Contour Approximation & Depth Image Coding for Virtual View Synthesis

Yuan Yuan [#1], Gene Cheung [*2], Pascal Frossard [&3], Patrick Le Callet [%4], Vicky H. Zhao [#5]

[#] *University of Alberta, Edmonton, AB Canada*
[1,5] {yyuan13, hvzhao}@ualberta.ca

[*] *The Graduate University for Advanced Studies, National Institute of Informatics, Tokyo, Japan*
[2] cheung@nii.ac.jp

[&] *École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*
[3] pascal.frossard@epfl.ch

[%] *Polytech Nantes / Université de Nantes, Nantes, France*
[4] patrick.lecallet@univ-nantes.fr

***Abstract*—A depth image provides geometric information of a 3D scene, namely the shapes of physical objects captured from a particular viewpoint. This information is important for synthesizing images corresponding to different virtual camera viewpoints via depth-image-based rendering (DIBR). Since it has been shown that blurring of object contours in the depth images leads to bleeding artefacts in virtual images. The most effective way to compress depth images relies on edge-adaptive image codecs that preserve contours, which are losslessly coded as side information (SI). However, lossless coding of the exact object contours can be expensive. In this paper, we argue that the contours themselves can be suitably approximated to save bits, while the depth images piecewise smooth (PWS) characteristic stays preserved. Specifically, we first propose a metric that estimates contour coding rate based on edge statistics. Given an initial rate estimate, we then pro-actively approximate object contours in a way that guarantees rate reduction when coded using arithmetic edge coding (AEC) as SI. Given the sharp but approximated contours, we finally encode the image using an edge-adaptive image codec with graph Fourier transform (GFT) for edge preservation. We show in our experiments that by maintaining sharp but slightly inaccurate object contours, the resulting quality of virtual views synthesized via DIBR exceeds those synthesized using depth images compressed with edge-adaptive codecs that losslessly encode object contours as SI, in particular when the total coding rate budget is low. This confirms that optimized coding of depth images results in an effective tradeoff in the representation of contour and respective depth information.**

I. Introduction

The advent of depth sensing technologies like Microsoft Kinect means that one can now acquire depth images (per-pixel distances between physical objects and capturing camera) of a 3D scene easily. Each depth map provides important geometric information—object shapes and contours from a camera viewpoint—which can be used, together with texture images (color images like RGB) from the same camera viewpoint(s), to synthesize novel images as observed from virtual viewpoints via *depth-image-based rendering* (DIBR) [1]. This ability of virtual view synthesis enables a plethora of 3D imaging applications, including free viewpoint TV [2], immersive video conferenceing [3], etc.

To enable decoder-side virtual view synthesis, depth images must be compressed together with texture images for bandwidth-constrained network transmission. It has been demonstrated [4] that blurring of object contours in a depth map leads to bleeding artefacts in virtual images synthesized via DIBR. Thus state-of-the-art depth image compression algorithms employ edge-adaptive transforms [5, 6] and wavelets [7] to preserve sharp object contours, which are losslessly coded separately as side information (SI). However, the SI coding cost can be expensive at low rates, amounting to 60% of the total bit budget in some cases [6].

In this paper, we argue that *while it is important to maintain a depth image's piecewise smooth (PWS) characteristic even during lossy compression, the object contours themselves can be suitably approximated to reduce SI coding costs.* Specifically, we first define a notion of complexity that estimates the SI coding costs of object contours in a given depth map based on edge statistics. Given an initial rate estimate, we then pro-actively approximate object contours in such a manner that guarantees rate reduction when the simplified contours are coded using arithmetic edge coding (AEC) [8]. Given the sharp but approximated contours, finally we encode the depth image using an edge-adaptive image codec with graph Fourier transform (GFT) for edge preservation [6, 9].

Through extensive experiments, we demonstrate that by maintaining sharp but slightly inaccurate object contours, even at the same compressed depth image PSNR, the resulting quality of DIBR-synthesized virtual views using our compressed depth images far exceeds those synthesized using depth images compressed with DCT-based codecs such as JPEG that blur edges at low bitrates. Further, we show that approximating object contours is indeed more beneficial at low bitrates than encoding the original detected contours losslessly

2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), Oct. 19-21, 2015, Xiamen, China.
978-1-4673-7478-1/15/$31.00

for edge-adaptive transform coding [9], demonstrating the benefit of contour approximation while maintaining the PWS characteristic. To the best of our knowledge, *we are the first in the literature to systematically approximate 2D images with a PWS constraint via contour approximation and edge-adaptive coding*.

The outline of the paper is as follows. We first overview related work in Section II. We then discuss how the PWS signal can be modeled and approximated for the 1D case and the 2D case in Section III and IV, respectively. Experiments and concluding remarks are presented in Section V and VI, respectively.

II. Related Work

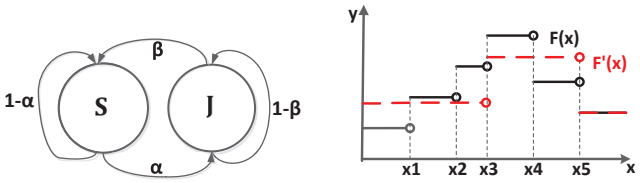
Typical image coding algorithms employ fixed block-based transforms like DCT [10] [11] and can only represent image patches with horizontal and vertical edges well. Directional transforms [12] can adapt to diagonal edges, but cannot deal with more arbitrarily shaped edges such as "L" and "V". Practically, that means coarse quantization of the high-frequency components in these transforms at low bit rates will lead to blurring of arbitrarily shaped edges in the reconstructed depth map.

Having observed that depth map edges play an important role in the quality of the DIBR-synthesized view [4], edge-preserving image coding algorithms have been proposed. [13] modeled depth images with piecewise linear functions (platelet) in each block. However, the representation inherently has a non-zero approximation error even at high rates, since depth images are not strictly piecewise linear but PWS. [14] and [7] proposed edge-adaptive wavelets, and [5] proposed block-based unweighted GFT for depth map coding. Extending [5], [6] used instead a weighted graph for GFT in a multi-resolution framework. In all these works, detected edges are encoded losslessly as SI, which can cost up to 60% of the total budget at low rates.

[8] [15] introduced a lossless AEC method for coding of contiguous object contours: the probabilities for possible directions of the next edge along a contour are estimated and used as context for arithmetic coding of the true edge. In this paper, we extend [8] to approximate object contours by selecting the direction with the largest estimated probability in an edge prediction model to reduce edge coding cost. With the approximated contours, depth maps are subsequently coded using GFT as done in [6].

III. Geometry Approximation: 1D Case

We start our discussion on approximation of PWS signals with the simpler 1D case first (depth pixel row). We first define a notion of complexity for 1D PWS signals, which we can use to estimate a signal's bit count during actual coding implementation. We then present an approximation algorithm that can reduce a signal's complexity while maintaining its PWS characteristic.



(a) Markov model (b) 1D Approximation

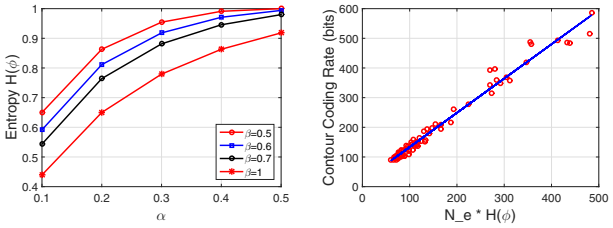
Fig. 1. (a) a two-state Markov model for PWS signal; (b) an example of 1D PWC signal $F(x)$ in black, and an approximated signal $F'(x)$ in red.

A. PWS Signal Complexity: 1D Case

It is observed that depth maps exhibit the PWS characteristic [5, 9], *i.e.*, smooth surfaces divided by sharp edges. For a 1D discrete signal (a pixel row), PWS means a smooth 1D signal interrupted occasionally by discontinuities. Given that the smooth portions can be efficiently coded using edge-adaptive wavelets [7] and transforms [5], we are interested here only in the complexity of representing the set of discontinuities. To model this set, we build a 2-state Markov model as shown in Fig. 1(a): $\mathbf{S}$ and $\mathbf{J}$ are the "smooth" and "jump" (discontinuity) states respectively, with state transition probabilities $Pr(\mathbf{J}|\mathbf{S}) = \alpha$ and $Pr(\mathbf{S}|\mathbf{J}) = \beta$. We now define the complexity of the 1D PWS signal as the entropy $H(\phi)$ of state random variable $\phi \in \{\mathbf{S}, \mathbf{J}\}$, where

$$H(\phi) = \log(\alpha+\beta) - \frac{\alpha}{\alpha+\beta}\log\alpha - \frac{\beta}{\alpha+\beta}\log\beta \qquad (1)$$

We assume that there cannot be two back-to-back discontinuities for a 1D signal, hence $\beta = 1$ and the complexity becomes: $H(\phi) = \log(1+\alpha) - \frac{\alpha}{1+\alpha}\log\alpha$. The more discontinuities a PWS signal has, the larger $Pr(\mathbf{J}|\mathbf{S}) = \alpha$ is, and the larger the complexity $H(\phi)$ is. Fig. 2(a) shows complexity $H(\phi)$ as a function of α ($\alpha \leq 0.5$) for different values of β.



(a) (b)

Fig. 2. (a) The entropy $H(\phi)$ in (1) as a function of α with different values of β; (b) The contour coding bits with respect to our complexity metric $H(\phi)$ for 2D depth map multiplied by the length of contour N_e.

We argue that complexity $H(\phi)$ of a PWS signal is positively correlated with the bit count required to encode the signal's discontinuities as side information (SI). In Fig. 2(b), we plotted the bit count of encoded contours in a 2D depth image using AEC (to be discussed), versus complexity of the contours multiplied by the contour lengths. We observe a positive correlation between bits per symbol and complexity.

B. Approximation of 1D PWS Signal

If the complexity $H(\phi)$ of a contour is large and hence a large coding cost, then one can reduce the number of discontinuities in the signal, resulting in a smaller α and $H(\phi)$. For simplicity, we assume further that the 1D signal is *piecewise constant* (PWC). It can be shown that an original PWC signal containing N jumps is best approximated (in mean squared error (MSE)) by a signal with only K jumps, $K < N$, by selecting K of the original N jump positions. The value of each new longer constant signal segment spanning multiple original pieces is simply the average of those pieces. Fig. 1(b) shows an example approximating original signal $F(x)$ to signal $F'(x)$. We compute the best approximate signal via *dynamic programming* (DP) to identify the best K jump locations among N possible choices as follows.

Denote by $\mathbf{x}$ a PWC signal with $N+1$ constant pieces and N jumps. A PWC signal $\mathbf{x}'$ with K jumps that best approximates $\mathbf{x}$ must eliminate $(N-K)$ original jumps. We define $D_{\mathbf{x}}(i,k)$ as a recursive function that returns the minimum MSE from the i-th constant piece of $\mathbf{x}$ to the $(N+1)$-th piece, given k more jumps need to be eliminated. We first define $d_{\mathbf{x}}(i,j)$ as the MSE when a constant signal segment approximates $j-i+1$ pieces with indices from i to j, where $j \in [i, N+1]$ and $d_{\mathbf{x}}(i,i) = 0$. $d_{\mathbf{x}}(i,j)$ eliminates $j-i$ jumps between i-th and j-th pieces. $D_{\mathbf{x}}(i,k)$ selects the j-th piece that results in the minimum total distortion:

$$D_{\mathbf{x}}(i,k) = \min_{j \in [i,N+1]} \left\{ d_{\mathbf{x}}(i,j) + D_{\mathbf{x}}(j+1, k-(j-i)) \right\} \quad (2)$$

With appropriate base cases defined, recursive call of $D_{\mathbf{x}}(1, N-K)$ would yield the minimum total distortion for approximating signal $\mathbf{x}$ with K jumps.

IV. Geometry Approximation: 2D Case

We now generalize our previous approximation of 1D PWS signal to 2D. We will first introduce a similar notion of complexity for contours in a depth map—an estimate of the overhead for contour coding in the image. Then, to reduce the complexity we describe a procedure to approximate the contours to simpler ones.

A. PWS Signal Complexity: 2D Case

PWS signal in 2D means smooth spatial regions separated by sharp edges. In a 2D depth image, we first detect the edges as done in [6], which outline contours of physical objects and exist *between* pixels. Fig. 3 shows an example of a 4×4 block with *between-pixel edges* $\{e_1, \ldots, e_6\}$—each edge is defined between two vertically or horizontally adjacent pixels—separating foreground and background depth values. The set of edges composing the contour is a 4-connected *chain code*, *i.e.*, next edge starts where current edge terminates, and thus can take on only one of three relative directions: *straight* $\vec{v}_s$ (same direction as previous edge), *left* $\vec{v}_l$ and *right* $\vec{v}_r$ (relative to previous edge). This contour representation is also known as *differential chain code* (DCC) [16].

Using a window of previous edges, [8] proposed a linear regression model to estimate the probabilities of the three possible directions of the next edge, which are subsequently used as context for arithmetic coding of the true next edge. Using a window of K previous edges $\{e_t, \cdots, e_{t-K+1}\}$, one can construct a best fitting line l with direction $\vec{u}_l$ via linear regression to predict the direction of next edge e_{t+1}. The Von Mises probability distribution model [1] is then used to assign probabilities to the three possible directions $\{\vec{v}_s, \vec{v}_l, \vec{v}_r\}$ given their angles with respect to $\vec{u}_l$; smaller angle between a possible direction $\vec{v}$ and $\vec{u}_l$ will result in a larger probability. The computed probabilities, which are synchronously computed at both the encoder and decoder, are then used as context for arithmetic coding of actual edge e_{t+1}. See [8] for details.

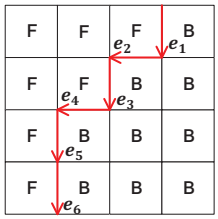


Fig. 3. Edges $e_1, \ldots, e_6$ in a 4×4 block that separate foreground F from background B.

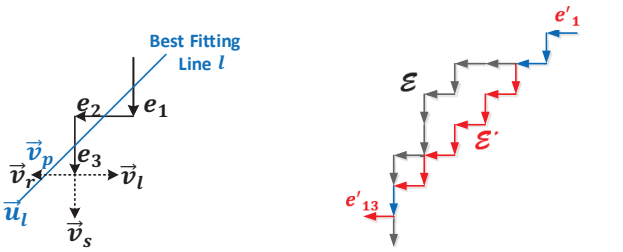
The described linear model provides a notion of "smoothness" in 2D: a smooth contour is one that is accurately predictable by the model. We can thus easily apply our 2-state Markov model for 1D PWS signal to 2D depth map for complexity computation. Among the three possible directions $\{\vec{v}_s, \vec{v}_l, \vec{v}_r\}$ for the next edge e_{t+1}, we term the one with the largest estimated probability the *predicted direction* $\vec{v}_p$. If e_{t+1}'s direction is indeed the same as $\vec{v}_p$, then we label the state $\mathbf{S}$. Otherwise, predicted $\vec{v}_p$ is incorrect, and we label the state $\mathbf{J}$. We thus have the same 2-state Markov model in Fig. 1(a), where $Pr(\mathbf{J}|\mathbf{J}) = 1-\beta > 0$, and we can also define the complexity of contours in 2D depth map as the entropy $H(\phi)$ in (1).

Fig. 4(a) shows an example of using $K = 3$ previous edges $\{e_1, e_2, e_3\}$ to predict the fourth edge e_4. With the best fitting line l, the predicted direction $\vec{v}_p$, one closest to the direction $\vec{u}_l$ of the constructed line l, is $\vec{v}_r$.

B. Approximation of 2D PWC Signal

We now describe a procedure to reduce the complexity $H(\phi)$ by approximating contours if complexity $H(\phi)$ (and hence bit count for coding the contours) is too large. We first define *horizon* h, which is the size of a window of candidate edges in which we will consider contour approximation. A larger h will naturally lead to a larger search space, resulting in a larger amount of approximation. At a given endpoint e_t, with a sequence of K previous edges $\{e_t, \ldots, e_{t-K+1}\}$ of original contour $\mathcal{E}$, we compute the predicted direction $\vec{v}_p$ using the previously described linear regression model. We then consider an approximate contour $\mathcal{E}'$ that replaces the original edge e_{t+1} with $\vec{v}_p$, and continues to replace future edges with next

[1] http://en.wikipedia.org/wiki/Von_Mises_distribution



(a) Edge Prediction [8] (b) Contour approximation

Fig. 4. (a) Edge Prediction: $\vec{v}_p = \vec{v}_r$; if edge e_4 is in direction $\vec{v}_r$, the state will be **S**. If edge e_4 is in direction $\vec{v}_s$ or $\vec{v}_l$, the state will be **J**. (b) Contour Approximation: the original contour $\mathcal{E}$ (in black) is approximated by contour $\mathcal{E}'$ (in red). Here $K = 3$ and $h = 13$. Edges in blue means original and approximated edges are coincident.

predicted $\vec{v}_p$, until either: i) a predicted edge $\vec{v}_p$ is actually part of the original contour $\mathcal{E}$, or ii) the approximation has reached length h. If it is the earlier termination condition, then we have successfully replace a section of the original $\mathcal{E}$ with a new segment that is perfectly predicted by the linear regression model, thus lowering α and the complexity $H(\phi)$. If it is the latter termination condition, then no approximation is executed, and the procedure repeats at edge e_{t+1}.

Fig. 4(b) shows an approximated contour $\mathcal{E}'$ to orginal $\mathcal{E}$. Fig. 5 shows a subjective result of our approximation for the `teddy` image, where larger h leads to smoother contours.

C. Edge-adaptive Coding of Depth Blocks

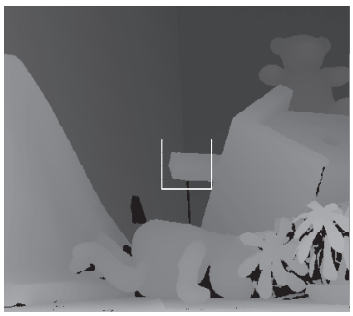
Given the approximated contours, coded losselessly using AEC [8], we now describe how actual pixel values of a depth image can be coded edge-adaptively block-by-block via GFT [9], preserving the image's PWS characteristic. First, pixel values on a depth image are exchanged according to the approximated contours, so that foreground depth values fall on one side of a contour and background depth values fall on the other. Then, for each pixel block a 4-connected graph is constructed, where each node corresponds to a pixel. An edge weight is assigned 1 if the connected pixels are on the same side of a contour, and 0 otherwise. Given the constructed graph, the transform is the eigen-matrix of the graph Laplacian. This weight assignment (deducible from the encoded contours) means filtering across sharp boundaries are avoided, preventing blurring of edges. The computed transform is multiplied with the vectorized depth block signal for the transform coefficients, which are quantized and entropy-coded into a bitstream. See [6] for details.

V. Experimentation

A. Experimental Setup

We performed extensive experiments to validate the effectiveness of our proposed depth map contour approximation method using texture-plus-depth image sequences from the Middlebury dataset[2]. We first approximated contours in each

[2]http://vision.middlebury.edu/stereo/data/



(a) `teddy`

(b) (c) (d) (e)

Fig. 5. (a) `teddy`. (b) original interception of (a). (c) $\sim$ (e) are the approximation results with increasing value of h. The contour becomes smoother with h increasing.

original depth map with different horizon scale λ, where the horizon value h is equal to the length of contours times the scale. We then deployed GFT [5] to code pixel blocks in depth maps (termed approximated GFT or `AGFT` for short). Finally combining with original or JPEG compressed texture maps, we synthesized middle views via DIBR [1], which were compared against original middle view images for distortion computation.

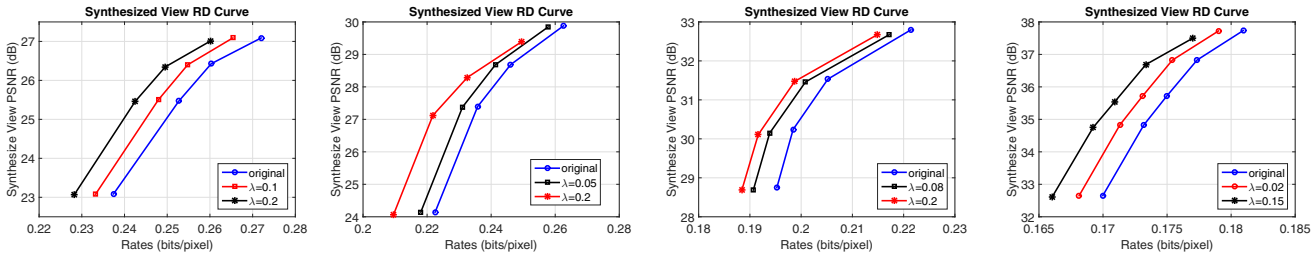
B. Results: Comparing with Original Depth Map

We first demonstrate the benefit of effectively approximating contours before edge-adaptive image compression. We compared the RD curves of virtual views synthesized using depth maps compressed by our proposed `AGFT` with those synthesized using depth maps compressed using GFT with original detected edges losslessly coded (termed `GFT`). JPEG was used to compress texture maps. Fig. 6 shows the RD curves for `cones`, `Dolls`, `Rocks2` and `Lampshade2` sequences, respectively. The x-axis is the depth map coding rate in bits per pixel (bpp), and the y-axis is the synthesized view quality in PSNR. The computed BG gains [17] are shown in Table I, where we see that the average BG gain in PSNR achieves 1.68dB over `GFT` and 4.67dB over JPEG.

Our approximated depth maps resulted in better RD performance at low bit rates, since by approximating contours we can save significant coding bits while the synthesized view distortion due to approximation is negligible. Subjective results will be shown in next subsection.

C. Results: Comparing with JPEG Compressed Depth Map

We now demonstrate the importance of edge preservation in compressed depth maps. We compared synthesized images



(a) cones (b) Dolls (c) Rocks2 (d) Lampshade2

Fig. 6. Synthesized virtual view RD curves for cones, Dolls, Rocks2 and Lampshade2, respectively. Here 'original' means virtual view synthesized using depth maps compressed adaptive to original edges. "λ" referred curves meaning virtual view synthesized with correspondingly approximated depth map based on our proposed method.

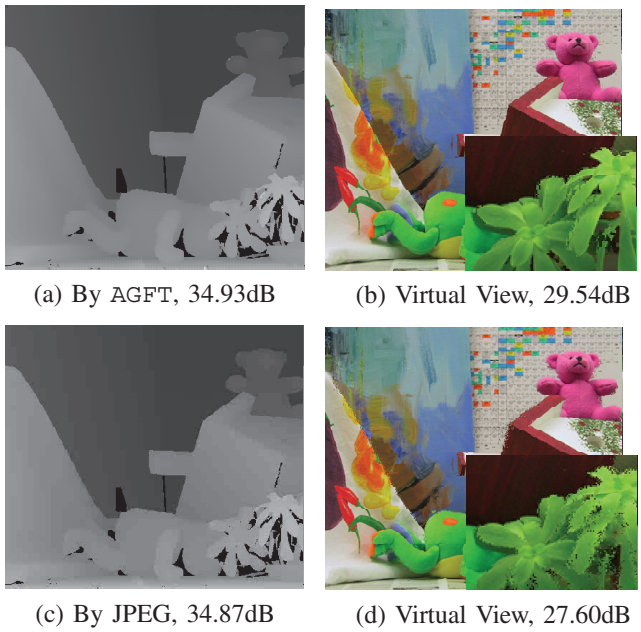
TABLE I
BG GAIN IN PSNR FOR MIDDLEBURY SEQUENCES

Sequences	GFT	JPEG
teddy	1.29	1.45
cones	0.92	3.34
Dolls	2.17	2.41
Rocks2	2.30	7.14
Lampshade2	1.72	9.01
Average	1.68	4.67

using depth images compressed by our AGFT, which preserves sharp edges, with depth images compressed by JPEG. Fixed DCT-transform coding in JPEG compression leads to blurred edges at low bitrates. Since it has already been demonstrated in [9] that GFT outperformed JPEG in depth map compression in terms of RD performance, we show here that even for similar depth map PSNR, the resulting synthesized images using AGFT compressed depth maps outperform images synthesized using JPEG compressed depth maps, since AGFT maintains the PWS characteristic.

Fig. 7 shows the AGFT and JPEG compressed depth maps and the corresponding synthesized virtual views. We observe that AGFT compressed depth map (a) have sharper edges comparing with JPEG compressed result (c), even though they have nearly equal PSNR values (the difference is less then 0.1dB for both left and right depth maps). (b) and (d) are the corresponding synthesized views. We enlarged the right-bottom corner for each image to enhance the visual quality, where there are corrupted edges in (d) while (b) looks clearer with sharp edges. This is also reflected in the corresponding PSNR values. The results illustrated that even for the similar quality of depth maps, an edge preserved depth map can result in a higher quality of synthesized view, which supports the importance and effectiveness of our edge adaptive approximation method.

Sub-regions of the synthesized views for the previous mentioned four sequences are shown in Fig. 8. The first row are synthesized by AGFT compressed depth maps and second row are by JPEG compressed depth maps, where the depth map quality are of nearly the same PSNR values. There are a lot of corrupted edges in the JPEG compressed results (e.g. cones and Rocks2), which badly affected visual quality. For AGFT compressed results, although there is also some distortion around edges due to approximation, the edges in synthesized views still look very sharp (e.g. Dolls).



(a) By AGFT, 34.93dB (b) Virtual View, 29.54dB

(c) By JPEG, 34.87dB (d) Virtual View, 27.60dB

Fig. 7. For teddy, the same quality of depth maps compressed by AGFT (a) (coding bits: 0.20 bpp) and JPEG (c) (coding bits: 0.72 bpp), respectively. Together with original texture maps, the corresponding synthesized views are shown in (b) and (d). Visually, (d) has a lot of noise around edges while (b) is very clean. The right-bottom corner is enlarged for better visual quality.

VI. CONCLUSION

Efficient coding of depth image is essential for decoder-side virtual view synthesis via depth-image-based rendering (DIBR). Existing works either employ fixed transforms like DCT that blur depth image's sharp edges at low rates, or use edge-adaptive transforms that require lossless coding of detected edges as side information, which accounts for a large share of the bit budget at low rates. In this paper, we propose to first approximate object contours to lower the edge coding

(a) cones (b) Dolls (c) Rocks2 (d) Lampshade2

Fig. 8. Sub-regions of the synthesized views by original texture maps and AGFT / JPEG compressed depth maps, respectively, where the PSNR quality of depth maps compressed by AGFT and JPEG are almost the same. The first row are by AGFT compressed while the second row are by JPEG. Rocks2 and Lampshade2 are enlarged to be more distinct. (We strongly recommend readers to read an electronic version to distinguish the differences.)

cost, then use edge-adaptive graph Fourier transform (GFT) for block-based coding so that sharp edges are preserved and the piecewise smooth (PWS) characteristic is maintained. Experiments show noticable performance gain over previous coding schemes using either fixed transform or edge-adaptive transform with lossless coding of detected contours.

REFERENCES

[1] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, "View synthesis techniques for 3D video," in *Applications of Digital Image Processing XXXII, Proceedings of the SPIE*, vol. 7443 (2009), 2009, pp. 74 430T–74 430T–11.

[2] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," in *IEEE Signal Processing Magazine*, vol. 28, no.1, January 2011.

[3] C. Zhang, Z. Yin, and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing," in *IEEE International Workshop on Multimedia Signal Processing*, Rio de Jeneiro, Brazil, October 2009.

[4] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map coding with distortion estimation of rendered view," in *SPIE Visual Information Processing and Communication*, San Jose, CA, January 2010.

[5] G. Shen, W.-S. Kim, S. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[6] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *Image Processing, IEEE Transactions on*, vol. 24, no. 1, pp. 419–433, 2015.

[7] M. Maitre, Y. Shinagawa, and M. Do, "Wavelet-based joint estimation and encoding of depth-image-based representations for free-viewpoint rendering," in *IEEE Transactions on Image Processing*, vol. 17, no.6, June 2008, pp. 946–957.

[8] I. Daribo, G. Cheung, and D. Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video compression," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*. IEEE, 2012, pp. 1541–1544.

[9] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[10] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103–1120, 2007.

[11] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *Signal Processing Magazine, IEEE*, vol. 18, no. 5, pp. 36–58, 2001.

[12] B. Zeng and J. Fu, "Directional discrete cosine transforms for image coding," in *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, 2006, pp. 721–724.

[13] Y. Morvan, P. de With, and D. Farin, "Platelets-based coding of depth maps for the transmission of multiview images," in *SPIE Stereoscopic Displays and Applications*, San Jose, CA, January 2006.

[14] A. Sanchez, G. Shen, and A. Ortega, "Edge-preserving depth-map coding using graph-based wavelets," in *43th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 2009.

[15] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped motion prediction for depth video compression using arithmetic edge coding," *Image Processing, IEEE Transactions on*, vol. 23, no. 11, pp. 4696–4708, 2014.

[16] H. Freeman, "On the encoding of arbitary geometrics configurations," in *IEEE Transactions on Electronic Computers*, vol. no.2, 1961, pp. 260–268.

[17] G. Bjontegaard, "Calcuation of average psnr differences between rd-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.