested parties to bring evidences on MVC technologies in October 2004 [1], some evidences were recognized in January 2005 and a call for proposals (CfP) on MVC has been issued in July 2005 [2]. Then, the responses to the call have been evaluated in January 2006 [3].

In this paper, we describe how to encode the multi-view video with depth information using the concept of the layered depth image (LDI) [4]. While most MVC techniques are based on the predictive coding structure, our framework [5] takes a completely different approach using LDI.

2 Algorithms and Tools for MVC

The major objective of multi-view videos is to provide free views or depth impression using captured videos at different viewpoints. In order to provide such a functionality, it is essential to compress a huge amount of multi-view video data. Recently, several algorithms have been proposed in response to the CfP on MVC [6] and have been evaluated at the 75th MPEG meeting in January 2006 [3].

Evaluation results of the responses to the CfP on MVC demonstrate that new MVC technologies provide significant gains over view independent coding using H.264/AVC as defined in the CfP, both in terms of better quality at the same rate and lower rate at the same quality. While all the submissions to the CfP were based on H.264/AVC, a number of different tools were proposed including hierarchical B pictures in both temporal and view dimensions, view interpolation prediction, illumination compensation, disparity vector prediction, asymmetric macroblock (MB) partitioning, etc [3].

The key idea of those MVC algorithms is to utilize spatial and temporal relationships in multiple videos. As we apply motion estimation along the temporal direction in a single video, we can perform disparity estimation among different views in the multi-view video. In other words, they try to exploit the spatio-temporal correlation among adjacent views. Those methods outperform the simulcast coding scheme that encodes each view independently using H.264/AVC. The following subsection describes various MVC tools briefly.

2.1 Hierarchical B Pictures in Both Temporal and View Prediction

In the current MPEG scalable video coding (SVC) scheme, temporal scalability is achieved by the motion compensated temporal filtering (MCTF) using prediction and update steps. The hierarchical B picture coding has the similar decomposition structure as the H.264/AVC-based MCTF layers without update steps. As shown in Fig. 1, the first picture is independently coded as IDR picture, and all remaining pictures are coded in "B...B I/P" group of pictures using the concept of hierarchical B pictures [7]. The coding order of the pictures in a GOP is "I I/P B^1 B^2 B^2 B^3 B^3 B^3 B^3..."

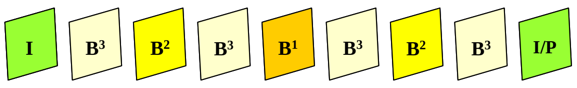

Fig. 1. The structure of hierarchical B pictures

The concept of hierarchical B pictures can be applied to inter-view prediction. In H.264/AVC, any group of picture (GOP) length can be selected and a maximum of four reference pictures can be used for each picture in general. These reference pictures are taken from temporal as well as inter-view direction, such that the encoder can choose the best reference. In order to exploit inter-view dependency, each second view uses reference pictures from neighboring views. For an even number of views, the last view has only references to one of the neighboring views. For memory optimization purposes, picture reordering before encoding and inverse reordering after decoding are applied [3].

2.2 View Interpolation Prediction

Three types of prediction schemes are mainly used in MVC: temporal, inter-view, and view interpolation prediction. The basic idea of the view interpolation prediction is that a set of one or more previously encoded/decoded frames are used to interpolate a new virtual frame that is used to predict the current frame to be coded. The interpolation process could exploit the camera parameters as well as a depth/disparity map to first compute the pixel locations in the reference frames corresponding to the pixel location to interpolate, and then computes a weighted average of these pixels to produce the interpolated pixel. One way is to generate the required disparity map at the decoder using previously received sequences combined with an optional depth correction values sent by the encoder. In the other way, the depth could be estimated at the encoder and encoded using H.264/AVC. Finally, each block of the frame to be coded may be predicted from the interpolated frame [3].

2.3 Illumination Compensation

In a practical scenario, multi-view video systems involving a large number of cameras might be built using heterogeneous cameras, or cameras that have not been perfectly calibrated. This leads to differences in luminance and chrominance when the same parts of a scene are viewed with different cameras [6]. Methods to compensate the illumination changes among cameras are include: (1) new syntaxes are introduced to convey the information used for illumination compensation, (2) macroblock/block level processing is used for illumination compensation, and (3) disparity estimation needs to be modified to take illumination compensation into account [3].

2.4 Disparity Vector Prediction

When camera arrangements are fixed with respect to each other, disparity relations are similar in short time intervals such as a single GOP. As motion vectors are used in the temporal direction, the reuse of disparity vectors among views is effective in multi-view cases. Decoded disparity vectors can be stored in a disparity vector memory, and the stored disparity vectors are used to predict disparity vectors for different pictures that belong to other views [3].

2.5 Asymmetric MB Partitioning

A boundary of objects typically exists in an asymmetrical position in a MB. Therefore, the symmetrical segmentation implemented in H.264/AVC does not always efficiently describe motion information. Although the adaptive block size technique as implemented in H.264/AVC is applied, many disparity/motion vectors are still necessary. The asymmetric MB partitioning (AMP) technique reduces disparity/motion vectors for a MB. The AMP technique efficiently represent motion information even in such a case by exploiting segmentation pattern information of a macroblock [3].

3 Coding of Multi-view Video with Depth Information using LDI

In order to efficiently compress, store, and transmit multi-view video data, we have proposed a framework utilizing the depth information based on the concept of LDI [5]. Unlike other algorithms based on H.264/AVC, our proposed framework is based on the conversion between multi-view videos and LDI frames. The generation of LDIs from natural multi-view video is described in "A Framework for Multi-view Video Coding using Layered Depth Images" [5]. In this section, we explain the encoding methods of the generated LDI frames.

3.1 Data Structure of LDI

LDI contains several attribute data together with multiple layers at each pixel location. A single LDI pixel, which is called layered depth pixel (LDP), contains different number of depth pixels (DPs). Each DP consists of color, depth, and auxiliary data that support reconstruction of multi-view images from the decoded LDI. Thus, multi-view video data can be stored within the data structure of LDI. The data structure of LDI is shown in Fig. 2 [4].

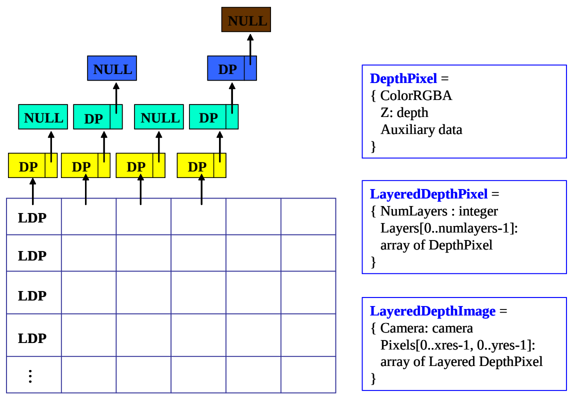

Fig. 2. Data structure of the constructed LDI

3.2 Characteristics of the Generated LDI

Before encoding the generated LDI frames, we first analyze the properties of the constructed LDIs. Three key characteristics of LDI are: (1) it contains multiple layers at each pixel location, (2) the distribution of pixels in the back layer is sparse, and (3) each pixel has multiple attribute values. Because of those special features, LDI enables us to render arbitrary views of the scene at new camera positions [5]. Table 1 lists the number of pixels included in each layer of the generated LDI. The LDI frame in Table 1 is constructed from the first eight color and depth frames of the "Breakdnacers" sequence with depth threshold 3.0. The basic assumption of our framework is that the depth information is available for the input multi-view video. Therefore, we focus on the "Breakdancers" sequence, which contains depth images for whole frames [8][9].

The maximum number of layer is determined by the number of cameras. As we can observe from the table, the pixel distribution becomes sparse as the layer number increases. It means that the data size of the original multi-view video could be reduced by converting them into the specific data structure based on LDI.

Table 1. Pixel distribution of the generated LDI for the 1st frames of "Breakdancers" (Depth threshold: 3.0)

Layer	Distribution [%]	Layer	Distribution [%]
1	100.0	5	46.8
2	90.4	6	34.7
3	69.8	7	29.0
4	47.0	8	19.1

3.3 Encoding of LDI Frames

After generating LDI frames from the natural multi-view video with depth, we separate each LDI frame into three components: color, depth, and the number of layers (NOL). Specifically, color and depth component consists of layer images, respectively. The maximum number of layer images is the same as the total number of views. In addition, residual data should be sent to the decoder in order to reconstruct multi-view images. Color and depth components are processed by data aggregation/layer filling to apply H.264/AVC. NOL could be considered as an image containing the number of layers at each pixel location. Since the NOL information is very important to restore or reconstruct multi-view images from the decoded LDI, it is encoded by using the H.264/AVC intra mode. Finally, the residual data, differences between the input multi-view video and reconstructed ones, are encoded by H.264/AVC. Figure 3 illustrates the encoder block diagram.

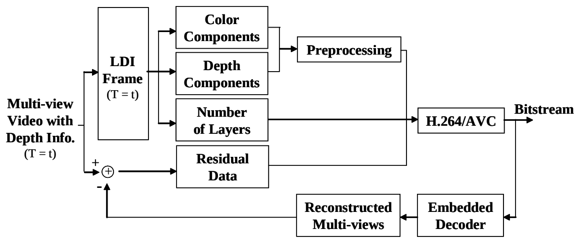

Fig. 3. Encoder block diagram

In this paper, we have experimented with two kinds of encoding algorithms for color and depth components of LDI. The first idea is that we need to aggregate scattered pixels into the horizontal or vertical direction [10][11] because each layer of the constructed LDI has different number of pixels. Moreover, there are lots of empty pixel locations in back layers. Although H.264/AVC is powerful to encode rectangular images, it does not support shape-adaptive encoding modes. Therefore, we aggregate each layer image and then fill the empty locations with the last pixel value of the aggregated image. First, the scattered pixels in each layer are pushed to the horizontal direction. Second, the locally aggregated images are merged into a single huge image and empty pixels are padded. For example, if each layer image has XVGA (1024 x 768) resolution, the single aggregated image becomes 8192 x 768. Finally, the generated one is divided into the images with pre-defined resolutions, e.g., 1024 x 768, to employ H.264/AVC.

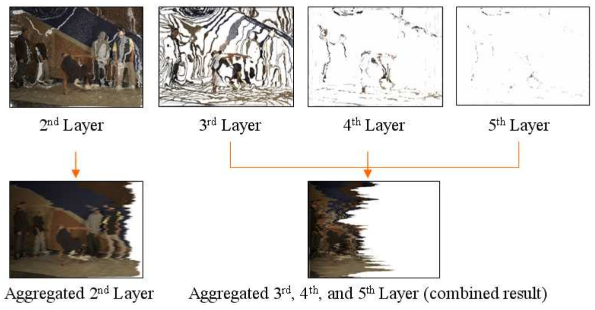

Fig. 4. Data aggregation with the horizontal direction

Figure 4 shows an example of the data aggregation with the horizontal direction. The maximum number of layers is eight, because there are eight different views in the "Breakdancers" sequence. One problem of the data aggregation is that the resultant images have severely different color distributions. It leads to poor coding efficiency because the prediction among aggregated images is difficult.

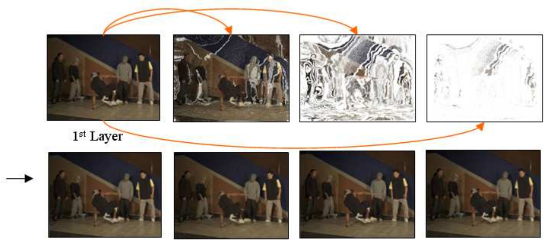

Fig. 5. Layer filling using the first layer information

The second method is called the layer filling, which is shown in Fig. 5. In order to solve the above problem, we can fill the empty pixel locations of all layer images using pixels in the first layer. Since the first layer has no empty pixels, we can use same pixels in the first layer to fill the other layers. This increases the prediction accuracy of H.264/AVC, therefore data size could be reduced further. We can eliminate the newly filled pixels in the decoding process because the information of NOL is sent to the decoder. It is an eight bit gray scale image that each pixel contains an unsigned integer number representing how many layers there are.

4 Experiments and Analysis

We have generated LDI frames from the "Breakdancers" sequence and encoded the constructed LDIs using the data aggregation or layer filling. The reconstruction procedure starting with the decoded LDI is as follows [12]. After moving whole pixels in the decoded LDI to world coordinates of the reference camera, for example, the central camera among eight cameras, we re-project those 3-D points into each camera location. In this step, the target of the inverse 3-D warping is each camera location. Consequently, a different number of pixels are located at each pixel location of each camera. Figure 6 shows results of inverse 3-D warping at several camera locations.

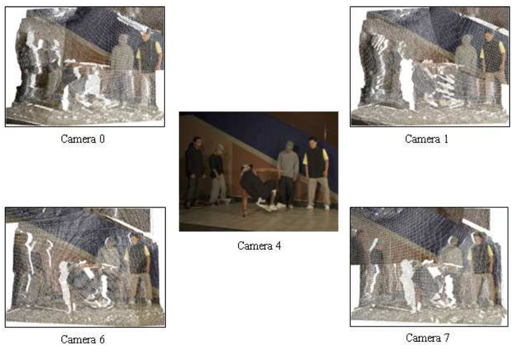

Fig. 6. Results of the inverse warping

There are several holes in the re-projected multi-view images because information is lost when pixels are moved to the reference camera location during the generation process of LDI. In order to fill those holes, we can use color and depth information contained in DPs of back layers. By using DPs moved from other camera locations, we can restore some empty pixels at the current camera view. The reconstruction results with back layer pixels is depicted in Fig. 7.

After restoring some empty pixels of each view, we can still see uncovered regions. The main reason is that several cameras cannot capture certain parts of the scene, for instance, the left-most side of the camera 0 and the right-most side of camera 7. Actually, the camera arrangement of MSR data is 1-D arc of eight cameras with 20cm spacing. Therefore, it is natural that we cannot find sufficient information to restore both left-most side and right-most side of the reconstructed images at certain camera

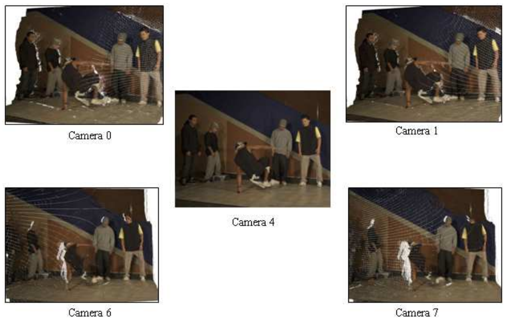

Fig. 7. Reconstruction with back layer pixels

locations. In this case, we need additional information to restore those regions. Although we can reduce the empty regions using interpolation methods, inaccurate pixel values could cause artifacts in the reconstruction results. Therefore, we need to fill the holes using the compensation module [5]. During the compensation process is performed, we have used the residual data extracted from the original multi-view images. The final reconstruction results with residual data is illustrated in Fig. 8.

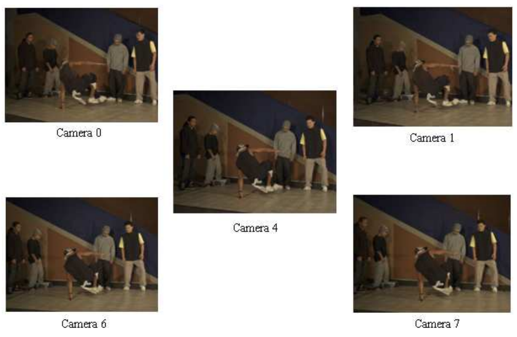

Fig. 8. Reconstruction with residual data

In Table 2, we have compared the data size between sum of frames of the test sequence and the generated LDI frame. In the table, sum of frames means the summation of eight color and depth images of the test sequence without encoding. Simulcast using H.264/AVC (color + depth) means the summation of data size calculated by the independent coding of color and depth images. Two kinds of encoding methods, one is data aggregation with the horizontal direction and the other is the layer filling, are used to encode the constructed LDI frames.

Table 2 shows the data size by changing the depth threshold value from 0.0 to 5.0, but the data

Table 2. Comparison of data size for the "Breakdnacers" sequence [kbytes]

	1st 8 Frames	2nd 8 Frames
Sum of frames	25,166	25,166
Simulcast (color+depth)	137.7	132.5
Simulcast (color only)	97.4	96.3
LDI frame (threshold=0.0)	24,520	24,644
Encoded LDI (Aggregation)	135.4	133.7
Encoded LDI (Layer filling)	71.4	72.9
LDI frame (threshold=3.0)	13,924	13,803
Encoded LDI (Aggregation)	131.7	133.8
Encoded LDI (Layer filling)	48.4	48.2
LDI frame (threshold=5.0)	13,808	13,723
Encoded LDI (Aggregation)	91.7	93.0
Encoded LDI (Layer filling)	46.3	47.0

size has not been decreased much as the threshold value is over 3.0 from our experiments. The depth threshold means the difference among actual depth values. For the "Breakdancers" sequence, the given depth range is from 44.0 to 120.0. The residual information mainly depends on the distance among cameras and the actual viewing range. In our experiments, the size of residual data has changed based on the depth thresholding.

There are several issues to be considered in the future experiments. A shape adaptive transform, such as a shape-adaptive discrete wavelet transform (SA-DWT), could be used to encode LDI data because H.264/AVC supports only the 4x4 integer transform. The other one is the temporal prediction of the constructed LDI frames. Because the performance comparison results in terms of PSNR vs. bitrates require the rate allocation for each LDI frame, it could be given after performing the temporal prediction of LDIs.

5 Conclusion

In this paper, we have described the encoding procedure for multi-view video with depth using the concept of the layered depth image (LDI). After generating LDI frames from the natural multi-view video, we have separated them into three components. For color and depth components, we have applied the data aggregation or the layer filling to efficiently encode them. Information for the number of layers (NOL) and residual data used for reconstruction are coded using H.264/AVC. The proposed encoding methods result in the smaller data size than the simulcast case in terms of depth coding. In other words, if we use multi-view video data with the depth information to provide 3-D depth impression, the proposed approach could be useful to process and encode those kinds of data.

Acknowledgements. This work was supported by the Ministry of Information and Communication (MIC) through the Realistic Broadcasting Research Center (RBRC) at Gwangju Institute of Science and Technology (GIST).

References

1. ISO/IEC JTC1/SC29/WG11 N6720: Call for Evidence on Multi-view Video Coding. October (2004)
2. ISO/IEC JTC1/SC29/WG11 N7327: Call for Proposals on Multi-view Video Coding. July (2005)
3. ISO/IEC JTC1/SC29/WG11 N7779: Subjective Test Results for the CfP on Multi-view Coding. January (2006)
4. Shade, J., Gotler, S., Szeliski, R.: Layered Depth Images. Proc. ACM SIGGRAPH July (1998) 291–298
5. Yoon, S.U., Kim, S.Y., Lee, E.K., and Ho, Y.S.: A Framework for Multi-view Video Coding using Layered Depth Images. Volume 3767 of Lecture Notes in Computer Science. November (2005) 431–442
6. ISO/IEC JTC1/SC29/WG11 m12969: Submissions Received in CfP on Multi-view Video Coding. January (2006)
7. Zheng, J., Ji X., Ni, G., and Gao, W.: Extended Direct Mode for Hierarchical B Picture Coding. International Conference on Image Processing, Vol. 2. May (2005) 265–268
8. Interactive Visual Media Group at Microsoft Research. http://research.microsoft.com/vision/InteractiveVisualMediaGroup/3DVideoDownload/
9. Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., and Szeliski, R.: High-quality Video View Interpolation using a Layered Representation. Proc. ACM SIGGRAPH. August (2004) 600–608
10. Duan, J., Li, J.: Compression of the LDI. IEEE Transaction on Image Processing, Vol. 12, No. 3. March (2003) 365–372
11. ISO/IEC JTC1/SC29/WG11 m12485: Generation and Coding of Layered Depth Images for Multi-view Video. October (2005)
12. ISO/IEC JTC1/SC29/WG11 m12849: Reconstruction of Multi-view Images from Layred Depth Images. January (2006)