Depth Map Coding using Graph Based Transform and Transform Domain Sparsification

Gene Cheung *1, Woo-Shik Kim #2, Antonio Ortega #3, Junichi Ishida $4, Akira Kubota $5

* *Digital Content and Media Sciences Research Division, National Institute of Informatics*
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 180-0022, Japan
1 cheung@nii.ac.jp

# *Department of Electrical Engineering, University of Southern California*
3740 McClintock Ave., Los Angeles, CA 90089
2 wooshikk@usc.edu, 3 antonio.ortega@sipi.usc.edu

$ *Department of Electrical, Electronic and Communication Engineering, Chuo University*
1-13-27 Kasuga, Bunkyo-ku, Tokyo, 112-8551, Japan
4 j.ishida@kubotalab.jp, 5 kubota@elect.chuo-u.ac.jp

***Abstract*—Depth map compression is important for compact "texture-plus-depth" representation of a 3D scene, where texture and depth maps captured from multiple camera viewpoints are coded into the same format. Having received such format, the decoder can synthesize any novel intermediate view using texture and depth maps of two neighboring captured views via depth-image-based rendering (DIBR). In this paper, we combine two previously proposed depth map compression techniques that promote sparsity in the transform domain for coding gain—graph-based transform (GBT) and transform domain sparsification (TDS)—together under one unified optimization framework. The key to combining GBT and TDS is to adaptively select the simplest transform per block that leads to a sparse representation. For blocks without detected prominent edges, the synthesized view's distortion sensitivity to depth map errors is low, and TDS can effectively identify a sparse depth signal in fixed DCT domain within a large search space of good signals with small synthesized view distortion. For blocks with detected prominent edges, the synthesized view's distortion sensitivity to depth map errors is high, and the search space of good depth signals for TDS to find sparse representations in DCT domain is small. In this case, GBT is first performed on a graph defining all detected edges, so that filtering across edges is avoided, resulting in a sparsity count ρ in GBT. We then incrementally add the most important edge to an initial no-edge graph, each time performing TDS in the resulting GBT domain, until the same sparsity count ρ is achieved. Experimentation on two sets of multiview images showed gain of up to 0.7dB in PSNR in synthesized view quality compared to previous techniques that employ either GBT or TDS alone.**

I. INTRODUCTION

Continuing cost reduction of consumer-level cameras means images and videos previously taken by one camera from a single viewpoint can now be captured economically by an array of cameras to record multiple viewpoints. If, in addition to captured texture maps (RGB images), depth maps (per-pixel physical distances between camera and the location of the scene's objects) are also available[1], then novel intermediate views can also be synthesized via *depth-image-based-rendering* (DIBR) techniques such as 3D warping [1], using neighboring texture and depth maps as anchors. Having both captured and synthesized intermediate views available at the client can translate to richer visual experiences, such as free viewpoint TV [2]. However, encoding and transmitting texture and depth maps of a large number of captured views—a format known as *texture plus depth* [3]—can incur a high transmission cost. One practical necessity for texture-plus-depth format then, is efficient encoding of depth maps.

Recent efforts to encode depth maps [4], [5] exploit depth signal's unique characteristics, such as smooth surfaces and sharp edges, for efficient compression. In particular, *graph-based transform* (GBT) was proposed to code blocks in depth maps [5] that contain prominent edges: a graph specifying edges in the block is first defined, then subsequent adaptive transform on the graph is performed, so that filtering across edges, which often results in large non-zero high frequency coefficients, is avoided. Though specification of edges in blocks incurs a coding overhead, [5] shows that the resulting sparsity in the adaptively defined GBT can lead to coding gain.

A different approach towards sparse representations in the transform domain is *transform domain sparsification* (TDS) [6], [7]. In a nutshell, [7] proposed to first define per-pixel synthesized view's distortion sensitivity to errors in the depth map. For example, an edge pixel will have high sensitivity, since a small error in edge pixel will translate to confusion of foreground and background, resulting in large synthesized view distortion. Given defined per-pixel sensitivity, [7] then searched for a sparse depth signal in a given orthogonal transform domain away from ground truth depth signal for compression gain, at the cost of a controlled increase in synthesized view distortion. [7] proposed a computation-efficient optimization technique to optimally trade off transform domain sparsity with resulting synthesized view distortion.

In this paper, we combine these two depth map compression techniques—GBT and TDS—together under one unified opti-

[1]Depth maps can be estimated from texture maps, or captured explicitly using time-of-flight cameras.

mization framework, so the appropriate combinations of GBT and TDS are applied adaptively for different blocks. Note that optimally combining GBT and TDS, each of which achieves transform domain sparsity with a very different approach, is non-trivial. For example, it is unclear that finding the best GBT first and then selecting a sparse representation for that transform will be optimal. Given the overhead required by GBT, it is possible that using a simpler transform (fewer edges) plus TDS will lead to overall better performance. Hence, the key to combining GBT and TDS in some optimal fashion is to *adaptively select the simplest transform per block that leads to a sparse representation*. By "simplest", we mean a block with as few defined edges as possible, thus resulting in a small coding overhead.

More specifically, for a block without detected prominent edges in the depth signal, per-pixel synthesized view distortion sensitivity to depth errors is in general low. Hence, there exists a relatively large search space of good depth signals with small synthesized view distortion for TDS to seek transform domain sparse representations. For such blocks, we employ TDS in fixed DCT domain—without coding overhead to describe the transform or any edges.

For blocks with detected prominent edges, synthesized view distortion sensitivity in edge pixels is high, and the search space of good signals for TDS to find sparse representations in DCT domain is more restricted. In this case, we first perform GBT on a graph describing all detected edges, resulting in a sparsity count ρ in GBT. Then, we order the edges in terms of importance, and incrementally add the next most important edge to an initial no-edge graph and perform TDS on the resulting GBT, until the same sparsity count ρ is achieved. Because the procedure often terminates before all detected edges are added, we achieve the same representation sparsity with a simpler transform (fewer edges) as compared to GBT only. Experimentation on two sets of multiview images showed gain of up to 0.7dB in Peak Signal-to-Noise Ratio (PSNR) in synthesized view quality compared to previous techniques that employ GBT or TDS alone.

The outline of the paper is as follows. We first overview related work in Section II. We then discuss the fundamentals of GBT and TDS in Section III and IV, respectively. We discuss how we combine these two techniques under the same optimization framework in Section V. Results and conclusion are presented in Section VI and VII, respectively.

II. Related Work

GBT was proposed in [5] to efficiently encode block in depth maps that contain complicated edges. GBT first specifies edges in a code block, so that subsequent transform avoids filtering across these specified edges. Doing so eliminates high-frequency coefficients, resulting in a sparse representation in the graph transform domain. Unlike edge-adaptive wavelet transforms [4], GBT is more suitable for block-based processing, and unlike directional transforms [8], GBT can handle more complicated edges such as the "L"- or "V"-shaped edges. (TDS) [6], [7] explicitly manipulates the depth signal (without causing severe synthesized view distortion) for compression gain, rather than preserving the original signal. [9] also presented a signal manipulation problem, where given the constraint that code blocks must fall within their assigned quantization bins of the compressed image, high frequency components across block boundaries are eliminated via projection on convex sets (POCS). [10] discussed *near-lossless image compression*, where any given pixel value can have an error of no more than $\pm v$ values during compression. TDS differs from these previous work in that TDS defines one penalty function per depth pixel to reflect the unique sensitivity of synthesized view distortion to the pixel value.

Our goal in this paper is to combine GBT and TDS together under one unified optimization framework to adaptively identify the best RD performing transform for each code block. We first overview the basics of GBT and TDS in order. We then present our optimization framework.

III. Graph Based Transform

While DCT has been widely used for block-based image and video compression, it is known that it does not code a block containing arbitrarily shaped edges efficiently. We introduced GBT in our previous work [5], where an edge-adaptive block transform is performed on a graph that defines edges in the block (no links are drawn between neighboring nodes (or pixels) across an edge). In this section, we describe how to construct the transform and apply it to a signal. Refer to [5] for detailed properties and analysis of the transform.

The transform construction procedure consists of three steps: (i) edge detection in the original block[2], (ii) generation of a graph from pixels in the block using the edge map, and (iii) construction of transform matrix from the graph.

In the first step, after the intra/inter prediction, edges are detected in a residual block based on the difference between the neighboring residual pixel values. A simple thresholding technique is used in our previous work [5] to generate the binary edge map. Then, the edge map is compressed and included into a bitstream, so that the same transform matrix can be constructed at the decoder side.

In the second step, each pixel position is regarded as a node in a graph, G, and neighboring nodes are connected either by 4-connectivity or 8-connectivity, unless there is edge between them. From the graph, the adjacency matrix $\mathbf{A}$ is formed, where $\mathbf{A}(i,j) = \mathbf{A}(j,i) = 1$ if pixel positions i and j are immediate neighbors not separated by an edge. Otherwise $\mathbf{A}(i,j) = \mathbf{A}(j,i) = 0$. The adjacency matrix is then used to compute the degree matrix $\mathbf{D}$, where $\mathbf{D}(i,i)$ equals the number of non-zero entries in the i-th row of $\mathbf{A}$, and $\mathbf{D}(i,j) = 0$ for all $i \neq j$.

In the third step, from the adjacency matrix and the degree matrix, the Laplacian matrix is computed as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [11]. Then, projecting the signal of a graph G onto the eigenvectors

[2]If differential coding like intra prediction in H.264 is deployed, then the residual block can be used instead, as done in [5].

of the Laplacian $\mathbf{L}$ yields a spectral decomposition of the signal; i.e., it provides a "frequency domain" interpretation of the signal on the graph. Thus, the transform matrix can be constructed from the eigenvectors of the Laplacian of the graph. Since the Laplacian $\mathbf{L}$ is symmetric, the eigenvector matrix $\mathbf{E}$ can be efficiently computed using the well-known cyclic Jacobi method [12], and its transpose, $\mathbf{E}^t$, is taken as GBT matrix.

Transform coefficients are computed as follows. For an $N \times N$ block of residual pixels, form a one-dimensional input vector $\mathbf{x}$ by concatenating the columns of the block together into a single $N^2 \times 1$ dimensional vector, i.e., $\mathbf{x}(Nj+i) = X(i,j)$ for all $i,j = 0,1,...,N-1$. The GBT coefficients are then given by $\mathbf{y} = \mathbf{E}^t \cdot \mathbf{x}$, where $\mathbf{y}$ is also an $N^2 \times 1$ dimensional vector. The coefficients are quantized with a uniform scalar quantizer followed by entropy coding.

To get the best performance, DCT and GBT can be used in a hybrid manner. For example, for each block the rate-distortion (RD) is calculated for DCT and GBT, respectively, and the best one is selected. The transform mode is encoded into the bitstream for each block, and the edge map is provided only for blocks coded using GBT.

IV. Sparse Representation

The objective of transform domain sparsification (TDS) [6], [7] is to search for a depth signal $\mathbf{s}$ in the pixel domain, replacing the original ground truth signal $\mathbf{s}^o$, such that its transform domain representation is sparse (for compression gain) without creating large adverse effects in synthesized view distortion. These two quantities—transform domain sparsity and resulting synthesized view distortion—must be optimally traded off for best possible RD performance. We first formally define transform domain sparsity, then discuss how we capture synthesized view distortion sensitivity to depth signal on a per-pixel basis using quadratic penalty functions. Finally, we discuss the computation-efficient optimization we use to trade off these quantities optimally.

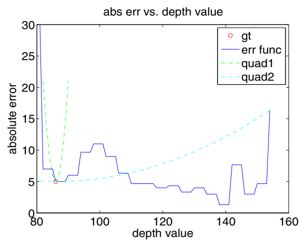
A. *Transform Domain Sparsity*

In general terms, an orthogonal transform coder maps a signal $\mathbf{s} \in \mathcal{R}^N$ to a set of N pre-defined basis functions ϕ_i's spanning the same signal space $\mathcal{R}^N$ of dimension N. In other words, a given signal $\mathbf{s}$ in $\mathcal{R}^N$ can be written as a linear sum of those basis functions using coefficients α_i's:

$$\mathbf{s} = \sum_{i=1}^{N} \alpha_i \phi_i \tag{1}$$

Only non-zero quantized versions $\hat{\alpha}_i$'s of transform coefficients α_i's are encoded and transmitted to the receiver for reconstruction of approximate signal $\hat{\mathbf{s}}$. Transform coefficients α_i's are obtained using a complementary set of basis functions $\bar{\phi}_i$'s; i.e., $\alpha_i = \sum_{i=1}^{N} s_i \bar{\phi}_i$.

Since only non-zero quantized transform coefficients are coded, the fewer the number of non-zero entries in α, the better the compression performance. The number of non-zero entries in α is equivalent to the l_0-norm $\|\alpha\|_{l_0}$ of α:



(a) Quadratic penalty function

(b) Penalty function curvature

Fig. 1. Construction of quadratic penalty function for one pixel, and penalty function curvature (parameter a) for view 6 of Teddy.

$$\|\alpha\|_{l_0} = |\{i|\alpha_i > 0\}| \tag{2}$$

Minimizing (2) would mean maximizing the signal's transform domain sparsity, leading to a more compact representation and compression gain.

B. *Quadratic Penalty Function*

We first assume a setup where two horizontally shifted cameras capture left and right views. Corresponding to each camera, both texture and depth maps are captured. We consider the encoding of left and right depth maps only in this paper.

A pixel $I_l(m,n)$ in the left texture map, where m is the pixel row and n is the pixel column, can be mapped to a correspondingly shifted pixel $I_r(m, n - D_l(m,n) * \gamma)$ in the right texture map, where $D_l(m,n)$ is the disparity value of the corresponding pixel in the left depth map, and γ is the shift scaling factor for this particular camera arrangement. To derive the synthesized view distortion sensitivity to pixel $D_l(m,n)$ of the left depth map, we define error function $E_l(e;m,n)$ given depth error e: it is the difference in texture pixel values between left pixel $I_l(m,n)$, and incorrectly mapped right pixel $I_r(m, n - (D_l(m,n)+e) * \gamma)$ due to error e. We write:

$$E_l(e;m,n) = |I_l(m,n) - I_r\left(m, n - (D_l(m,n)+e) * \gamma\right)| \tag{3}$$

Error function $E_r(e;m,n)$ for the right depth map can be derived similarly. As an example, the blue curve in Fig. 1(a) is the resulting $E_r(e;m,n)$ for the right view (view 6) of multiview image set Teddy [13]. One can see that as the depth value deviates from the ground truth disparity value $D_r(m,n)$ (red circle), the error increases in general.

For computation efficiency of later discussed optimization, we fit a per-pixel quadratic penalty function $g_i(s_i)$ to the error function and use $g_i(s_i)$ instead:

$$g_i(s_i) = (1/2)a_i s_i^2 + b_i s_i + c_i \tag{4}$$

where s_i is the disparity value corresponding to pixel location i, and a_i, b_i and c_i are the quadratic function parameters. The procedure we use to fit $g_i(s_i)$ to the error function is as follows. Given threshold τ, we first seek the nearest disparity $D_l(m,n) - e$ value below ground truth $D_l(m,n)$ that results in error $E_l(-e;m,n)$ exceeding $\tau + E_l(0;m,n)$. Using only two data points at $D_l(m,n) - e$ and $D_l(m,n)$, and assuming

$g_i(s_i)$ has minimum at ground truth depth value $D_l(m,n)$, we can construct one quadratic function. Similar procedure is applied to construct another penalty function using two data points at $D_l(m,n)+e$ and $D_l(m,n)$ instead. The sharper of the two constructed functions (larger a) is the chosen penalty function for this pixel.

Continuing with our earlier example, we see in Fig. 1(a) that two quadratic functions (in dashed lines) with minimum at ground truth depth value are constructed. The narrower of the two is chosen as the penalty function. In Fig. 1(b), the per-pixel curvature (parameter a) of the penalty functions of the right depth depth of `Teddy` is shown. We can clearly see that larger curvatures (larger penalties) occur at object boundaries, agreeing with our intuition that edge pixels have higher synthesized view's distortion sensitivity to depth map errors.

C. Iterative Quadratic Programming

To maximize sparsity in the transform domain without incurring a large penalty in synthesized view distortion, we define the following objective function:

$$\min_{\alpha} \quad \|\alpha\|_{l_0} + \lambda \sum_i g_i(\phi_i^{-1}\alpha) \tag{5}$$

where ϕ_i^{-1} is the i-th row of the inverse transform Φ^{-1}, and λ is weight parameter to trade off transform domain sparsity and resulting synthesized view distortion.

Because the l_0-norm is combinatorial and non-convex, (5) is difficult to solve efficiently. Instead, we leverage on previous work on sparse signal recovery [14] and replace the l_0-norm in (5) with a weighted l_2-norm instead:

$$\min_{\alpha} \quad \sum_i w_i\alpha_i^2 + \lambda \sum_i g_i(\phi_i^{-1}\alpha) \tag{6}$$

For a *fixed* set of weights w_i's, (6) can be efficiently solved as a quadratic program [15]. More specifically, we can rewrite (6) into standard quadratic programming form:

$$\min_{\alpha} \quad (1/2)\alpha^T \mathbf{P}\alpha + \mathbf{q}^T\alpha + r \tag{7}$$

where the constants $\mathbf{P}$, $\mathbf{q}$ and r are:

$$\begin{aligned} \mathbf{P} &= \begin{bmatrix} 2w_1 & 0 & \dots & \\ 0 & 2w_2 & 0 & \dots \\ 0 & \dots & \ddots & \end{bmatrix} + \lambda \sum_i a_i \left(\phi_i^{-1}\right)^T \phi_i^{-1} \\ \mathbf{q}^T &= \lambda \sum_i b_i \phi_i^{-1} \qquad r \;=\; \lambda \sum_i c_i \end{aligned} \tag{8}$$

The optimal solution to (7) can be easily and efficiently found by solving a set of linear equations [15]:

$$\mathbf{P}\alpha^o = -\mathbf{q} \tag{9}$$

The challenge is how to choose weights w_i's such that when (6) is solved iteratively, minimizing weighted l_2-norm equates to minimizing l_0-norm; i.e., select weights w_i's in each iteration so that the loop is sparsity-promoting. To accomplish this, we adopt the *iterative re-weighted least squares* (IRLS) work in [14] for our purpose. The key point is that after obtaining a solution α^o in one iteration, each weight w_i is assigned $1/|\alpha_i^o|^2$ if $|\alpha_i^o|$ is sufficiently larger than 0, so that contribution of the i-th non-zero coefficient $w_i|\alpha_i^o|^2$ is roughly 1. The algorithm is shown in Fig. 2. Quantization parameters Q_i's are considered in addition so that only quantized non-zero coefficients are counted. ϵ is needed for numerical stability when $|\alpha_i^o|$ is close to zero. Because each iteration in the iterative algorithm only requires solving a set of linear equations (9), and the number of iterations needed for convergence (found experimentally) is small, the algorithmic complexity is small. Detailed explanation is discussed in [7].

1) Initialize weights $w_i = 1/(|\alpha_i^t| + \epsilon)^2$, where α_i^t is the i-th transform coefficient of the ground truth depth signal $\mathbf{s}^t$.
2) Find optimal α^o to (6) for weighted l_2-norm minimization with penalties by solving (9).
3) Set each weight w_i to $(|\alpha_i^o|^2 + \epsilon^2)^{-1}$ if $|\frac{\alpha_i^o}{Q_i}| \geq 0.5$, and ϵ^2 otherwise.
4) Repeat Step 2 to 3 until convergence in quantized coefficients $\hat{\alpha}_i^o$'s.

Fig. 2. Iterative algorithm to solve weighted l_2-norm minimization with penalty functions $g_i(s_i)$'s.

V. Optimization Framework

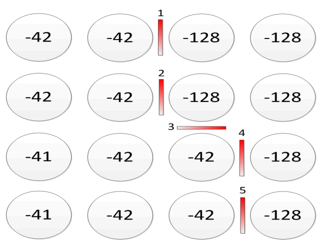
While the goal of both GBT and TDS is to find a transform domain sparse representation of the depth signal for compression gain, their approaches are quite different. GBT identifies a transform that naturally leads to a sparse representation, and TDS actively searches for a sparse representation (within some well defined search space) for a given orthogonal transform. Adaptively finding the best transform per block that leads to the best coding performance, assuming both GBT and TDS are used, is the technical challenge. In this section, we discuss how we identify appropriate transforms and apply combinations of GBT and TDS judiciously in an optimization framework.

A. Detection of Prominent Edges

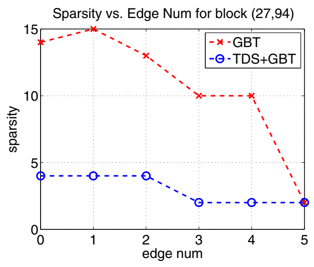
Unlike typical edge detection in computer vision literature for semantic-related tasks like object segmentation or recognition, the goal here is strictly for coding; i.e., identification of "edges" that cannot be efficiently coded using non-adaptive transforms. For simplicity, we first detect prominent edges in depth maps based on the absolute difference between neighboring pixels; i.e., check if the difference exceeds a threshold θ [5]. Depending on whether there are detected prominent edges in a block or not, we perform different operations to select the appropriate transform for this block, as described in Section V-B and V-C, respectively. Note again that for the sole purpose of coding, even when there are many detected prominent edges in a block, we are not obligated to code *all* of them for optimal coding performance.

B. Blocks without Prominent Edges

We first consider the easier of the two cases: blocks without detected prominent edges. As one can see in Fig. 1(b), the



(a) Quadratic penalty function



(b) Penalty function curvature

Fig. 3. Edges for block $(27, 94)$ in the right depth map (view 6) of `Teddy`, and sparsity versus number of edges specified for GBT.

synthesized view distortion sensitivity to depth signal errors is highest around object boundaries, and lowest in an object interior. That means blocks without edges have flatter quadratic penalty functions, and hence TDS can have a larger search space of signals with small synthesized view distortion to find sparse representations, even for non-edge-adaptive transforms like DCT. And because DCT is a fixed transform, there is no coding overhead to convey to the decoder the construction of the particular transform used for this block (beyond the 1 bit needed to inform decoder that the transform used is DCT). For this case then, we simply apply TDS on DCT to sparsify the depth signal in DCT domain.

C. Blocks with Prominent Edges

In the case when there are detected prominent edges, identifying the appropriate transform is more challenging. On one hand, as more edges are specified in a graph, the representation in the associated GBT is more likely to be sparse, since there are fewer occurrences of filtering across edges. This is demonstrated by the red curve in Fig. 3(b), where the number of non-zero quantized coefficients (denoted as *sparsity count*) decreases as the number of specified edges increases for this example block $(27, 94)$ of the right depth map of `Teddy`, as shown in Fig. 3(a). On the other hand, the overhead in encoding edges for the adaptive transform increases (roughly linearly) as more edges are specified.

To identify a good transform at low computation complexity, we perform the following simple procedure, so that the *same* sparsity count achieved by GBT-only can be accomplished using a graph transform with *fewer* specified edges, by utilizing TDS. In details, we do the following:

1) Given detected edges using threshold θ, construct a graph using *all* detected edges and perform GBT on the block. This results in sparsity count ρ.
2) Rank the importance of edges based on difference of depth pixel values across the edges. Initialize a no-edge graph (nodes representing neighboring pixels are all connected by links).
3) Add the next most important edge to the graph (remove the link between nodes representing neighboring pixels crossing the edge).
4) For given graph, sparsify the graph transform representation using TDS. If sparsity count is $> \rho$, go to step 3.

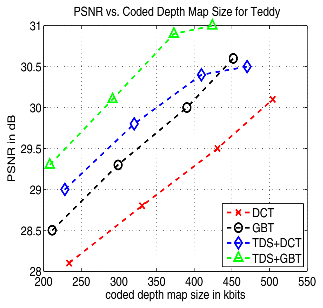
Because TDS finds sparser representations easier as more edges are added, the procedure tends to terminate with sparsity count $\leq \rho$ before all the detected edges are added, resulting in a net-positive coding gain. Moreover, ρ is a sparsity count that is demonstrably achievable, since it was the pre-set value for GBT-only. This means the procedure is guaranteed to exit, in the worst case when all the detected edges are specified. Continuing with our earlier example, we see in the blue curve in Fig. 3(b) that when TDS is used, the same sparsity count $\rho = 2$ is achieved when only three of the five detected edges are specified.

VI. Experimentation

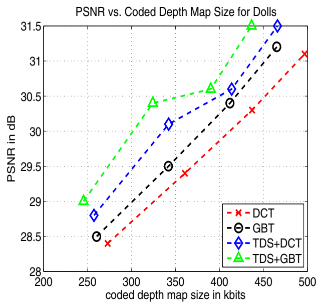
The proposed method is evaluated using multiview image sets `Teddy` and `Dolls` [13], where the ground truth disparity maps for the left and the right view are compressed, and the decoded disparity maps are used for synthesis of the middle view between the left and right views. The original intensity image is used for the view synthesis. Even though disparity maps are used for the experiments, similar performance is expected when the proposed method is applied to depth map coding considering the relationship between them [16]. The view synthesis is performed by warping the left and right views to the target view position (middle view), then blending process is applied if more than one pixels are mapped to the same position, which is weighted averaging using the distance from the reference view to the target view. For example, the weight is half if the middle view is synthesized between the left and right views. When there is no pixel mapped to a position, hole filling process is applied for this position by copying the horizontally nearest neighboring pixel value.

To compare the performance of the proposed method, which is the combination of TDS and GBT (TDS+GBT), various conventional methods are applied such as DCT, GBT, and TDS+DCT to compress the disparity maps. A block size of 4×4 is used for the transforms, followed by uniform quantization and CABAC as entropy coding. We used the integer transform in H.264/AVC as DCT, and the reference software JM 17.1 is used for the experiments. The same software is used to implement GBT and the proposed method. Note that for exploration purpose neither intra nor inter prediction is performed in our experiments. The inter-view prediction is not applied either.

Fig. 4 shows the RD curves generated using various coding methods such as DCT, GBT, TDS+DCT, and the proposed method, TDS+GBT. Fixed quantization parameter (QP) values of 24, 28, 32, and 36 were used to code left and right disparity maps. The original captured middle view is used for computation of PSNR. While both GBT and TDS+DCT performed better than DCT, TDS+GBT provided additional coding gain for both multiview image sets, by up to 0.7 dB and 0.6 dB in PSNR for `Teddy` and `Dolls`, respectively. In terms of BD-PSNR [17], the gains are 0.6 dB and 0.3 dB for `Teddy` and `Dolls`, respectively, as shown in Table I. The shape of the RD curves of TDS+GBT looks less straight compared to DCT or GBT, which is inherited from TDS+DCT. One reason for this is the same λ value in (5) is applied for all QP values for TDS and TDS+GBT: 0.05 for `Teddy` and 0.5



(a) Teddy



(b) Dolls

Fig. 4. Rate-distortion curves of DCT, GBT, TDS+DCT, and the proposed method, TDS+GBT. x-axis: total bits required to code two disparity maps; y-axis: PSNR of luminance component between the rendered view and the ground truth.

TABLE I
BD-PSNR (dB) and BD-bitrate (BDBR, %) results of GBT, TDS, and the proposed methods, GBT+TDS, compared to DCT.

Method	Teddy		Dolls	
	BD-PSNR	BDBR	BD-PSNR	BDBR
GBT	0.7	-24	0.3	-7
TDS+DCT	1.0	-31	0.9	-19
TDS+GBT	1.6	-34	1.2	-32

for Dolls, respectively. It is expected that better performance can be achieved by adapting λ to QP value as the Lagrange multiplier can be adapted to QP value for optimization [18].

Fig. 5 shows the subjective quality comparison between the synthesized view using DCT coded disparity map (left) and the one with the proposed method, TDS+GBT (right). In the image of DCT, it can be easily noticed that there is distortion along the foreground object boundary. Using the TDS+GBT, the distortion along the object boundary area is reduced. Both results are generated using QP value of 24, where the bitrate of TDS is 16% lower than that of DCT.

VII. Conclusion

In this paper, we present a unified optimization framework to combine two depth map compression techniques—graph-based transform (GBT) and transform domain sparsification (TDS) for optimal coding performance. The key is to adaptively select the simplest transform per block that leads to a sparse representation. Experimentation shows that gain up to 0.7dB in PSNR in synthesized view quality can be observed compared to previous techniques that employ either GBT or TDS alone.

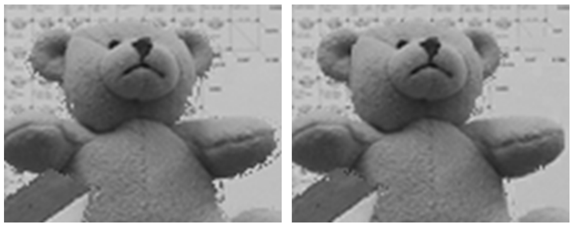


Fig. 5. Subjective quality comparison of synthesized view of Teddy using DCT (left) and the proposed method, TDS+GBT (right).

References

[1] Y. Morvan, D. Farin, and P. H. N. de With, “Multiview depth-image compression using an extended H.264 encoder,” in *Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Sciences*, vol. 4678, 2007, pp. 675–686.
[2] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, “Multi-view imaging and 3DTV,” in *IEEE Signal Processing Magazine*, vol. 24, no.6, November 2007.
[3] 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.
[4] 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.
[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] G. Cheung, A. Kubota, and A. Ortega, “Sparse representation of depth maps for efficient transform coding,” in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.
[7] G. Cheung, J. Ishida, A. Kubota, and A. Ortega, “Transform domain sparsification of depth maps using iterative quadratic programming,” in *(accepted to) IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011.
[8] B. Zeng and J. Fu, “Directional discrete cosine transforms for image coding,” in *IEEE International Conference on Multimedia and Expo*, Toronto, Canada, June 2006.
[9] R. Rosenholtz and A. Zakhor, “Iterative procedures for reduction of blocking effects in transform image coding,” in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no.1, March 1992, pp. 91–95.
[10] R. Ansari, N. Memon, and E. Ceran, “Near-lossless image compression techniques,” in *Journal of Electronic Imaging*, vol. 7, no.3, July 1998.
[11] D. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” in *Elsevier: Appplied and Computational Harmonic Analysis*, vol. 30, April 2010, pp. 129–150.
[12] H. Rutishauser, “The jacobi method for real symmetric matrices,” *Numerische Mathematik*, vol. 9, no. 1, November 1966.
[13] “2006 stereo datasets,” http://vision.middlebury.edu/stereo/data/scenes2006/.
[14] I. Daubechies, R. Devore, M. Fornasier, and S. Gunturk, “Iteratively re-weighted least squares minimization for sparse recovery,” in *Commun. Pure Appl. Math*, 2009.
[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, 2004.
[16] Video, “Report on experimental framework for 3D video coding,” ISO/IEC JTC1/SC29/WG11, Document N11631, October 2010.
[17] G. Bjøntegaard, “Calculation of average PSNR differences between RD-curves,” ITU-T SG.16 Q.6, Document VCEG-M33, April 2001.
[18] 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.