SPARSE DYADIC MODE FOR DEPTH MAP COMPRESSION

Shujie Liu*, PoLin Lai†, Dong Tian†, Cristina Gomila†, Chang Wen Chen*

*Dept. of Computer Science and Eng., State Univ. of New York at Buffalo, NY 14260
†Technicolor Research & Innovation, 2 Independence Way, Princeton, NJ 08540

ABSTRACT

In order to enable new video applications such as 3DTV and free-viewpoint video, new data formats including both 2D video sequences and corresponding depth map sequences have been proposed. One major characteristic making the depth maps different from video frames is that they typically consist of homogeneous areas separated by sharp edges representing depth discontinuities. Another characteristic of depth map sequences is that the edges exhibit quite similar boundary behaviors as the edges in the corresponding video frames. In this paper, we propose a novel sparse dyadic mode in the design of an efficient depth map compression algorithm through appropriately exploiting these characteristics. With sparse representations of depth blocks and effective reference of edge information from the corresponding video frames, sparse dyadic mode can achieve up to 1.5 dB gain on rendering quality as compared to depth sequences coded using MVC at the same bitrate.

Index Terms— Depth Compression, 3D Video, Sparse Representations, Sparse Dyadic Mode

1. INTRODUCTION

With recent development of three dimensional display technologies and interactive multiview applications, 3D video (3DV) has attracted significant interest both in industries and academics. As a collection of signals that can provide depth perception of a given scene, 3DV has many applications, including 3D Television (3DTV) and Free Viewpoint Video (FVV). With FVV, viewers are able to choose any viewpoint in a 3D space for a given scene [1], while 3DTV can display stereo video (two views) or multiple views simultaneously for 3D perception [2].

To enable 3DV applications, supplemental information of the 3D scene has to be transmitted together with classical 2D video, such as video captured from different viewpoints and the corresponding depth signal of a given 2D video. With video sequences and depth map sequences, virtual views can be generated using Depth Image Based Rendering techniques [3]. Since depth map sequences are assumed to be transmitted with the corresponding video sequences, compression of depth maps needs to be investigated in 3DV coding.

Typically, a depth map consists of depth samples with a scaled 8 bit depth value, each corresponding to a pixel in the video frame. Depth map sequence can be regarded as a grayscale video sequence. The most straightforward approach to compress depth map sequences is to encode them using conventional image/video compression algorithms. However, depth map exhibits unique characteristics compared to conventional video frame: 1) it contains homogeneous areas separated by sharp edges, with very limited texture; 2) it usually has similar boundary information as the corresponding video frame; 3) instead of being displayed directly, depth map is used in view rendering, where the accuracy of boundaries is crucial in terms of rendering quality.

Based on these characteristics, several approaches have been proposed for depth compression. The first category includes methods to encode depth map as independent images without temporal prediction. They include platelet-based coding [4] and shape adaptive wavelet based coding [5]. Based on a quadtree partition algorithm, platelet-based coding describes different areas with three piecewise linear functions and different block sizes. Because of the complex rate-distortion optimization process in function selection and quadtree decomposition, this approach results in very high computational complexity. Shape adaptive wavelet coding [5] is proposed to jointly encode the edge information shared by depth map and color image only once, while encode the texture in color image and depth map separately.

Another category of depth coding methods attempts to exploit temporal correlation, similar to the techniques for coding video sequences. In [6], the authors proposed to utilize motion information of the corresponding video for encoding depth map sequences. In this approach, candidate coding modes and motion vectors (MVs) are generated based on the video bitstream. The mode decision and motion estimation for depth map are then performed based on these candidate coding modes and MVs. It was proposed not to send the bits for motion information because it may take up to 40% of bit budget at low bitrate. In our work, instead of using the motion information from the video, we utilize video pixel information in encoding depth maps.

Besides approaches for depth map compression, there are also methods for preserving edges in video compression. For example, a geometry adaptive intra mode [7] was proposed to partition a video block into two parts by choosing a partition boundary from a wide range of directions and positions. It might not achieve high efficiency if used directly in depth compression, since depth maps usually have similar boundary information to the corresponding video frames, which could be utilized to achieve efficient depth map compression.

In this paper, a novel coding mode, Sparse Dyadic Mode, is proposed to encode depth map sequences. Simple yet efficient representations are utilized to encode blocks of depth samples, leading to much less side information than the function parameters of platelet-based coding [4] and the partition boundary parameters in geometry adaptive mode [7]. By appropriately referencing edge information from the corresponding video frame, high fidelity depth edges can be reconstructed under the Sparse Dyadic Mode.

The remainder of this paper is organized as follows. Section 2 describes the block representations of Sparse Dyadic Mode. Section 3 provides in detail the coding process of the proposed Sparse Dyadic Mode. Experimental results are shown in Section 4 to confirm the performance of the proposed scheme. Section 5 concludes this paper with a summary.

This work is supported by Technicolor Research & Innovation.
2. SPARSE DYADIC MODE: THE REPRESENTATIONS

To achieve sufficient quality in the rendered views, it is shown that the good depth boundaries have to be well preserved [8]. Conventional video coding techniques typically result in large artifacts around sharp edges. Faithful representation of the depth edges would cost significantly more bits than coding other regions.

However, corresponding video frames can be utilized to recover details such as edges in depth maps; it may be desired not to encode depth maps with unnecessarily high fidelity. Moreover, depth map usually has limited texture information. Therefore, it is advantageous to represent a block in the depth map coarsely and recover the depth map by combining edge information from the coarse depth map representations with corresponding edge information from corresponding video frames. We introduce in this paper a novel coding mode, Sparse Dyadic (SD) Mode, for the compression of depth maps. We shall demonstrate that this simplified representation in SD Mode, with proper referencing of video frame, is sufficient to recover high fidelity depth map at the decoding end with improved rendering quality.

As a new coding mode, the name sparse dyadic reflects the fact that an M×N block is represented by only two areas, separated by a straight line, with one or two representative depth values. There are five candidate partitions as shown in Figure 1, where MODE_FLAT means the block is treated as part of flat area and represented by only one value. For the other four partition modes, all depth sample values in area Pa are set as A, while all depth sample values in area Pb are set as B. The generated block is called SD block, and the selected partition is indicated using SD_partition_mode.

![Figure 1 Candidate SD partition modes](image)

We propose to apply SD Mode to the following macroblock (MB) partition sizes similar to H.264 /AVC: 16×16, 16×8, 8×16 and 8×8, denoted as MB_depth_mode, as shown in Figure 2. For example, each of the 16×8 blocks in MODE_16×8 can be represented with one of the candidate SD partition modes as shown in Figure 1.

![Figure 2 Depth modes to which SD partitions can be applied](image)

Therefore, an MB coded by SD Mode is represented by three (or four) parameters and the residue information for this MB. Among the set of parameters, MB_depth_mode indicates the dyadic block size, SD_partition_mode indicates which one of the five partitions shown in Figure 1 is used, A (or B) represents corresponding representative depth value for Pa (or Pb) shown in Figure 1. The corresponding MB consisting of SD block(s) is called SD MB.

As a comparison, the platelet method in [4] will need to transmit more side information: the quadtree decomposition, the platelet boundary which can take on several possible locations and directions, the coefficients of different platelet functions, and the quantization size for the coefficients. As for the geometry-adaptive intra mode in [7], their partition boundary can also take from a wide range of directions and positions, leading to more signaling bits as compared to only 5 partition modes as in Figure 1.

While simple representations are used in SD Modes, the depth block predictor with detailed edge will be constructed using the corresponding video block, and the representative values A and B have to be determined. Then, the residue between depth block predictor and the original depth block is encoded. In Section 3, we will explain each of steps for depth coding with SD Modes.

3. SPARSE DYADIC MODE: THE CODING PROCESS

There are mainly four steps in SD Mode encoding: 1. Block refinement that refines the SD blocks using video frame to obtain detailed depth blocks; 2. Depth representation estimation that determines the depth representative values at encoder side; 3. Depth representative prediction that performs predictive coding in order to encode depth representatives more efficiently; and 4. Trilateral filter [9] as an in-loop filter that removes coding artifacts for SD blocks. All these encoding steps, as well as the decoding process, will be described in detail in following subsections.

3.1. Refinement Process for SD Mode

From the simple SD partition, this step intends to reconstruct detailed depth block by taking the corresponding MB in video frame as reference. The main idea is to recover the boundary similar to that of video MB. The generated depth reconstruction is called Refined SD block.

![Figure 3 Refinement process with video block](image)

As an example shown in Figure 3, for an SD block with specified partition areas Pa and Pb (SD_partition_mode = MODE_VER), the refinement process will generate corresponding refined block with two refined partitions Pa’ and Pb’, with the utilization of information from corresponding video frame block T. The approach is described as follows:

Step 1. For a given depth sample at location p, find collocated sample Iq in video frame block T.

Step 2. Within block T, find all samples Iq that satisfies |I_q - I_p| < th, where th is a threshold chosen by the user.

Step 3. The set of all sample locations found in previous step form a similarity area in depth block for depth sample p, indicated as the gray area Sp in Figure 3.

Step 4. If Sp has larger overlapping area with Pa than with Pb, assign sample p to the partition Pa’, else assign to the partition Pb’.

Step 5. Repeat this process for samples in the current SD block.

It can be observed that, in Step 3, the similarity among video samples is used to refine the SD partitions. The refined partition Pa’ will consist of all depth samples with their similarity area overlapping with Pa larger than with Pb.
Note that the refinement process is performed using only the SD partition and the similarity among video pixels. The depth representative values A and B are not required. In fact, they have not yet been determined. In the next subsection, we will describe how to compute the representative values using the refined partitions Pa’ and Pb’.

### 3.2. Depth Representative Estimation

The depth representatives A and B are generated based on the depth samples within the refined partitions Pa’ and Pb’ respectively. Different methods can be used to calculate the depth representations, for example, by taking the mean value of all the samples within the refined partition, or by choosing the sample value with the most appearance. In this paper, we select the depth representatives which will lead to minimum SAD for all samples $S_i$ within the corresponding partitions:

$$A = \arg \min_{S_a \in Pa'} \sum_{S_i \in Pa'} |S_i - E_a|, \quad B = \arg \min_{S_b \in Pb'} \sum_{S_i \in Pb'} |S_i - E_b|$$  \hspace{1cm} (1)

In fact, from the basic principle in optimization, the value A (or B) that satisfies Equation (1) is the median of all samples $S_i$ within $Pa’$ (or $Pb’$).

### 3.3. Depth Representative Prediction

In order to encode depth representatives (value A and B) in SD partitions efficiently, we perform predictive coding, rather than directly encoding them. The predictors can be derived from neighboring blocks, such that only the differences between the predictors and the depth representatives are coded.

In this paper, spatial correlation is used, where the depth representation predictors of current encoding block are generated from neighboring depth samples, either within the same MB or from neighboring MBs.

As shown in Figure 4, for a given block with refined partitions Pa’ and Pb’ obtained as in Section 3.1, the corresponding neighboring depth samples on L1, L2 and L3 will be used to generate predictors. First, samples on L1 and L2 are populated into two sets Sa and Sb, based on the refined partitions to which their neighboring samples in current block belong, as indicated in Figure 4(a). If all samples on L1 and L2 belong to one set, for example in Figure 4(b) where no sample in L1 and L2 belongs to Sb, samples along L3 will be further checked and assigned to the empty set Sb if corresponding samples with 45 degree angle in the block belongs to Pb’.

**Figure 4** Prediction for depth representatives

After generating Sa and Sb, median value of each set is calculated as the representative value predictor of the corresponding partition. If one of the set is empty, the two predictors are all set as the median value of the non-empty set.

The differences between the predictors and their corresponding depth representative values obtained in Section 3.2 will be encoded and transmitted in the bitstream.

### 3.4. Trilateral Filtering

In order to remove the coding artifacts for SD blocks, edge-preserving trilateral filtering [9] is applied as an in-loop filter. As indicated in Equation (2), for a given position $p$ in depth map, the filtered output $S'_p$ is a weighted average of reconstructed neighboring depth samples $S_q$ at position $q$ (within a range $\Omega$). The weights are determined by three filters: a domain filter $f$ based on the distance between $p$ and $q$, a range filter $g'$ based on the difference between collocated sample values in video pixels $I_p$ and $I_q$, and another range filter $g''$ based on the difference between depth sample values $S_p$ and $S_q$. $K_p$ is the normalization factor as specified in Equation (3).

$$S'_p = \frac{1}{K_p} \sum_{q \in \Omega} f(p-q) g'(I_p-I_q) g''(S_p-S_q)$$  \hspace{1cm} (2)

where $K_p = \sum_{q \in \Omega} f(p-q) g'(I_p-I_q) g''(S_p-S_q)$  \hspace{1cm} (3)

In this work, the domain filter $f$ selects a window of $\pm 5$ neighboring depth samples, with equal weights on them. Within this window, the two range filters will identify pixels satisfying $|S_q-S_p| < \delta$ and $|I_q-I_p| < \delta$, where $\delta$ was set to 10. The filtered result $S'_p$ will be the average of these identified depth samples.

Note that in this paper, the trilateral filtering is only applied to SD MBs as the in-loop filter, while for other blocks the de-blocking filter in H.264/AVC is applied.

### 3.5 Decoding Process for SD MBs

Figure 5 shows the decoding process for an SD MB. After parsing / decoding information in the bitstream, refined partitions for each SD block are generated as in Section 3.1 and the corresponding depth representative predictors are calculated as in Section 3.3. Reconstructed blocks are obtained by adding the representative value differences to the predictors, and then adding the block residue. Finally, trilateral filtering in Section 3.4 is used instead of de-blocking filter as the in-loop filter for SD MBs.

**Figure 5** Decoding Process for SD MB

### 4. SIMULATIONS

#### 4.1. Simulation Setup

The proposed SD mode is incorporated into MVC reference software JMVC [10]. Test sequences of Ballet and Breakdancers are used with resolution of 1024x768 [11]. For each test sequence, we encode video and the depth map sequences of view 3 and view 4 with inter-view prediction from view 3 to view 5. Total 97 frames for each sequence are encoded with GOP size 8 and hierarchical B structure. Depth map sequences are encoded with 4 different basic QPs: 22, 27, 32 and 37 while the corresponding video sequences are encoded with a fixed basic QP 22. After finishing decoding, virtual video of view 4 is generated by VSRS...
3.0 (View Synthesis Reference Software) provided by MPEG
Error! Reference source not found.

4.2. Simulation Results

Figure 6 and Figure 7 provide the Rate Distortion (R-D) curves for depth coding of Ballet and Breakdancers Error! Reference source not found. where “MVC” indicates JMVC encoder, and “MVC + SD” indicates adding SD Mode as an additional mode to MVC. In the two R-D curves, the rate indicates the total bitrate of the compressed depth map sequences used for view synthesis (i.e. compressed depth map sequences of view 3 and view 5 when rendering view 4). Since the depth map sequences are used for rendering instead of being viewed directly, in Figure 6 and Figure 7, we compute the PSNR between the rendered view using compressed video/depth sequences and the rendered view using uncompressed video/depth sequences. From the simulation results, it can be seen that by adding the proposed SD Mode as an additional mode, we can achieve up to 1.5 dB gain (ballet with depth basic QP 37) for rendered video quality with same depth encoding bitrate. For both sequences, the gain is larger in lower bitrate scenarios. For depth with basic QP 22, a gain of more than 0.6 dB is observed.

Furthermore, we also collect the statistics of the mode selection results. For depth with basic QP 37, about 5% of blocks in B frames are coded by SD Mode, with a majority of them located along boundaries; while for depth with basic QP 22, about 15% blocks in B frames are coded by SD Mode, including blocks located along both boundaries and in flat areas.

From both the coding performance and mode selection results, it can be seen that the proposed SD Mode is able to compress depth information efficiently with much better synthesized results, especially in lower bitrate scenarios.

5. CONCLUSIONS

In this paper, a novel depth coding mode, Sparse Dyadic Mode, is proposed for coding depth maps. Instead of spending lots of bits in coding precise depth edges, we utilize simplified representations for depth blocks, which require very little side information. By referencing detailed edges information from the corresponding video frame, this mode is able to compress depth more efficiently, especially for blocks containing depth edges and also homogeneous blocks. Simulation results demonstrate that, by adding the proposed SD Mode as an addition mode to MVC to encode depth map sequences, we can achieve up to 1.5 dB gain in rendering quality as compared to MVC at same bitrate.

6. REFERENCES