DEPTH MAP COMPRESSION USING MULTI-RESOLUTION GRAPH-BASED TRANSFORM FOR DEPTH-IMAGE-BASED RENDERING

Wei Hu^o, Gene Cheung$^\#$, Xin Li*, Oscar Au^o

^{o} Hong Kong University of Science and Technology, $^\#$ National Institute of Informatics,
* West Virginia University

ABSTRACT

Depth map compression is important for efficient network transmission of 3D visual data in texture-plus-depth format, where the observer can synthesize an image of a freely chosen viewpoint via depth-image-based rendering (DIBR) using received neighboring texture and depth maps as anchors. Unlike texture maps, depth maps exhibit unique characteristics like smooth interior surfaces and sharp edges that can be exploited for coding gain. In this paper, we propose a multi-resolution approach to depth map compression using previously proposed graph-based transform (GBT). The key idea is to treat smooth surfaces and sharp edges of large code blocks separately and encode them in different resolutions: encode edges in original high resolution (HR) to preserve sharpness, and encode smooth surfaces in low-pass-filtered and down-sampled low resolution (LR) to save coding bits. Because GBT does not filter across edges, it produces small or zero high-frequency components when coding smooth-surface depth maps and leads to a compact representation in the transform domain. By encoding down-sampled surface regions in LR GBT, we achieve representation compactness for a large block without the high computation complexity associated with an adaptive large-block GBT. At the decoder, encoded LR surfaces are up-sampled and interpolated while preserving encoded HR edges. Experimental results show that our proposed multi-resolution approach using GBT reduced bitrate by 68% compared to native H.264 intra with DCT encoding original HR depth maps, and by 55% compared to single-resolution GBT encoding small blocks.

Index Terms— Depth map coding, multi-resolution, graph-based transform

1. INTRODUCTION

Among many proposed representations of 3D visual data in the literature is *texture-plus-depth* format [1], where texture maps (RGB images) and depth maps (per-pixel physical distances between captured objects in the 3D scene and the capturing camera) of multiple closely spaced viewpoints are encoded and transmitted from server to the observing client. The observer can in turn synthesize an image of any freely chosen viewpoint via depth-image-based rendering (DIBR) techniques like 3D warping [2], using received texture and depth maps of neighboring viewpoints as anchors. Transmission of multiple large texture and depth maps of different viewpoints, however, translates to a high bandwidth requirement and expensive network costs. Thus, compression of 3D data in texture-plus-depth format is an important research problem.

While compression of texture maps is well studied, compression of depth maps is relatively new, and has been the focus of many recent research efforts [3, 4, 5]. Typical depth maps exhibit unique characteristics, such as sharp edges and smooth surfaces interior to the sharp edges, that are quite different from texture maps, and previous depth map compression algorithms have attempted to exploit these characteristics for coding gain. For example, *graph-based transform* (GBT) [4] is an adaptive block-based transform (thus can be easily integrated into block-based coding standards like H.264) that avoids filtering across defined edges. For picewise-smooth signals like depth maps, GBT produces small or zero high-frequency components and leads to a compact representation in the transform domain. Because GBT is block-adaptive (defined using detected edges in the code block), GBT must be computed on-the-fly via eigen-decomposition, which is computationally expensive for large blocks (the computational complexity of eigen-decomposition for a $n \times n$ matrix is $\mathcal{O}(n^3)$ in practice). Thus, GBT was used only for small blocks (4×4 in [4]).

In this paper, to encode large blocks with GBT efficiently—leading to potentially more coding gain for large smooth surfaces—we propose a new *multi-resolution* approach using GBT to code piecewise-smooth 2D signals such as depth maps. The key observation is that to exploit the surface-smoothness prior in a large block in a computation efficient manner—where by "smooth" we mean the surfaces inside sharp edges have predominantly low-frequency components—one can first down-sample the interior surface from original high resolution (HR) to low resolution (LR) before encoding in LR GBT for computation efficiency. The surface-smoothness prior ensures us that high-frequencies lost (if any) during low-pass filtering before down-sampling (to avoid aliasing) would be minimal. On the other hand, to preserve sharpness (which greatly affects DIBR-synthesized view quality for depth maps [5]), we should encode edges in original HR.

Specifically, we perform the following operations. Given a target down-sampling factor K and pixel block of size $Kn \times Kn$ (here $n = 4$), we first detect and encode prominent edges in the block losslessly. Then, we perform low-pass-filtering in the pixel domain for anti-aliasing purposes prior to down-sampling. We down-sample the block to $n \times n$, quantize and encode LR GBT coefficients of the now smaller block for transmission. At decoder, the $n \times n$ block is reconstructed in pixel-domain, and interpolated back to $Kn \times Kn$ while respecting the losslessly encoded HR edges. That means a missing pixel is interpolated using neighboring pixels on the same side of HR edges, thus preserving edge sharpness. Experimental results show that our proposed multi-resolution approach using GBT reduced bitrate by 68% compared to native H.264 intra with DCT encoding original HR depth maps, and by 55% compared to single-resolution GBT encoding small blocks.

The outline of the paper is as follows. We first discuss related work in Section 2. We then overview fundamentals of GBT in Section 3. We discuss our multi-resolution GBT encoder and decoder in Section 4 and 5, respectively. Finally, results and conclusions are presented in Section 6 and 7, respectively.

2. RELATED WORK

One line of attack for depth map coding is to exploit the observation that different depth pixels affect the synthesized view distortion unequally during DIBR. For example, depth pixels close to an object edge are likely to affect synthesized view distortion more than depth pixels interior to an object. Exploiting this observation, [6] optimized mode selection during H.264 coding of depth video, while [5] manipulated unimportant depth pixels (without causing severe synthesized view distortion) towards a sparse representation of the depth signal in the transform domain. In contrast, our work exploits the unique characteristics of depth maps (sharp edges and smooth interior surfaces) for coding gain.

Like edge-adaptive wavelets [3], block-based GBT [4] avoids filtering across pre-defined edges, resulting in a compact transform domain representation of the signal, even though lossless encoding of the block edges entails an overhead for the adaptive transform. [7] proposed directional transform to align transform with the predominant direction in the block's textural content. Unlike directional transforms [7], GBT can handle more complicated edges such as the "L"- or "V"-shaped. We extend work in [4] to efficient coding of large depth blocks via a multi-resolution approach.

3. GRAPH-BASED TRANSFORM

We first overview the three-step construction procedure of GBT [4]. First, prominent edges in a $n \times n$ pixel block are detected. Then, a graph describing the pixel connectivity given the detected edges (two neighboring pixels are connected except when divided by an edge) is constructed. Finally, an adaptive transform is built based on the connectivity graph.

In the first step, we detect edges in a block based on the difference between the neighboring pixel values using a simple thresholding technique [4]. Because the transform is adaptive, the decoder must have the edge information available to built the GBT, which usually entails lossless encoding of the block edges.

In the second step, we treat each pixel in the $n \times n$ block as a node in a graph $\mathcal{G}$, and connect it to its four or eight immediate neighbors in the block, resulting in a 4- or 8-connectivity graph. Then, if there is an edge between two neighboring pixels / nodes, we eliminate their connection. Given the connectivity graph, we can define an adjacency matrix $\mathbf{A}$, where $\mathbf{A}(i,j) = \mathbf{A}(j,i) = 1$ if pixel positions i and j are connected, and 0 otherwise. We can similarly compute the degree matrix $\mathbf{D}$, where $\mathbf{D}(i,i)$ is the number of connections for node i, and $\mathbf{D}(i,j) = 0$ for all $i \neq j$.

In the third step, using computed $\mathbf{A}$ and $\mathbf{D}$, we can compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [8]. If we now project a signal $\mathbf{x}$ in the graph $\mathcal{G}$ onto the eigenvectors of the Laplacian $\mathbf{L}$, it becomes the spectral decomposition of the signal; i.e., it provides a "frequency domain" interpretation of signal $\mathbf{x}$ given graph support $\mathcal{G}$. Hence, we can construct GBT transform using eigenvectors of $\mathbf{L}$. In particular, we can stack pixels in the $n \times n$ block into a length-n^2 vector and compute $\mathbf{y} = \mathbf{E}^t \cdot \mathbf{x}$, where $\mathbf{E}$ is a matrix with eigenvectors of $\mathbf{L}$ as columns. GBT transform coefficients are then $\mathbf{y}$, which we can quantize and entropy-encode for compression & transmission.

4. MULTI-RESOLUTION GBT ENCODING

We propose a multi-resolution depth map compression scheme using GBT, exploiting depth maps' unique characteristics of sharp edges and smooth interior surfaces (piecewise-smooth) of large code blocks. The key is to encode edges and surfaces in different resolutions: encode edges in original HR version losslessly to preserve sharpness, and encode smooth surfaces in low-pass-filtered and down-sampled LR version in GBT domain to save coding bits while reducing computation complexity. At the decoder, the LR surfaces are up-sampled and interpolated while respecting the losslessly encoded HR edges. Further, because edge pixels have higher synthesized view distortion sensitivity than smooth surfaces during DIBR [5] (e.g., wrong edge pixels would lead to confusion of foreground and background during view synthesis), our proposed depth map coding scheme results in better synthesized views.

Our proposed depth map compression scheme can be explained step-by-step as follows. At encoder, for a target down-sampling factor K and $Kn \times Kn$ pixel block, there are three steps: i) detect edges in the block for losslessly encoding, and down-sample them to LR edges for definition of LR GBT of the down-sampled $n \times n$ small block; ii) low-pass-filter $Kn \times Kn$ block in the pixel domain in an edge-adaptive way and down-sample the block to $n \times n$; and iii) perform LR GBT on $n \times n$ block, quantize the resulting transform coefficients, then transmit both losslessly encoded HR edges and quantized LR GBT coefficients to the decoder.

At the decoder, we do the following two-step reconstruction procedure: i) perform inverse quantization and inverse LR GBT using LR edges (down-sampled from the decoded HR edges) to reconstruct $n \times n$ block; and ii) up-sample to $Kn \times Kn$ block and interpolate missing pixels while respecting the losslessly encoded HR edge map. The overall compression scheme is shown in Fig.1. We describe the three-step encoding procedure in the following, and the two-step decoding procedure in Section 5.

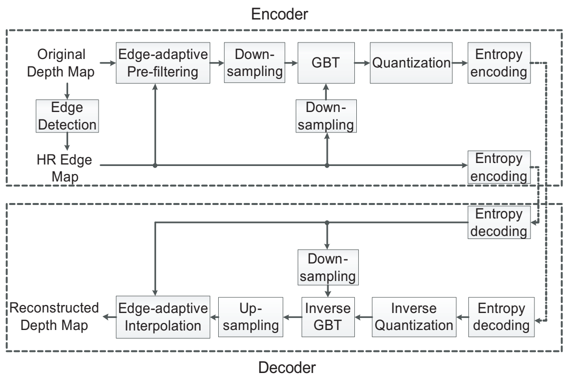


Fig. 1. Multi-resolution depth map coding scheme based on GBT.

4.1. Edge Detection & Down-sampling

In the first step, we detect edges in the original $Kn \times Kn$ block for lossless encoding, and down-sample HR edges for definition of LR GBT. We use the edge detection technique in [9], where the presence of edges between every pixel and its immediate four (or eight) neighbors is provided by an edge map in full (or half)-pixel positions. The edges of the original depth map, which we refer to as the HR edges, are then down-sampled to LR edges with the same down-sampling rate as its corresponding pixel block, so that the LR edges can be used for definition of LR GBT for $n \times n$ block. Fig. 2 shows an example of a 6×6 block with edges separating foreground and background depth pixels, which is subsequently down-sampled to a 3×3 block with corresponding LR edges.

The edge down-sampling is actually a process to decide if there are edges between every pixel and its immediate neighbors on the LR pixel grid. Since the immediate neighbors on the LR pixel grid are not directly adjacent on the HR pixel grid, while the HR edge map only shows edge information among immediate neighbors, we find the LR edge information by searching along a straight path connecting a pair of adjacent LR pixels on the HR pixel grid. If there is no edge between any pair of adjacent pixels along this path on the HR pixel grid, we assume there is no edge between the two pixels on the LR pixel grid.

4.2. Low-pass-filtering & Down-sampling

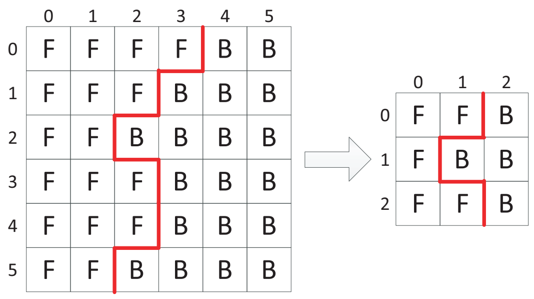


Fig. 2. The proposed depth map down-sampling approach.

In the second step, we low-pass-filter and down-sample the $Kn \times Kn$ block uniformly to a $n \times n$ block. To avoid aliasing caused by down-sampling, we perform low-pass filtering in the pixel domain before down-sampling. Many designs of anti-aliasing filters have been proposed, such as adaptive directional low-pass filters in [10], where directional Gaussian low-pass filters are used according to edge direction. However, the filtering is performed in four fixed directions, so pixels across edges in other directions will be smeared. Since edges of depth maps are very important to DIBR, we instead low-pass filter while respecting the original edges. More specifically, a pixel is low-pass filtered by taking average of its neighbors on the same side of HR edges within a $(2K-1) \times (2K-1)$ window centering at the to-be-filtered pixel. The advantage of this edge-adaptive low-pass filtering is that filtering across edges will not occur, so pixels across edges will not contaminate each other through filtering.

4.3. LR GBT Coding

In the third step, we perform LR GBT on the filtered $n \times n$ block and quantize the resulting transform coefficients. Then, the encoded quantized coefficients and HR edges are sent to the decoder. Note that the HR edges will be losslessly encoded and transmitted instead of the LR edges, so that depth map interpolation after up-sampling respecting the HR edges can be performed at the decoder. Further, using the received HR edges, the decoder can mimic the edge down-sampling process discussed in Section 4.1 to get the same LR edges for construction of the LR GBT.

5. GBT DECODING & INTERPOLATION

At the decoder, we first perform inverse quantization and inverse LR GBT to recover the LR $n \times n$ pixel block. To identify the correct adaptive LR GBT transform used at encoder, we down-sample the encoded HR edges to LR edges as done at the encoder. After recovering the LR $n \times n$ block, we up-sample it to the original HR $Kn \times Kn$ block, and fill in missing pixels via image-based *edge-adaptive interpolation* as follows.

We interpolate a pixel x by taking average of its nearest connected LR pixels within a $(2K-1) \times (2K-1)$ window centering at pixel x. By "connected" we mean that the LR pixels are on the same side of losslessly encoded HR edges with the missing pixel, so that pixels across edges will not contaminate the missing pixel through interpolation. The equation below describes our interpolation approach:

$$x = \sum_{i \in W(x)} \delta_i w_i y_i, \qquad (1)$$

where

$$\delta_i = \begin{cases} 1, & \text{if } y_i \text{ is connected to } x \\ 0, & \text{o.w.} \end{cases} \qquad (2)$$

and $W(x)$ is the index set which corresponds to LR pixels y within the interpolation window of pixel x. w_i are the weights, which are inversely proportional to the geometric distance between y_i and x and sum up to 1.

Our interpolation method is effective in preserving the original edges because missing pixels next to edges are interpolated parallel to the losslessly encoded HR edges. Further, the method is much simpler than most methods in current super-resolution literature, such as the super-resolution algorithm via TV-regularization [11].

6. EXPERIMENTATION

6.1. Experimental Setup

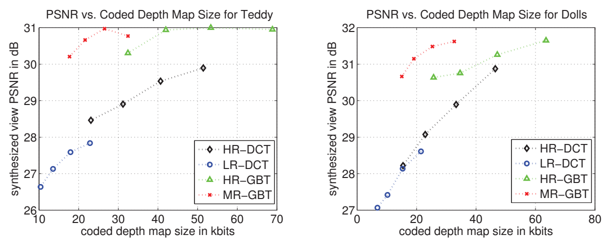
We implemented our proposed multi-resolution depth map coding scheme (`MR-GBT`) inside H.264/AVC Reference Software JM 17.1[1]. Depth edges are encoded using CABAC, as done in H.264/AVC. Two Middlebury multiview image sets `Teddy` and `Dolls`[2] were tested. We encoded and decoded the ground truth left and right depth maps, and together with original left and right texture maps, we synthesized the texture image of the middle view. Here, DIBR was performed using a simple implementation of 3D warping [2]. Our compression scheme was compared against three other schemes: (1) H.264 intra with DCT encoding original HR depth maps (`HR-DCT`); (2) H.264 intra using DCT encoding reduced resolution LR depth maps with isotropic Gaussian low-pass filtering and Total Variation (TV)-regularized interpolation [11] (`LR-DCT`); and (3) single-resolution GBT encoding original HR depth maps with 4×4 code blocks (`HR-GBT`). For all schemes, uniform quantization was used, with fixed quantization parameter (QP) values of 24, 28, 32 and 36.

6.2. Experimental Results

Fig. 3 shows RD curves of the aforementioned four coding schemes, where the synthesized view PSNR is calculated with respect to the ground truth middle image and the down-sampling factor is $K = 2$ for `LR-DCT` and `MR-GBT`. We see that while `HR-GBT` performed better than `HR-DCT`, `MR-GBT` reduced bit rate by 40% and 55% for `Teddy` and `Dolls` respectively, compared to `HR-GBT`, and reduced by 68% and 65% compared to `HR-DCT`. It is expected that better performance can be achieved with more efficient encoding of the HR edge map (account for up to 39% and 48% of the bitrate at coarse QP for `Teddy` and `Dolls`, respectively). `LR-DCT` had the worst RD performance; though bit rate was low, the TV-regularized interpolation method produced noisy and blurred depth edges, leading to poor synthesized view quality. This shows that while coding depth maps at LR can save bits, edges must be well preserved to maintain high reconstruction quality, as done in `MR-GBT`.

The RD performance of our proposed `MR-GBT` using different down-sampling factors is shown in Fig. 4 to test how far we can down-sample the interior surfaces inside edges while keeping acceptable RD performance. With larger down-sampling factor, bit

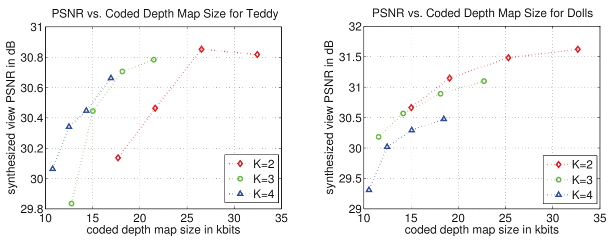
[1]http://iphome.hhi.de/suehring/tml/
[2]http://cat.middlebury.edu/stereo/newdata.html



(a) (b)

Fig. 3. RD curves of different coding schemes for `Teddy` and `Dolls` respectively. A down-sampling factor $K = 2$ in each dimension is used for both `LR-DCT` and our `MR-GBT`.

rate was further reduced for `Teddy` while the RD performance worsened for `Dolls`. This is because the down-sampling limit is dependent on the surface-smoothness prior, which determines the extent of high-frequency lost during anti-aliasing low-pass filtering. In our case, the piecewise-smooth prior was stronger (more smooth) for `Teddy` than `Dolls`, resulting in difference in RD performance for various down-sampling factors.



(a) (b)

Fig. 4. RD curves of our `MR-GBT` using different down-sampling factors for `Teddy` and `Dolls` respectively.

Finally, Fig. 5 shows the subjective quality comparison of reconstructed depth maps and resulting synthesized views for `LR-DCT` and `MR-GBT` at comparable bitrate with the down-sampling factor $K = 2$. We see that while there exist blocky artifacts in the interior surfaces of both `LR-DCT` and `MR-GBT`, the edges produced by `MR-GBT` are much sharper and the surfaces are not contaminated by noise. As a consequence, the synthesized view generated using `MR-GBT` was also cleaner than one by `LR-DCT`.

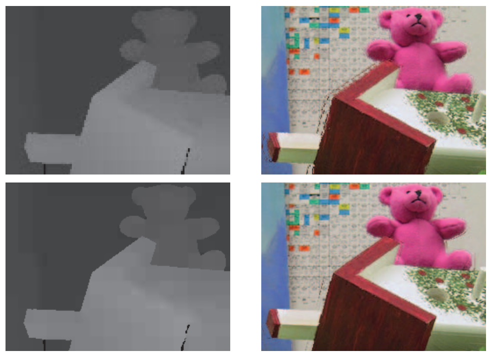


Fig. 5. The reconstructed left depth maps and synthesized views of `Teddy` using `LR-DCT` (first row) and our `MR-GBT` (second row) respectively, with code rate of `MR-GBT` 7% less and the down-sampling factor $K = 2$.

7. CONCLUSION

In this paper, we propose a multi-resolution GBT based approach to depth map compression. Taking advantage of the piecewise-smooth characteristic of depth maps, we encode edges in original resolution to preserve sharpness, and encode smooth surfaces in down-sampled low resolution to save coding bits while achieving computation efficiency. GBT-based low-pass-filtering is proposed to avoid aliasing prior to down-sampling, while edge-adaptive interpolation is proposed to restore the decoded LR depth maps to the original resolution. Both methods preserve edges well, which along with down-sampling leads to significant coding gain, with up to 68% bit rate reduction compared to native H.264 intra with DCT encoding original HR depth maps, and 55% bit rate reduction compared to single-resolution GBT encoding original HR depth maps with small code blocks. Better performance is expected with more efficient edge map coding.

8. ACKNOWLEDGMENT

This work has been supported in part by the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China.

Also, we thank Dr. Woo-shik Kim and Prof. Antonio Ortega of University of Southern California for sharing their GBT software.

9. REFERENCES

[1] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.

[2] W. Mark, L. McMillan, and G. Bishop, "Post-rendering 3D warping," in *Symposium on Interactive 3D Graphics*, New York, NY, April 1997.

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

[4] G. Shen, W.-S. Kim, S.K. 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.

[5] G. Cheung, J. Ishida, A. Kubota, and A. Ortega, "Transform domain sparsification of depth maps using iterative quadratic programming," in *IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011.

[6] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map distortion analysis for view rendering and depth coding," in *IEEE International Conf. on Image Processing*, Cairo, Egypt, November 2009.

[7] B. Zeng and J. Fu, "Directional discrete cosine transforms for image coding," in *IEEE International Conference on Multimedia and Expo*, Toronto, Canada, June 2006.

[8] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," in *Elsevier: Appplied and Computational Harmonic Analysis*, April 2010, vol. 30, pp. 129–150.

[9] 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.

[10] X. Zhang and X. Wu, "Can lower resolution be better?," in *IEEE International Conf. on Data Compression*, Snowbird, UT, March 2008.

[11] A. Marquina and S. Osher, "Image super-resolution by tv-regularization and bregman iteration," in *Journal Of Scientific Computing*, 2008, vol. 37, no.3, pp. 367–382.