MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Multi-Layered Coding of Depth for Virtual View Synthesis

Sehoon Yea, Anthony Vetro

TR2009-033 July 2009

Abstract

It is well-known that large depth-coding errors typically occuring around depth edge areas lead to distorted object boundaries in the synthesized texture images. This paper proposes a multi-layered coding approach for depth images as a complement to the popular edge-aware approaches such as those based on platelets. It is shown that guaranteeing a near-lossless bound on the depth values around the edges by adding extra enhancement layers is an effective way to improved the visual quality of the synthesized images.

Picture Coding Symposium 2009

MULTI-LAYERED CODING OF DEPTH FOR VIRTUAL VIEW SYNTHESIS

Sehoon Yea and Anthony Vetro

Mitsubishi Electric Research Labs
201 Broadway, Cambridge, MA 02139, USA

ABSTRACT

It is well-known that large depth-coding errors typically occurring around depth edge areas lead to distorted object boundaries in the synthesized texture images. This paper proposes a multi-layered coding approach for depth images as a complement to the popular edge-aware approaches such as those based on platelets. It is shown that guaranteeing a near-lossless bound on the depth values around the edges by adding extra enhancement layers is an effective way to improve the visual quality of the synthesized images.

Index Terms— depth coding, near-lossless, multi-layered

1. INTRODUCTION

Emerging 3D video applications such as 3DTV and FTV (Free-viewpoint TV) require efficient dissemination of depth information as it is used to generate virtual views, which can be used for free-viewpoint navigation of the scene or various other display processing purposes [1]. A multi-view video plus depth format is currently being explored as part of the recent exploration experiments going on within MPEG on 3D video and FTV. In the multi-view plus depth format, a depth image associated with each view of the multi-view texture videos is estimated, where a pixel in a depth image represents the distance of a 3D scene point seen from the chosen view. Maintaining its fidelity is important because the quality of virtual view synthesis is highly dependent on the accuracy of the geometric information provided by depth. Therefore, it is crucial to strike a good balance between the fidelity of depth data and the associated bandwidth requirement.

In this paper, we propose a multi-layered coding approach for depth images as a complement to the popular edge-aware approaches such as those based on platelets [2]. While the latter focuses on improving the coding efficiency by providing a better representation of edge-like features in the depth image, the proposed multi-layered approach aims at complementing it by explicitly enhancing the fidelity of depth information around the edges. For example, using a platelet-based lossy coder as the base-layer might enhance the coding performance of the overall multi-layered coder compared with the use of other conventional lossy coders as the base layer.

The rest of this paper is organized as follows. Section 2 describes the virtual view synthesis procedure used in this work. Section 3 investigates how the depth coding error affects the texture-mapping process in virtual view synthesis. Next, we discuss the benefits of a multi-layer approach for coding the depth edge area and describe the proposed coding approach in Section 4. Experimental results are shown in Section 5, where it is shown that guaranteeing a near-lossless bound on the depth values around the edges by adding an extra enhancement layer is an effective way to improve the visual quality of the synthesized images. Section 6 concludes the paper.

2. VIRTUAL VIEW SYNTHESIS

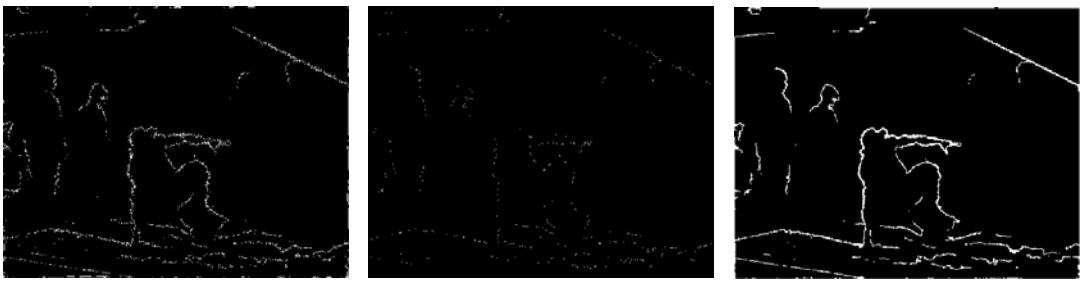
The basic idea of virtual view synthesis is to use the camera geometry along with depth information of the scene to find the texture values for the pixels in a synthesized view from the corresponding points of the neighboring views [1]. In this work, we use two neighboring views in order to synthesize a virtual view at an arbitrary position between them. First, every point in each of the two neighboring views is projected to the corresponding point in the virtual view plane. We assume the well-known pinhole camera model to project the pixel location (x, y) in a neighboring view c into world coordinates $[u, v, w]$ via

$$[u, v, w]^T = R_c \cdot {A_c}^{-1} \cdot [x, y, 1]^T \cdot d[c, x, y] + T_c, \quad (1)$$

where d is the depth defined with respect to the optical center of the camera c, and A, R and T are camera parameters [3]. Then, the world coordinates are mapped into the target coordinates $[x', y', z']$ of the frame in camera v, the virtual view, according to:

$$X_v = [x', y', z']^T = A_v \cdot {R_v}^{-1} \cdot [u, v, w]^T - T_v. \quad (2)$$

After normalizing $X_v = [x', y', z']^T$ by z', we obtain a point in the virtual view $[x'/z', y'/z']$ corresponding to $[x, y]$ in the neighboring view. To perform texture mapping, we copy the depth value together with the corresponding texture value $I[x, y]$ from the current neighboring view (c) into the corresponding location $[x'/z', y'/z']$ in the virtual view depth/texture buffers. Two depth and two texture buffers are



(a) Map of depth coding error ($\delta \geq 4$) (b) Map of texture mapping error (larger than half the maximum) (c) Depth edge-map

Fig. 1. Depth Coding Error vs. Texture Mapping Error vs. Edge Map

maintained since two neighboring views are being used to generate a synthesized view. Due to quantization in finding the projected location in the virtual buffer, the values for some pixels in the virtual view buffers are missing or undefined. In order to render a view, we scan through each location in the two virtual view depth buffers and apply the following procedure:

- If both depth values are zero (no depth information is present), there is no texture information either, causing a hole.
- If only one of the two depth values is non-zero, use the texture value corresponding to the non-zero depth value.
- If both depth values are non-zero, take a weighted sum of the two corresonding texture values.

Various techniques could be applied to improve the quality of the final rendered view such as filtering and in-painting. For simplicity, we only applied a median filter of size 3×3 to recover undefined areas (small stripes) in the synthesized view.

3. EFFECTS OF DEPTH CODING ON SYNTHESIZED VIEWS

The direct transformation from the current camera to the virtual camera can be obtained as follows by combining the equations (1) and (2):

$$X_v = [x', y', z']^T = M_1 \cdot d \cdot X_c + M_2 \tag{3}$$

where $M_1 = A_v \cdot {R_v}^{-1} \cdot R_c \cdot {A_c}^{-1}$ and $M_2 = A_v \cdot {R_v}^{-1} \cdot \{T_c - T_v\}$. If there exists depth-coding error Δd, the correponding error in the location in the virtual camera ΔX_v is given as follows:

$$\Delta X_v = M_1 \cdot X_c \cdot \Delta d \tag{4}$$

Note both X_v and $X_v + \Delta X_v$ need to be normailzed to find corresponding coordinates in the target camera. After such normalization, the texture-mapping error is given as follows:

$$E_{map} = [\frac{x'}{z'}, \frac{y'}{z'}] - [\frac{x' + \Delta X_v(1)}{z' + \Delta X_v(3)}, \frac{y' + \Delta X_v(2)}{z' + \Delta X_v(3)}] \tag{5}$$

Figure 1 shows that due to the tendency that larger depth coding errors occur along the object boundaries (Fig. 1(a)), the texture-mapping errors are also larger around the same boundaries (Fig. 1(b)). Note that we can easily detect these boundaries by using any edge detectors on the depth images or the depth coding residuals (Fig. 1(c)).

4. MULTI-LAYERED CODING OF DEPTH EDGES

From (5), one can observe that the texture-mapping error depends not only on depth coding error but also on other parameters such as camera configurations and the coordinate of the point to be mapped. However, assuming one obtains camera parameters and depth information of sufficient accuracies for the application at hand, having a certain strict control on the fidelity of the depth will be beneficial as it represents geometrical distance of the scene. This is especially true around the depth edges as they typically determine the object boundary which is one of the most important features of any scene.

One way to incorporate such a critical requirement is to use an edge-aware coding technique such as platelets to better represent the edges with a given bit budget. However, such a better representation alone may not always provide enough fidelity of depth around the edges. Therefore, in this work, we propose to add L^∞ fidelity enhancement layers around depth edges in order to impose a strict control on the depth coding error. Previous work in the area of L^∞-error scalable coding is quite sparse [5, 6, 7]. In this work, we adopted the technique in [5], to which the reader is referred to for details.

5. EXPERIMENTAL RESULTS

As an initial attempt to see how imposing a strict control on the depth fidelity of the entire depth-map compares with the

Table 1. Lossy vs. Near-Lossless (δ=7)

	Lossy	Near-Lossless
PSNR of depth (view2), [dB]	48.03	47.25
PSNR of depth (view4), [dB]	48.51	47.97
L^∞-error of depth (view2)	15	7
L^∞-error of depth (view4)	14	7
PSNR (synthesized texture), [dB]	40.35	40.09

case of pure lossy coding in terms of depth coding performance and the corresponding virtual view rendering quality, we compared a two-stage near-lossless coder proposed in [4] against SPIHT as a pure lossy coder.

Figure 2 shows the result of coding depths of view 2 and view 4 using these two coding techniques and the corresponding virtual view synthesis procedure. The maximum per pixel error (δ) for the depth in the two-stage case was set to 7. The bitrate for the lossy coder was set equal to the total bitrate used for the two-stage coder. Table 1 compares the PSNRs and the L^∞-errors of the coded depth using the two approaches. It also compares the PSNRs of the resulting synthesized views that were measured with respect to the one based on uncoded depths. In both approaches, we see noticeable artifacts around object boundaries. For example, the forehead of the person on the left and the top of the hat of the person on the left in Fig 2(a) are significantly distorted or dwarfed.

In order to reduce the above-mentioned distortions around the object boundaries, multiple enhancement layers with decreasing L^∞ errors (δ's) of 3, 1, and 0 were added on top of the near-lossless reconstruction with δ of 7 : this enhancement was applied only to the boundary region specified by the edge map shown in Figure 1(c). Figure 3 shows the result of coding depths of view 2 and view 4 using SPIHT as a pure lossy coder and the multi-layered near-lossless coder [5] as mentioned in the previous section. Table 2 compares the corresponding PSNRs of the coded depths as well as the resulting synthesized views using the two approaches. Note that the result shows only the final reconstruction with $\delta = 0$ for the boundary region and with $\delta = 7$ for other areas although the refinement bitstream was scalable in the L^∞-error sense (i.e., 3,1, and 0) for the boundary area. The bitrate for the 'Lossy' SPIHT was again set equal to the total bitrate used for the multi-layered coder (i.e. the sum of the bitrate for the two-stage with $\delta = 7$ and those of the three refinement layers corresponding to δ's 3,1, and 0).

Observe that although the PSNRs of the whole area of the depth maps are worse in the proposed method, the quality of synthesized view is superior both objectively (PSNRs) and subjectively (Figure 3). Especially we could confirm the distorted boundary areas pointed out in Figure 2 show significant improvement. This shows the proposed approach of explicitly enhancing the depth edges is an effective way to improve the quality of the synthesized view. As a matter of fact, with $\delta = 0$, the area corresponding to the used edge-map has zero texture-mapping error defined by (5).

Table 2. Lossy vs. Multi-Layered Near-Lossless (δ=7,0)

	Lossy	Near-Lossless
PSNR of depth (view2), [dB]	52.02	49.06
PSNR of depth (view4), [dB]	53.18	50.32
PSNR (synthesized texture), [dB]	42.46	44.37

6. CONCLUSION

In this paper, we proposed a multi-layered coding approach for depth images as a complement to the popular edge-aware approaches such as those based on platelets. It is shown that guaranteeing a near-lossless bound on the depth values around the edges by adding extra enhancement layers is an effective way to improve the visual quality of the synthesized images. The proposed approach is flexible in the sense that it could incorporate any lossy coder as the base layer and easily extendible to video cases. This is a notable advantage over platelet-like approaches where their extension to video is still an open problem.

7. REFERENCES

[1] P. Kauff, et al., "Depth Map Creation and Image Based Rendering for Advanced 3DTV Services Providing Interoperability and Scalability", *Signal Processing: Image Communication*, Vol. 22, No. 2, Feb. 2007.

[2] Y. Morvan, D. Farin, and P. H. N. de With, "Depth-Image Compression based on an R-D Optimized Quadtree Decomposition for the Transmission of Multiview Images," *Proc. IEEE Int'l Conf. Image Proc.*, San Antonio, Tx, Sept. 2007.

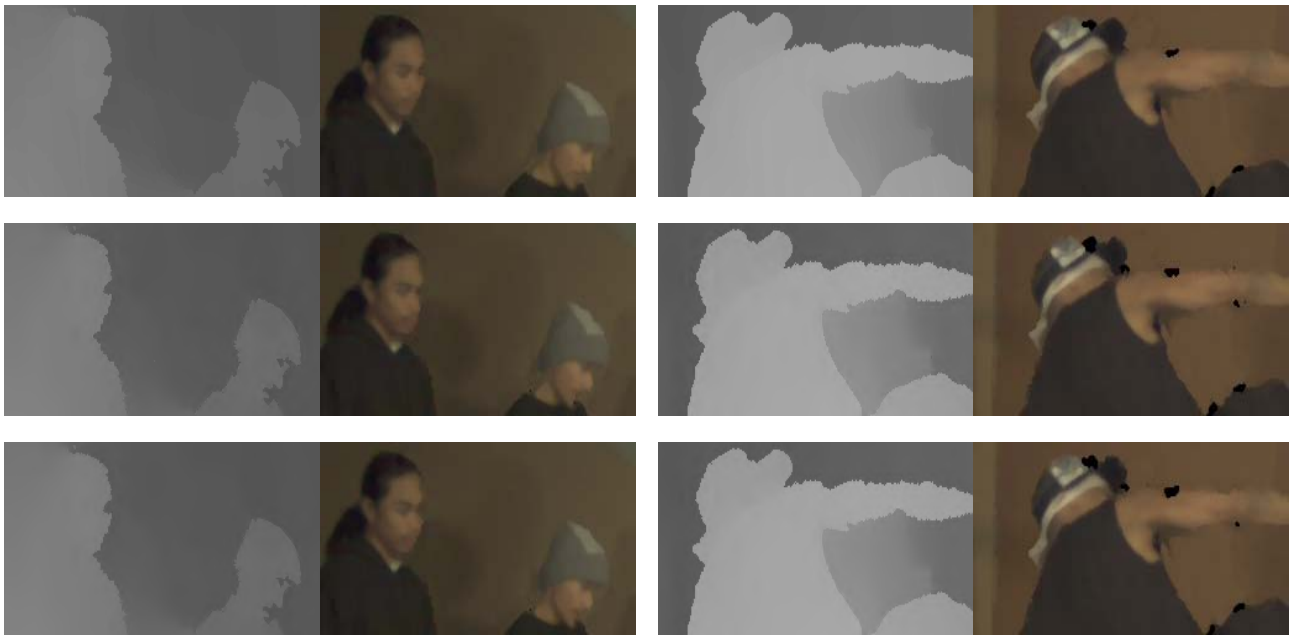
[3] E. Martinian, A. Behrens, J. Xin and A. Vetro, "View synthesis for multiview video compression", *Proc. Picture Coding Symp.*, Beijing, China, Apr. 2006.

[4] S. Yea and W.A. Pearlman, "A Wavelet-Based Two-Stage Near-Lossless Coder," *IEEE Trans. Image Processing*, vol. 15, No. 11, pp. 3488-3500, Nov. 2006.

[5] S. Yea and W.A. Pearlman, "A Wavelet-Based Two-Stage Near-Lossless Coder with L-infinity Error Scalability," *Proc. of SPIE/IS&T Conf. on Electronic Imaging*, San Jose, CA, 2006.

[6] I. Avcibas, N. Memon, B. Sankur and K. Sayood, "A successively refinable lossless image-coding algorithm." *IEEE Trans Communications*, vol. 53, No. 3, Mar. 2005.

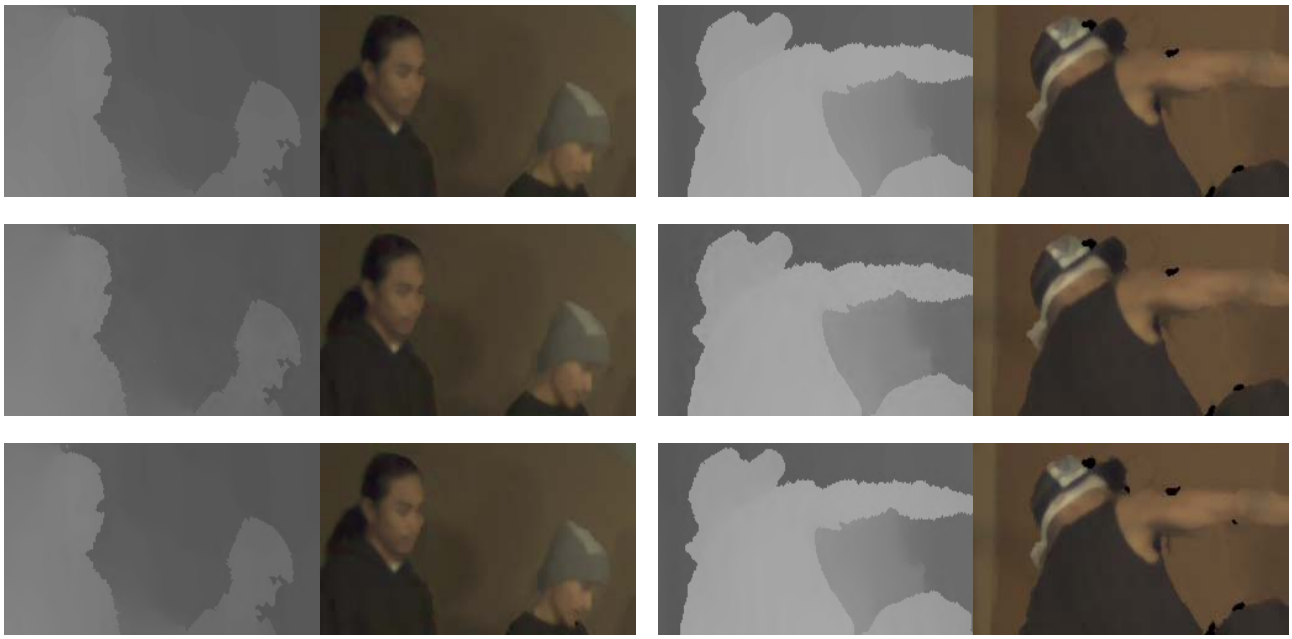
[7] M. U. Celik, G. Sharma and A. M. Tekalp, "Gray-level-embedded lossless image compression," *Signal Processing: Image Communication*, vol. 18, pp. 443-454, 2003.



(a) left: depth (view2), right: synthesized texture (view3)

(b) left: depth (view2), right: synthesized texture (view3)

Fig. 2. From Top to Bottom: Uncoded, Near-lossless ($\delta = 7$), Lossy (0.139 bpp for view2, 0.147 bpp for view4)



(a) left: depth of view2, right: synthesized view3

(b) left: depth of view2, right: synthesized view3

Fig. 3. From Top to Bottom: Uncoded, Multi-Layered Near-Lossless ($\delta = 0$ around the boundary and $\delta = 7$ elsewhere), Lossy (0.268 bpp for view2, 0.296 bpp for view4)