Depth Sequence Coding with Hierarchical Partitioning and Spatial-domain Quantisation

Shampa Shahriyar, Manzur Murshed, Mortuza Ali, and Manoranjan Paul

***Abstract*—Depth coding in 3D-HEVC for the multiview video plus depth (MVD) architecture (i) deforms object shapes due to block-level edge-approximation; (ii) misses an opportunity for high compressibility at near-lossless quality by failing to exploit strong homogeneity (clustering tendency) in depth syntax, motion vector components, and residuals at frame-level; and (iii) restricts interactivity and limits responsiveness of independent use of depth information for "non-viewing" applications due to texture-depth coding dependency. This paper presents a standalone depth sequence coder, which operates in the lossless to near-lossless quality range while compressing depth data superior to lossy 3D-HEVC. It preserves edges implicitly by limiting quantisation to the spatial-domain and exploits clustering tendency efficiently at frame-level with a novel binary tree based decomposition (BTBD) technique. For mono-view coding of standard MVD test sequences, on average, (i) lossless BTBD achieved $\times 42.2$ compression-ratio and $-60.0\,\%$ coding gain against the pseudo-lossless 3D-HEVC, using the lowest quantisation parameter $QP = 1$, and (ii) near-lossless BTBD achieved $-79.4\,\%$ and $6.98\,\mathrm{dB}$ Bjøntegaard delta bitrate (BD-BR) and distortion (BD-PSNR), respectively, against 3D-HEVC. In view-synthesis applications, decoded depth maps from BTBD rendered superior quality synthetic-views, compared to 3D-HEVC, with $-18.9\,\%$ depth BD-BR and $0.43\,\mathrm{dB}$ synthetic-texture BD-PSNR on average.**

***Index Terms*—Depth map sequence coding, hierarchical partitioning, lossless/near-lossless coding, multiview extension of High Efficiency Video Coding (3D-HEVC), multiview video plus depth (MVD), spatial-domain quantisation.**

I. Introduction

MULTIVIEW video is an emerging trend in the development of interactive digital video systems to offer immersive viewing experiences. Acquisition and transmission of multiview video content is constraint by the available storage and bandwidth. The bitrate requirement for transmitting encoded multiview video content increases almost linearly with the number of views (aka viewpoints). To circumvent this only a limited number of views, spaced evenly in terms of the viewing-angle, are encoded and virtual-views for intermediary viewpoints are synthesised on-demand from the adjacent encoded-views using the depth image based rendering (DIBR) techniques [1]. The essential idea of DIBR is to extrapolate the texture, a collective term referring to the luminance (intensity) and chrominance (colour) values of pixels, in a virtual-view from the texture and associated depth map, representing the distance of pixels from the corresponding points on the surfaces of objects in the 3-dimensional (3D) scene, in the encoded-views that are adjacent to the virtual-view.

Compared to texture, depth map typically exhibits distinct structural properties with higher homogeneity and sharp edges at object boundaries. Consequently, traditional video coding techniques render ineffective to compress depth map sequences at the high ratio expected from exploiting homogeneity. To address this, 3D-HEVC [2], [3], the multiview extension of the latest High Efficiency Video Coding (HEVC) standard, has adopted the *multiview video plus depth* (MVD) architecture. In contrast to video (texture) coding, 3D-HEVC encodes depth map sequences with fewer and additional coding paths/modes to exploit the spatio-temporal correlations at the intra-view, inter-view, and inter-domain (texture-depth) levels.

Depth map compression techniques [4]–[6] adopted in 3D-HEVC, in general, exploit smooth-regions at block-level. A coding-unit block, which partially covers two or more regions, is partitioned into homogeneous segments by approximating (i) the boundaries with wedgelets or contours and (ii) the segments with constant partition values (CPVs). However, the potential compression efficiency from exploiting higher homogeneity in depth maps is too great to extract at block-level. Not only is an opportunity to encode smooth-regions as-a-whole at frame-level missed but noises are also introduced at the sharp boundaries, the most-sensitive components of depth maps. Consequently, depth map coding in 3D-HEVC suffers from the following three shortcomings that curb the potential use of depth data in much wider applications.

Firstly, the lossy techniques [4]–[6] obscure sharp edges due to edge-approximation, spatial- to frequency-domain transformations, and aggressive quantisations [7]. As a result, noticeable shape deformations are introduced at object boundaries in the synthesised views rendered from the decoded depth maps of 3D-HEVC. Consequently, overall view-synthesis quality is compromised.

The proposed motion-compensated depth map sequence coder preserves edges *inherently* by limiting quantisation to the spatial-domain with a small scalar step size on the pixel-level residual values. Moreover, the decoded depth maps are near-lossless. On standard MVD test sequences, it has retained very high quality with average peak signal-to-noise ratio (PSNR) of $52.2\,\mathrm{dB}$ for the recommended maximum quantisation step size $Q = 15$. Retaining such high quality in the decoded depth

S. Shahriyar was with the Faculty of Information Technology, Monash University, Victoria, Australia (e-mail: shampa077@gmail.com).

M. Murshed is and M. Ali was with the Faculty of Science and Technology, Federation University Australia, Churchill Vic 3842, Australia (e-mail: manzur.murshed@federation.edu.au; mortuza94@gmail.com).

M. Paul is with the School of Computing and Mathematics, Charles Sturt University, Bathurst NSW 2795, Australia (e-mail: mpaul@csu.edu.au).

This research is supported by the Australian Research Council Discovery Project DP130103670.

maps may be explained as follows. In audio and image coding, prediction residuals, rounded to the nearest integers, tend to follow two-sided geometric (TSG) distribution [8], modelled as $\Pr(r=k)=((1-p)/(1+p))^{|k|}p$, $k\in\mathbb{Z}$, where parameter p is the proportion of zeros. For any arbitrary quantisation step size $Q=2D+1$, the mean squared error (MSE) is estimated as

$$\text{MSE}=2p\sum_{k=1}^{D}k^2\sum_{i=0}^{\infty}\Big(\frac{1-p}{1+p}\Big)^{i(2D+1)+(-1)^i k} \tag{1}$$

by considering the symmetry in squared errors about the y-axis. Depth residuals typically have very high proportion of zeros e.g., $p\in\{0.8,0.9\}$. For $Q=15$, the MSE is typically expected to be in the range 0.117–0.282 with PSNR $20\log_{10}(255/\sqrt{\text{MSE}})$ in the range $52.63\,\text{dB}$ to $57.44\,\text{dB}$.

Secondly, quality of decoded depth maps by 3D-HEVC, even at low quantisation, is not acceptable for many "non-viewing" applications such as auto-navigation, night vision imagery, object tracking, action recognition, and many computer vision applications.

Efficient near-lossless (or even lossless) compression of depth maps can provide an attractive alternative to lossy coding, since little (or no) distortion is introduced in the synthesised views by the depth map coder. For natural images (texture), lossless coding typically attains compression ratio of merely $\times2$ to $\times3$ that prohibits any bitrate-sensitive applications. A much higher compression ratio is achievable for lossless coding of depth maps by exploiting the prevalent high level of spatio-temporal homogeneity. At lossless mode, the proposed coder has achieved average compression ratio of $\times42.2$ on multiview standard test sequences. The compression ratio at near-lossless mode, which further applies modest spatial-domain quantisation, is so high (on average $\times608.8$ while introducing negligible distortion at PSNR $52.2\,\text{dB}$) that the need for any lossy depth map sequence coder is no longer justified.

Achieving such high compression ratio for depth map sequences without incurring significant distortions may be explained as follows. Motion-compensated depth-residuals are mostly low values, or simply zeros, due to the prevalent high temporal correlation in successive depth frames, except in the moving regions where noticeable variance in depth may be observed. Intra-predicted depth-residuals also exhibit dominant low values due to the high level of spatial correlation in depth values from the same object in the 3D scene. On top, the low dynamic range in typical depth signals makes sure that a large proportion of a depth residual frame has very low magnitude values after applying modest spatial-domain quantisation. Ultimately, high *clustering tendency* is exhibited in depth residual frames as well as the corresponding syntax (coding tree unit divisions and coding unit modes) and motion (x- and y-components) frames where regions with highly skewed distribution of values emerge. This unique property is exploited by the proposed coder with a novel *hierarchical partitioning* technique for frame-level data maps to achieve significant compression efficiency gain by encoding each partition separately using *arithmetic coding*, which is particularly effective on very skewed probability distribution.

Thirdly, the texture-depth coding dependency in 3D-HEVC restricts interactivity and limits responsiveness of independent use of depth information for non-viewing applications as a depth map cannot be decoded without decoding the corresponding texture frame. Moreover, the inter-view coding path, where depth maps in already-encoded views are used as references to compress the depth map of another view, curbs random access functionality and restricts interactivity.

The paper develops an independent (standalone) depth map sequence coder, which can operate both at lossless and near-lossless mode by appropriately setting the frame-level scalar quantisation step to zero and a small positive value, respectively. High clustering tendency in the frame-level data maps are exploited efficiently with a hierarchical partitioning technique, namely *binary tree based decomposition* (BTBD). Using a code-length estimator of arithmetic coding, BTBD splits a data map recursively at a hyperplane orthogonal to one of the axes using a greedy optimization heuristic. Effectively, it divides the data maps into relatively-homogeneous cuboids of arbitrary sizes that are encoded independently with overall compression efficiency significantly higher than that without partitioning. As the level of distortion in a decoded depth frame depends solely on the quantisation step size used for the corresponding residual frame, the rate-distortion optimisation (RDO) is simplified to consider only the code-length of different coding modes. Being independent from texture coding, the proposed depth sequence coder is highly responsive (fast random access functionality) and flexible (can be used with coders of other modalities) to offer wider applications of depth data.

Effectiveness of spatial-domain quantisation in inherently preserving edges in depth maps was first demonstrated in [7], [9] where 3D-HEVC depth residuals were quantised at pixel-level and each frame is encoded in whole with lossless JPEG (JPEG-LS) [10], the lossless compression scheme developed by the Joint Photographic Expert Group (JPEG), to exploit long 1D-runs of very low values. The preliminary concept of BTBD was first introduced in [11], [12] to exploit the prevailing clustering tendency in 3D-HEVC depth coding modes (2D-runs) and depth motion vector components (3D-runs). Full potential of these novel ideas, however, cannot be realised within the 3D-HEVC framework. Firstly, its edge-approximation coding modes contradict the philosophy of inherent edge preservation. Secondly, spatial-domain quantisation renders its RDO ineffective or, at best, sub-optimal. Finally, the it has no provision for lossless depth map coding.

This paper presents a comprehensive depth map sequence coder, independent from the 3D-HEVC framework, with the following key contributions: (i) comparative analysis of clustering tendency in motion-compensated texture and depth residual frames; (ii) in-depth analysis of achievable depth map quality with spatial-domain quantisation; (iii) rank-based signed residual mapping; (iv) RDO-based coding tree unit (CTU) division and coding unit (CU) mode selection; (v) formation of 2D/3D data maps to exploit clustering tendency in depth syntax, motion, and residual frames; (vi) enhanced context modelling for context-adaptive arithmetic coding (CAAC) of BTBD partitions; and (vii) comprehensive performance

analysis of BTBD at lossless and near-lossless modes. Some preliminary results of this work, limited to only lossless mode with a fixed size CUs (8×8 pixel) and approximated bitrate (using estimated code-length of arithmetic coding), have been published recently in [13].

Rest of the paper is organised as follows. In Section II, state-of-the-art depth map coding techniques are reviewed. Clustering tendency in depth maps is analysed in Section III to provide motivation of the proposed hierarchical partitioning based coding scheme, which is elaborated in Section IV. Section V presents simulation results with analyses and Section VI concludes the paper.

II. Related Work

Both lossy and lossless depth map coding schemes have been proposed in the literature.

A. Lossy coding of depth maps

Depth map coding schemes proposed in the literature can be broadly categorized into three classes: segmentation, block partitioning, and edge-adaptive transformation (EAT) based approaches.

1) Segmentation based approach: The schemes proposed in [14]–[17] explored the idea of explicitly identifying object boundaries from depth maps using segmentation techniques. These schemes then aim at separately encoding the object boundaries followed by compression of the smooth regions. In this approach, the overhead of explicitly coding the contours outweighs the gain achieved by efficient compression of the smooth regions, affecting the overall coding efficiency. Another important limitations of this approach is that the explicit segmentation process is computationally expensive.

2) Block partitioning based approach: Alternative schemes that avoid explicit object segmentation have also been proposed in the literature [18]–[20]. The essential idea is to represent a frame using a quad-tree partition where object boundaries at leaf nodes are modeled using simpler geometric primitives. Morvan *et al.* proposed platelet-based depth coding in [18], [19]. It assumes that a leaf node in the quad tree is piecewise planar and thus can be modelled with two regions of constant gradient separated by a straight line. Thus, the scheme uses piecewise-linear functions (platelets) to approximate the edges in the blocks. While this scheme partitions and models the original depth maps, the scheme proposed in [20] models the residual, resulting from intra-prediction, using platelets. This scheme was later extended by Merkle *et al.* in [4]. It proposed using wedglet, instead of platelets, to model the residual, which has been adopted in the latest 3D-HEVC standard [2], [5].

Zamarin *et al.* [6] introduced a new intra-mode, specifically targeted to depth macroblocks with arbitrarily shaped edges. The scheme also partitions the edge-macroblock into two regions, each approximated by a flat surface. Considering the edge structure of previously encoded macroblocks as the context, the scheme proposed context based encoding of the edge information. To obviate the necessity of transmitting the edge information, Kang *et al.* [21] proposed partitioning a macroblock using the edge estimated from the geometric structure of the previously encoded neighboring blocks.

These block-based depth coding techniques mostly use intra-coding that cannot take the advantage of high temporal correlations in successive depth maps and spatial correlations in inter-coded depth residuals. Besides, their handling of edges by approximating them with piecewise linear functions essentially distorts the shape of the object.

By assuming significant statistical correlation between texture and depth maps, many techniques tried to reuse block partitioning information for depth coding from coded texture images by recognizing the structural similarity between the texture image and the corresponding depth map [5], [22], [23]. Merkle *et al.* [4] have also introduced contour based depth edge modelling using information from the corresponding texture image. The assumption that an edge in texture image is reflected on the corresponding depth map does not always hold in practice. If an object has textured patterns, abrupt discontinuity of colour values appear inside as well as at the object boundary in the texture image. However, the corresponding depth map exhibits discontinuity of depth values only at the boundary as the inner part of the object is represented by a constant depth value.

There are also several methods [24]–[27] that take the advantage of shared motion information (generated from block based motion search [28]) between texture and depth data for coding efficiency. Due to lighting condition changes, motion compensation are not similar for texture and depth. While searching for a best matching block, texture motion search can result in long motion vectors and large residual values. On the other hand, corresponding depth block is more similar to co-located or neighbouring blocks in the reference frame. Thus, reuse of texture motion vectors is not guaranteed to achieve optimal rate-distortion performance [7].

3) EAT based approach: Maitre and Do [29] proposed a depth map compression scheme based on a shape-adaptive wavelet transform by generating small wavelet coefficients along depth edges. Shen *et al.* [30] proposed a set of EATs to replace the standard discrete cosine transform (DCT). This block based scheme essentially (i) detects the edges in a block; (ii) constructs a graph based on the edge map of the block, and (iii) computes an EAT for the graph that minimizes a number of non-zero coefficients that must be encoded for the block. Motivation for this approach stems from the fact that for piecewise smooth signals e.g., depth maps, EAT would yield sparser representation.

Compressed sensing (CS) is an emerging area in signal processing, which suggests that any sparse signal can be reconstructed from the sub-samples using efficient recovery algorithms. Recently, depth map coding schemes [31]–[33] have been proposed under the CS framework. By assuming that image blocks are sparser in the pixel domain than in the residual domain, Do *et al.* [32] and Duan *et al.* [31] proposed CS method based on down-sampling of the 2D-DCT coefficients. However, the choice of DCT as sparsifying basis is inefficient for blocks containing arbitrarily shaped edges. Therefore, a CS based scheme using EAT as sparsifying basis has been proposed in [33].

One of the shortcomings of the EAT based approaches is that they require explicit edge detection and lossless transmission of the edge map to the decoder so that the decoder can also construct the same EAT. Besides, the process of constructing the EAT is computationally expensive.

B. Lossless coding of depth maps

Very few works on efficient lossless compression of depth maps have so far been proposed in the literature. Kim *et al.* proposed a bit-plane based scheme in [34]. The scheme proposed to decompose the original frame into several bit planes by first transforming the depth values using gray code. Then, the decomposed bit planes were encoded independently and in-order, starting from the MSB bit plane to the LSB. The encoding process is thus repeated eight times. Zamarin and Forchhammer [35] later modified this scheme by changing the prediction template (inter and intra) and extending the prediction into view level.

Since CABAC (context-adaptive binary arithmetic coding) [36] was originally designed for lossy texture coding, it was unable to provide the optimum coding performance for lossless depth map coding. In [37], Heo and Ho proposed an improved version of the arithmetic encoder of H.264/AVC using significance map targeting depth map encoding. The technique, however, used only fixed $4 \times 4\,\text{pixel}$ blocks.

III. Clustering Tendency in Depth Maps

In this section, a comparative analysis of clustering tendency in motion-compensated texture and depth residual frames is briefly presented. Let I_{texture} be the intensity map (represented with a 8-bit grayscale image) of a texture frame and I_{depth} be the corresponding depth map (represented with another 8-bit grayscale image) in an MVD sequence. Let $\hat{I}$ be the motion-compensated predicted image, $\Delta I = I - \hat{I}$ be the corresponding residual frame of signed integers, and $\Delta\mathbb{I}$ be the corresponding absolute residual frame of residual magnitudes for a grayscale image I.

Fig. 1 presents typical $\Delta\mathbb{I}_{\text{texture}}$ and $\Delta\mathbb{I}_{\text{depth}}$, both lossless and with spatial-domain quantisation by scalar step $Q = 3$, for an arbitrary frame 23 in view 1 of *UndoDancer* multiview standard test sequence. The following characteristically unique observations can be made on clustering tendency in depth residuals.

Firstly, due to high spatio-temporal correlations in depth maps, almost $90\,\%$ values in $\Delta\mathbb{I}_{\text{depth}}$ are zeros, compared to only $20\,\%$ to $30\,\%$ in $\Delta\mathbb{I}_{\text{texture}}$. Secondly, dominant zeros are clustered densely in depth residuals; whereas non-dominant zeros are spread thinly across texture residuals. Thirdly, when quantisation is applied, the clustering tendency in depth residuals is intensified. Although, zeros become dominant (around $60\,\%$) in texture residuals, the clustering tendency is still not that prominent. Only when the quantisation level is increased, clustering in smaller values are introduced for the high motion sequence. Finally, the dynamic range in $\Delta\mathbb{I}_{\text{depth}}$ is significantly low, compared to $\Delta\mathbb{I}_{\text{texture}}$. In both lossless and quantised domains, the dynamic range in the former is 30–40 lower.

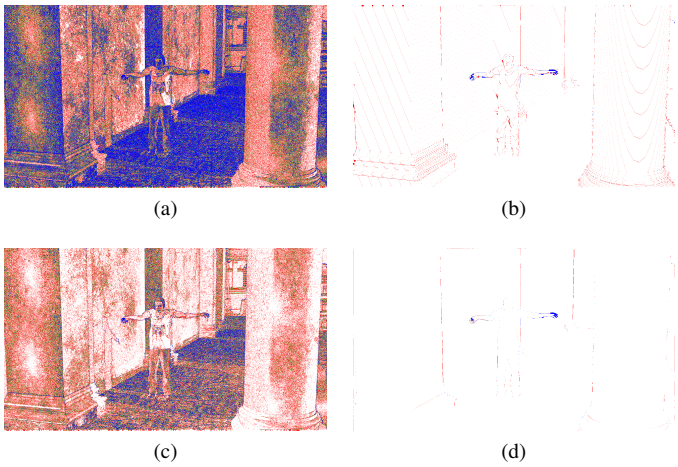


Fig. 1. Magnitude of residuals for (a) lossless intensity; (b) lossless depth; (c) quantised intensity; and (d) quantised depth maps of *UndoDancer* sequence (view 1, frame 23), where white, red, green, and blue represent magnitudes 0, 1, 2, and $\geq$3, respectively, and $Q = 3$ is used for quantisation. (best viewed in colour)

Similarly strong clustering tendency is evident in CTU divisions (Fig. 4c), CU prediction modes (Fig. 4c), and motion components (Fig. 4a) of a depth map. Such high clustering tendency in depth residual frames leads to localised regions of skewed probability distribution. This may be exploited by arithmetic coding, along with the low dynamic range, to achieve significant compression efficiency after isolating the regions with any density-based partitioning.

IV. Proposed Depth Map Sequence Coder

In this section, an independent depth map sequence coder is proposed based on the above concept. It decides quadtree based CTU divisions (similar to 3D-HEVC) and the prediction modes of CUs (of size $64 \times 64\,\text{pixel}$, $32 \times 32\,\text{pixel}$, $16 \times 16\,\text{pixel}$, and $8 \times 8\,\text{pixel}$) for a particular frame based on a rate-distortion optimisation scheme (Section IV-A). The CU-level signed motion-compensated depth residuals are first quantised, if near-lossless mode is used, and then mapped to unsigned integers using a generic ranking-based residual mapping technique (Section IV-B). Depth syntax and mapped-residual data are arranged into a number of frame-level bitmaps or integer maps (intmaps) (Section IV-C). The clustering-correlations in these maps are exploited using BTBD partitioning (Section IV-D), where large homogeneous blocks with similar values are isolated and then encoded using context-adaptive arithmetic coding. The leaf nodes generated from map partitioning are used as the coding blocks. Unlike 3D-HEVC, coding block size is independent of CU size and adaptive to clustering-correlations. Finally, these coding blocks are encoded using CAAC (Section IV-E).

A. CTU division and prediction mode selection

Let $B(I, r, c, m)$ denote a CU of size $m \times m\,\text{pixel}$ ($m = 2^{6-k}$ and $0 \leq k \leq 3$) in depth frame I at pixel coordinate (r, c), i.e., the CU represents $I(r, \ldots, r+m-1; c, \ldots, c+m-1)$ values of the frame covering all pixel coordinates (i, j)'s

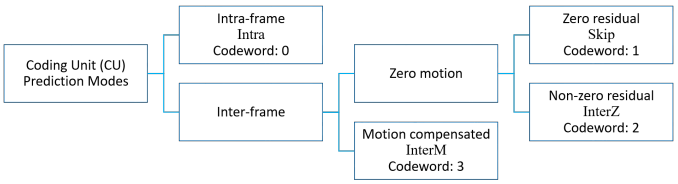


Fig. 2. Prediction modes of depth map coding units.

such that $i \in \{r, \ldots, r+m-1\}$ and $j \in \{c, \ldots, c+m-1\}$. Each frame I of size $H \times W$ pixel is divided into $\frac{H}{m}$ rows and $\frac{W}{m}$ columns of non-overlapping coding units such that the CU at row r_m and column c_m refers to $B(I, (r_m-1)m+1, (c_m-1)m+1, m)$, for all $1 \leq r_m \leq \frac{H}{m}$ and $1 \leq c_m \leq \frac{W}{m}$.

In inter prediction coding, each of the $\frac{H}{m} \times \frac{W}{m}$ CUs in the current frame I_t is predicted from the already encoded reference frame I_{t-1} using block-based motion search. Let $MV_m(p, r_m, c_m)$ denote the motion vector (MV) components of CU at row r_m and column c_m where $p \in \{1 \equiv x\text{-component}, 2 \equiv y\text{-component}\}$. For intra prediction coding, each of the $\frac{H}{m} \times \frac{W}{m}$ CUs in the current frame I_t is predicted from already coded similar neighbourhood CUs in I_t (using CALIC's gradient adjusted predictor [38]).

For a fixed coding unit size $m \times m$ pixel, let $\hat{I}^A_{t_m}$ and $\Delta I^A_{t_m} = I_t - \hat{I}^A_{t_m}$ denote the predicted and residual frame, respectively, such that for each CU $B(I, (r_m-1)m+1, (c_m-1)m+1, m)$, the co-located blocks $B(\hat{I}^A_{t_m}, (r_m-1)m+1, (c_m-1)m+1, m)$ and $B(\Delta I^A_{t_m}, (r_m-1)m+1, (c_m-1)m+1, m)$ represent its prediction and residual, respectively, for all $1 \leq r_m \leq \frac{H}{m}$, $1 \leq c_m \leq \frac{W}{m}$, and $A \in \{\text{inter}, \text{intra}\}$.

For each 64×64 pixel CTUs in frame I_t, division and mode selection decisions are made based on RDO. First, the optimal prediction mode of the CTU is decided and then, a quad-tree based sub-division into four CUs is recursively decided if further optimality can be achieved. Compared to 3D-HEVC, the RDO of the proposed coder is simple. The maximum pixel-level distortion D in a frame is controlled using a fixed scalar quantisation step $Q = 2D+1$. While operating at near-lossless mode ($Q > 1$), distortion depends solely on the scalar quantisation step Q, irrespective of prediction modes. Hence, the bitrate-optimal prediction mode for any CU $B(I, (r_m-1)m+1, (c_m-1)m+1, m)$ is decided by comparing the *estimated* bits needed to encode $B(\Delta I^{\text{intra}}_{t_m}, (r_m-1)m+1, (c_m-1)m+1, m)$ against the same for $B(\Delta I^{\text{inter}}_{t_m}, (r_m-1)m+1, (c_m-1)m+1, m)$ and the MV $(MV_m(1, r_m, c_m), MV_m(2, r_m, c_m))$. For $m > 2^3$, the CU is recursively divided into four $\frac{m}{2} \times \frac{m}{2}$ pixel CUs only if bits are saved. A code-length estimator (Section IV-D2) for encoding the residual block with CAAC, after mapping to unsigned integers (Section IV-B), is used. Bits of encoding each MV component is estimated $2\lceil \log_2(|MV_m(p, r_m, c_m)|+1)\rceil + 1$ bit, with signed Exp-Golomb codes [39], $p \in \{1, 2\}$.

Four different predictive modes are used as depicted in Fig. 2 with their hierarchical relationships. A CU is intra- or inter-predicted. If intra-predicted (Intra), only the residuals need to be encoded. Otherwise, there are two possibilities: (a) both MV components are zeros (zero motion), when the best match is found in the temporally co-located CU; or (b) at least one MV component is non-zero. In the second case (InterM), residuals as well as the MV need to be encoded for the motion-compensated CU. In the first case, there can be further two possibilities: (a) CU exactly matches the co-located CU (Skip) for which neither residual nor MV need encoding; or (b) otherwise (InterZ) for which only the residuals need to be encoded.

B. Residual mapping with quantisation

Consider predictive coding of a sequence $X = x_0, x_1, \ldots$ drawn from an integer-valued source. Let $\hat{x}_t$ be the integer-valued prediction for x_t, computed from the past samples $x_0, x_1, \ldots, x_{t-1}$, and $\epsilon_t = x_t - \hat{x}_t$ be its signed residual.

With an 8-bit grayscale image source, the domains of x and ϵ are $[0, R]$ and $[-R, R]$, respectively, where $R = 2^8 - 1$. For any integer-valued prediction $\hat{x}$, however, there can be at most $R+1$ distinct signed residual values, $V_{\hat{x}} = \{-\hat{x}, 1-\hat{x}, \ldots, R-\hat{x}\}$. It is, therefore, possible to establish a one-to-one correspondence (aka bijection) map between the signed residuals and the unsigned integer domain $[0, R]$ by using the rank r_ϵ of a signed residual ϵ in the corresponding $V_{\hat{x}}$ w.r.t. the magnitude as follows:

$$r_\epsilon = \begin{cases} 2\epsilon, & |\epsilon| \leq \min(\hat{x}, R-\hat{x}) \wedge \epsilon \geq 0; \\ -2\epsilon - 1, & |\epsilon| \leq \min(\hat{x}, R-\hat{x}) \wedge \epsilon < 0; \\ \min(\hat{x}, R-\hat{x}) + |\epsilon|, & |\epsilon| > \min(\hat{x}, R-\hat{x}). \end{cases} \tag{2}$$

This ranking based mapping is partly influenced by the Rice mapping [40]. Similar to the Rice mapping, starting from zero, the ranking technique maps successive -ve and +ve residuals to odd and even values, respectively, until one side is exhausted, then it uses linear mapping for the other side. In doing so, it keeps the dynamic range unchanged, whereas the Rice mapping can potentially double the dynamic range and hence degrade compression efficiency of arithmetic coding.

Spatial-domain quantisation guarantees that every reconstructed sample does not diverge from the original signal by more than a preset amount $D \geq 0$. The classical approach to near-lossless coding is to uniformly quantise the integer-valued residual ϵ into quantisation bins of size $Q = 2D+1$, with reproduction at the centre of the bins. Let $\epsilon_Q = \lceil \frac{\epsilon}{Q} \rfloor$ be the quantised-residual rounded to the nearest integer (denoted by the $\lceil \cdot \rfloor$ operator), with scalar quantisation step Q. Note that the inverse-quantised value $Q\epsilon_Q$ has quantisation noise $\epsilon - Q\epsilon_Q$ in the range $[-D, D]$ for all $Q > 1$. No quantisation takes place for $Q = 1$, which is used for coding at lossless mode.

The ranking based mapping can be easily extended to quantised-residuals. Note that the quantisation process effectively reduces the domain of residuals to $[-\lceil \frac{R}{Q} \rfloor, \lceil \frac{R}{Q} \rfloor]$. Hence, the rank r_{ϵ_Q} of ϵ_Q in the corresponding $V_{\lceil \frac{\hat{x}}{Q} \rfloor}$ w.r.t. the magnitude can be estimated by substituting ϵ, $\hat{x}$, and R in (2) with ϵ_Q, $\lceil \frac{\hat{x}}{Q} \rfloor$, and $\lceil \frac{R}{Q} \rfloor$, respectively.

C. Data map formation

To exploit strong clustering tendency in depth maps, CTU divisions, CU prediction modes, MV components, and

quantised-residual ranks of each frame I_t are arranged into a number of bitmaps/intmaps so that BTBD partitioning can be applied to isolate clusters before encoding.

1) CTU division bitmaps: Quad-tree based CTU division decisions of a frame is represented jointly with three bitmaps. For each of the first three division levels $l \in \{0,1,2\}$, where decisions on splitting a $2^{6-l} \times 2^{6-l}$ pixel CU into four $2^{5-l} \times 2^{5-l}$ pixel CUs are made, a bitmap $\mathcal{M}_{\text{div}_{2^{6-l}}}$ of size $\frac{H}{2^{6-l}} \times \frac{W}{2^{6-l}}$ is used, where each bit represents whether the corresponding $2^{6-l} \times 2^{6-l}$ pixel CU is divided further (1) or not (0). Note that no bitmap is needed for 8×8 pixel CUs as these are never split.

For the two lower levels, $l \in \{1,2\}$, bits corresponding to non-existent CUs, due to not-split decisions at a level above, are ignored. A virtual don't-care symbol $\times$ is used to identify the ignored CUs by setting $\mathcal{M}_{\text{div}_{2^{6-l}}}(r_{2^{6-l}}, c_{2^{6-l}}) = \times$ if $\exists k < l : \mathcal{M}_{\text{div}_{2^{6-k}}}(\lceil \frac{r_{2^{6-l}}}{2^{l-k}} \rceil, \lceil \frac{c_{2^{6-l}}}{2^{l-k}} \rceil) = 0$. The symbol $\times$ is considered virtual as it can always be deducted from other maps and it is never encoded.

In order to deal uniformly with variable CU sizes, a frame is considered as a $\frac{H}{8} \times \frac{W}{8}$ grid of non-overlapping CUs of the smallest size and a virtual significance bitmap $\mathcal{M}_{\times}$ of size $\frac{H}{8} \times \frac{W}{8}$ is used. The CTU at row r_{64} and column c_{64} is represented by the block of 8×8 bit in $\mathcal{M}_{\times}$ starting at coordinate $(8r_{64} - 7, 8c_{64} - 7)$. In that CTU, for each leaf (no further split) CU of size $2^{6-k} \times 2^{6-k}$ pixel, $k \in \{0,1,2,3\}$, only the top-left bit position of the corresponding $2^{3-k} \times 2^{3-k}$ bit in $\mathcal{M}_{\times}$ is set to 0 and the remaining bits are set to 1. Bitmap $\mathcal{M}_{\times}$ is considered virtual as it is never encoded; it just helps simplifying the process of defining the remaining data maps.

2) Mode intmap: Four CU prediction modes Intra, Skip, InterZ, and InterM are assigned code word 0, 1, 2, and 3, respectively. Prediction modes of all leaf CUs in a frame are represented with an intmap $\mathcal{M}_{\text{mode}}$ of size $\frac{H}{8} \times \frac{W}{8}$. For each bit $0 \in \mathcal{M}_{\times}$, the co-located integer in $\mathcal{M}_{\text{mode}}$ is set to the code word of the prediction mode used by the corresponding leaf CU. The remaining coordinates of $\mathcal{M}_{\text{mode}}$ are ignored, i.e., $\mathcal{M}_{\text{mode}}(r_8, c_8) = \times$ if $\mathcal{M}_{\times}(r_8, c_8) = 1$.

3) Zero/non-zero MV component bitmap: Depth MV components are predominantly zero-valued. For any leaf CU using a prediction mode other than InterM, the MV either does not exist (Intra) or it is fixed (Skip and InterZ) with both components zero-valued. There is no need to encode MV for these CUs.

A 3D bitmap $\mathcal{M}_{\overline{\text{MVZ}}}$ of size $2 \times \frac{H}{8} \times \frac{W}{8}$ is used to represent the zero/non-zero binary-classification of all encoded MV components in a frame. For each bit $0 \in \mathcal{M}_{\times}$, the co-located 2-bits in $\mathcal{M}_{\overline{\text{MVZ}}}$ are set to 0 (zero-valued) or 1 (non-zero-valued), based on two MV components of the corresponding leaf CU, if that CU is using 'Inter-MC' prediction mode. The remaining coordinates of $\mathcal{M}_{\overline{\text{MVZ}}}$ are ignored, i.e., $\mathcal{M}_{\overline{\text{MVZ}}}(p, r_8, c_8) = \times$ if $\mathcal{M}_{\times}(r_8, c_8) = 1$ or $\mathcal{M}_{\text{mode}}(r_8, c_8) \neq 3$ for all $p \in \{1,2\}$.

4) Residual intmap: For any leaf CU using prediction mode 'Skip', the residuals are fixed, all zero-valued, even after quantisation and ranking based mapping. There is no need to encode the residuals for these CUs.

An intmap $\mathcal{M}_{\text{res}}$ of size $H \times W$ is used to represent the pixel-level ranks of quantised-residuals that need encoding. For each bit $0 \in \mathcal{M}_{\times}(r_8, c_8)$, ranks of quantised-residuals of the corresponding leaf CU are stored in $\mathcal{M}_{\text{res}}$ starting at coordinate $(8r_8 - 7, 8c_8 - 7)$ if that CU is not using 'Skip' prediction mode. Otherwise, these coordinates in $\mathcal{M}_{\text{res}}$ are ignored.

D. Map partitioning with BTBD

The BTBD map partitioning uses a greedy heuristic on minimizing the estimated code-length of encoding partitions with CAAC.

1) Context modelling: Context $\mathbb{C}_{\mathcal{M}}(\cdot)$ of any coordinate $(\cdot)$ in a data map $\mathcal{M}$ is calculated using the values in adjacent coordinates along all dimensions. If any adjacent value is the don't-care symbol $\times$, it is replaced with 0; except for $\mathcal{M}_{\text{mode}}$ maps, where $\times$ is replaced with the code word of the corresponding leaf CU. For $\mathcal{M}_{\text{res}}$, the magnitude of the inverse-quantised residual $|Q\epsilon_Q|$ corresponding to the adjacent value r_{ϵ_Q} is used. Let $\mathcal{A}_d(\mathcal{M}(\cdot))$ denote the adjacent value along dimension d of coordinate $(\cdot)$ in $\mathcal{M}$ as defined above.

Among the six data maps $\mathcal{M}_{\text{div}_{64}}$, $\mathcal{M}_{\text{div}_{32}}$, $\mathcal{M}_{\text{div}_{16}}$, $\mathcal{M}_{\text{mode}}$, $\mathcal{M}_{\overline{\text{MVZ}}}$, and $\mathcal{M}_{\text{res}}$, the first five represent nominal (non-collatable) data and only the last one represents ordinal (collatable) data.

For a nominal map $\mathcal{M}$ of dimension $\mathcal{D} \in \{2,3\}$, all possible combinations of adjacent values are considered as different contexts that are mapped to unique integer values as

$$\mathbb{C}_{\mathcal{M}}(\cdot) = \sum_{d=1}^{\mathcal{D}} n^{d-1} \mathcal{A}_d(\mathcal{M}(\cdot)), \tag{3}$$

where $n = 2$ and 4 for bitmap and intmap, respectively. With nominal bitmaps, the domain of $\mathbb{C}_{\mathcal{M}}$ is $[0, 2^2 - 1]$ and $[0, 2^3 - 1]$ for 2D maps ($\mathcal{M}_{\text{div}_{64}}$, $\mathcal{M}_{\text{div}_{32}}$, and $\mathcal{M}_{\text{div}_{16}}$) and 3D map ($\mathcal{M}_{\overline{\text{MVZ}}}$), respectively. The same for the nominal intmap $\mathcal{M}_{\text{mode}}$ is $[0, 4^2 - 1]$.

For the ordinal map $\mathcal{M}_{\text{res}}$, the range of an adjacent value (magnitude of inverse-quantised residual) is $[0, 255]$. Collectively, the adjacent values along two dimensions can have 2^{16} combinations that are far too many to consider as different contexts. Nevertheless, the ordinality of residuals can be effectively used to define only four contexts by dividing the range $[0, 510]$ of the sum of adjacent residuals into four bins of geometrically increasing length as follows:

$$\mathbb{C}_{\mathcal{M}_{\text{res}}}(\cdot) = \begin{cases} 0, & 0 \leq \mathcal{A}_1(\mathcal{M}(\cdot)) + \mathcal{A}_2(\mathcal{M}(\cdot)) \leq 4; \\ 1, & 5 \leq \mathcal{A}_1(\mathcal{M}(\cdot)) + \mathcal{A}_2(\mathcal{M}(\cdot)) \leq 22; \\ 2, & 23 \leq \mathcal{A}_1(\mathcal{M}(\cdot)) + \mathcal{A}_2(\mathcal{M}(\cdot)) \leq 117; \\ 3, & 118 \leq \mathcal{A}_1(\mathcal{M}(\cdot)) + \mathcal{A}_2(\mathcal{M}(\cdot)) \leq 510, \end{cases} \tag{4}$$

Note that the specific bin edges in (4) are empirically obtained through sensitivity analysis on multiview test sequences.

2) Code-length estimation of CAAC: Let $\mathcal{M}$ be a bitmap/intmap having N non-ignored valued, drawn from the range $[0, R]$, where context-model $\mathbb{C}_{\mathcal{M}}$ has been used to divide them into k subsets $\mathbb{C}_{\mathcal{M},i}$'s, $0 \leq i < k$. Let $n_{i,j}$ be the count of value j in context $\mathbb{C}_{\mathcal{M},i}$ such that $N_i = \sum_{j=0}^{R} n_{i,j}$ and $\sum_{i=0}^{k-1} N_i = N$. The code-length $\mathcal{L}(\mathcal{M})$ of encoding $\mathcal{M}$ with

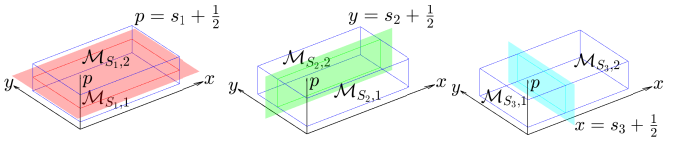


Fig. 3. Splitting a 3D map into halves with a plane orthogonal to (left) p-axis; (middle) y-axis; and (right) x-axis. (best viewed in colour)

CAAC can be estimated by adding the zero-order entropy $\mathcal{H}_0(\mathbb{C}_{\mathcal{M},i})$ and the associated model cost $\mu(\mathbb{C}_{\mathcal{M},i})$ of all k context as follows:

$$\begin{aligned}\mathcal{L}(\mathcal{M}) &= \left\lceil \sum_{i=0}^{k-1} N_i \mathcal{H}_0(\mathbb{C}_{\mathcal{M},i}) + \mu(\mathbb{C}_{\mathcal{M},i}) \right\rceil \\ &= \left\lceil -\sum_{i=0}^{k-1}\sum_{j=0}^{R} n_{i,j} \log_2 p_{i,j} + \mu(\mathbb{C}_{\mathcal{M},i}) \right\rceil \text{ bit},\end{aligned} \tag{5}$$

where $p_{i,j} = \frac{n_{i,j}}{N_i}$ is the static probability of symbol j in context i.

Model cost of arithmetic coding is the implicit overhead of encoding probability model parameters $p_{i,j}$'s with sufficient precision so that correct decoding is guaranteed. For sufficiently large N_i, it has been shown that each free (independent) parameter incurs $\frac{1}{2}\log_2 N_i$ bits of model cost [41]. Considering the density constraint $\sum_{j=0}^{R} p_{i,j} = 1$, each context $\mathbb{C}_{\mathcal{M},i}$ has at most R free parameters and hence,

$$\mu(\mathbb{C}_{\mathcal{M},i}) \approx \frac{R}{2}\log_2 N_i \text{ bit}, \tag{6}$$

for all $0 \le i < k$.

When N_i is not sufficiently large, the number of free parameters is estimated more accurately using the following empirically obtained correction:

$$\hat{R} = \frac{R}{2^{\frac{R+1}{\log_2 N_i}\left(2^{\frac{1-\log_2(R+1)}{2}} - \hat{p}_i\right)}}, \tag{7}$$

where $\hat{p}_i$ is the estimated parameter of the geometric probability density function that fits the distribution of residuals in $\mathbb{C}_{\mathcal{M},i}$ best. Parameter $\hat{p}_i$ is estimated from $n_{i,j}$'s by the method of moments [42] as

$$\hat{p}_i = \frac{(R+2)\sum_{j=0}^{R} n_{i,j} - 2\sum_{j=0}^{R} j n_{i,j}}{(R+1)\sum_{j=0}^{R} j n_{i,j} - \sum_{j=0}^{R} j^2 n_{i,j}}. \tag{8}$$

3) Binary tree based decomposition (BTBD): BTBD uses a greedy divide-and-conquer heuristic. The *divide* step tries to optimally split a bitmap/intmap into halves recursively so that encoding bits are saved. Consider a map $\mathcal{M}$ of size $\psi_1 \times \psi_2 \times \psi_3$ (representing its length in p-, y-, and x-axis, respectively) with $N \le \prod_{d=1}^{3} \psi_d$ non-ignored elements. If $\mathcal{M}$ contains only one symbol, it is classified as a Type I (all 0's) or Type II (all 1's for bitmap and all same non-zero (SNZ) values for intmap) leaf node. Otherwise, it is split recursively into two halves with a plane orthogonal to p-, y-, or x-axis (see Fig. 3), respectively classified as Type P, Type Y, or Type X split, such that the overall code-length is minimised.

Algorithm 1: BTBD ($\mathcal{M}$)

Input: Bitmap $\mathcal{M}$
Output: Binary partition-tree T

```
if M contains only one symbol (excluding x) then
    T ← ⟨Type I or II leaf node⟩;
else
    Find the optimal split s*_{d*};
    T_1 ← BTBD(M_{s*_{d*},1});
    T_2 ← BTBD(M_{s*_{d*},2});
    if L_{s*_{d*}}(M) < L(M) then
        T ← ⟨Type d* internal node, T_1, T_2⟩;
    else
        T ← ⟨Type III leaf node⟩;
    end if
end if
```

Let a split s_d of $\mathcal{M}$ along dimension $d \in \{1,2,3\}$ form two cuboid halves $\mathcal{M}_{s_d,1}$ and $\mathcal{M}_{s_d,2}$. If these cuboids are encoded independently, the overall code-length $\mathcal{L}_{s_d}$ due to split s_d can be measured as

$$\mathcal{L}_{s_d}(\mathcal{M}) = \mathcal{L}(\mathcal{M}_{s_d,1}) + \mathcal{L}(\mathcal{M}_{s_d,1}) + \mu_s(s_d), \tag{9}$$

where $\mu_s(s_d)$ is the split cost, representing the overhead of encoding the split dimension and position. Split dimension d is encoded with variable-length Huffman codes (see Table I). A split s_d actually represents a plane $p = s_1 + \frac{1}{2}$, $y = s_2 + \frac{1}{2}$, and $x = s_3 + \frac{1}{2}$ for $d \in \{1,2,3\}$, respectively. Along each dimension d, there are $\psi_d(\mathcal{M}) - 1$ possible splits denoted by the range $1 \le s_d < \psi_d(\mathcal{M})$. For bitmaps, all possible splits are considered and hence, s_d is encoded with fixed-length codes of $\log_2(\psi_d(\mathcal{M}) - 1)$ bits. For integer maps, only the halfway split $s_d = \lceil \frac{1}{2}(\psi_d(\mathcal{M}) - 1) \rceil$ along each dimension d is considered and hence, s_d requires no encoding.

The greedy heuristic finds the optimal split $s^*_{d^*}$ among all possible splits under consideration such that $\mathcal{L}_{s_d}(\mathcal{M})$ is minimised as follows:

$$s^*_d = \begin{cases} \underset{1 \le s_d < \psi_d(\mathcal{M})}{\arg\min} \ \mathcal{L}_{s_d}(\mathcal{M}), & \mathcal{M} \text{ is a bitmap;} \\ \lceil (\psi_d(\mathcal{M}) - 1)/2 \rceil, & \mathcal{M} \text{ is an intmap.} \end{cases} \tag{10}$$

$$d^* = \underset{d \in \{1,2,3\}}{\arg\min}\ \mathcal{L}_{s^*_d}(\mathcal{M}). \tag{11}$$

The *conquer* step accepts $s^*_{d^*}$ for partitioning if and only if bits are saved, i.e., $\mathcal{L}_{s^*_{d^*}}(\mathcal{M}) < \mathcal{L}(\mathcal{M})$. In that case, the the halves $\mathcal{M}_{s^*_{d^*},1}$ and $\mathcal{M}_{s^*_{d^*},2}$ are considered for further recursive splitting. Otherwise, it is classified as a Type III leaf node (mixed).

The greedy BTBD heuristic is formally presented in Algorithm 1.

E. Encoding and decoding

For each frame, the encoder first compresses data maps $\mathcal{M}_{\text{div}_{64}}$, $\mathcal{M}_{\text{div}_{32}}$, $\mathcal{M}_{\text{div}_{16}}$, $\mathcal{M}_{\text{mode}}$, $\mathcal{M}_{\overline{\text{MVZ}}}$, and $\mathcal{M}_{\text{res}}$ in order to preserve don't-care dependencies. Each data map is partitioned with BTBD and the associated partition-tree and leaf nodes are encoded with Huffman coding and CAAC, respectively. Finally, the encoder compresses the non-zero MV components.

TABLE I
HUFFMAN CODES OF PARTITION-TREE NODES

Node	Type	Description	Codeword	
			Bitmap	Intmap
Leaf	I	all 0's	00	001
	II	all same non-zero values	1000	000
	III	mixed values	101	01
Split	X	along x-axis	11	11
	Y	along y-axis	01	10
	P	along p-axis	1001	N/A

TABLE II
MULTIVIEW VIDEO PLUS DEPTH (MVD) TEST SEQUENCES

Sequence	Views			Frame size
	Cameras	Selected	Synthesized	Frame rate
1 *Balloons*	7	1, 3	1.5, 2, 2.5	
2 *Newspaper*	9	2, 4	2.5, 3, 3.5	1024×768p
3 *Lovebird1*	12	4, 6	4.5, 5, 5.5	30 fps
4 *Kendo*	7	1, 3	1.5, 2, 2.5	
5 *PoznanStreet*	9	5, 3	4.5, 4, 3.5	
6 *PoznanHall2*	9	7, 5	6.5, 6, 5.5	1920×1088p
7 *UndoDancer*	CGI	1, 5	2, 3, 4	25 fps
8 *GTFly*	CGI	9, 1	7, 5, 3	

1) Partition-tree coding: The binary partition-tree is encoded following the pre-order depth-first traversal where the root of the tree is encoded first, then recursively the left sub-tree, and finally, recursively the right sub-tree. A tree can have six types of nodes, three types (X, Y, and P) of internal nodes representing binary split along the respective axis and three types (I, II, and III) of leaf nodes representing coding blocks. These types are encoded using Huffman codes in Table I that are generated by analysing the typical probability of each type in multiview test sequences. For a bitmap, the split position of the internal nodes are also encoded using fixed-length codes (see Section IV-D3). For an intmap, however, the split positions are fixed and hence, no encoding is needed.

2) Leaf node coding: Once a partitioning-tree is encoded, the contents of its Type II (intmap only) and III leaf nodes are encoded following the same tree-traversal order. For Type II leaf nodes in an intmap, the SNZ value of each node is encoded using arithmetic coding over all such nodes.

Type III leaf nodes are encoded with multiple coding modes to exploit varieties in probability distribution and the level of clustering tendency. Using partitioning or not, with context-adaptive or context-free arithmetic coding, differentiates four coding modes PC, P$\bar{\text{C}}$, $\bar{\text{P}}$C, and $\bar{\text{P}}\bar{\text{C}}$. In addition, using JPEG-LS without any partitioning is also available for $\mathcal{M}_{\text{res}}$ as the fifth coding mode J) to exploit 1D runs where no clustering is evident. For each map, the coding mode with the shortest code-length is selected and signalled using fixed-length 2-bit (all but residual maps) or 3-bit (residual maps) codes. For residual maps, the range $[0, R]$ is also signalled using an 8-bit unsigned integer.

If a coding mode with CAAC is selected, different context model is used for nominal 2D bitmaps ($\mathcal{M}_{\text{div}_{64}}$, $\mathcal{M}_{\text{div}_{32}}$, and $\mathcal{M}_{\text{div}_{16}}$), the nominal 3D bitmap ($\mathcal{M}_{\overline{\text{MVZ}}}$), the nominal intmap ($\mathcal{M}_{\text{mode}}$), and the ordinal intmap ($\mathcal{M}_{\text{res}}$), as outlined in Section IV-D1.

3) Motion vector coding: As zero-valued MV components in a frame have already been encoded with 2D bitmap $\mathcal{M}_{\overline{\text{MVZ}}}$, only the non-zero MV components need encoding in row-major scanning-order. Let ω be the search-width used in motion search. Then the non-zero MV components are drawn from a signed-integer source with 2ω symbols $\in [-\omega, \omega] \setminus \{0\}$.

3D-HEVC encodes MV components independently using predictive coding to exploit that spatial correlations in neighbouring motion vectors. The median of already-encoded adjacent MV components is considered as the prediction and the difference $\epsilon_p \in [-2\omega, 2\omega]$, $p \in \{1, 2\}$, is encoded with Exp-Golomb codes of order-0 using $2\lceil \log_2(|\epsilon_p| + 1) \rceil + 1$ bit. This technique is used by the first coding mode PG. When ϵ_p follows TSG distribution, due to high spatial correlations, the Exp-Golomb code is optimal. To cover cases with weaker correlations, a second coding mode PA is introduced, which uses arithmetic coding after signalling the range of symbols in $\lceil \log_2 2\omega \rceil$ bit.

If predictive coding is avoided, the absence of zero-values itself can be exploited to achieve additional half a bit compression efficiency, on average, per non-zero MV component. This third coding mode $\bar{\text{P}}$G uses modified Exp-Golomb codes of order-0 such that all positive or negative values, whichever is the majority in a frame, are encoded with codes of their preceding values of same sign (0 is assumed of both signs), after signalling the sign of majority in 1 bit. Similar to predictive coding, a fourth coding mode $\bar{\text{P}}$A is introduced, which uses arithmetic coding after signalling the range of symbols in $\lceil \log_2 \omega \rceil$ bit.

For each frame, the MV coding mode with the shortest code-length is selected and signalled using fixed-length 2-bit codes.

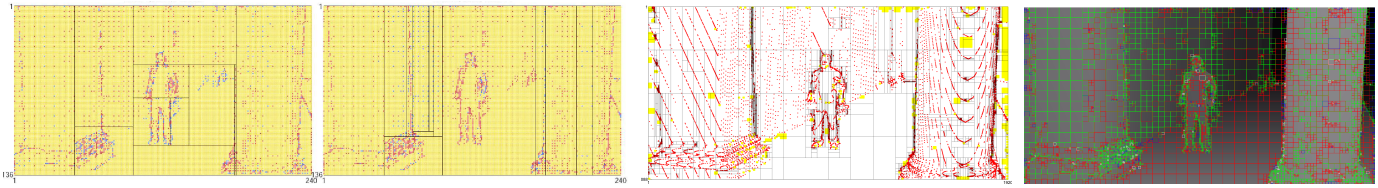
4) Decoding: All operations used for encoding have inverse-operations. Hence, the decoding is straightforward, applying corresponding inverse-operations while maintaining the order used during encoding.

V. EXPERIMENTAL RESULTS AND ANALYSES

This section presents simulation results and analyses to demonstrate the superiority of the proposed hierarchical cuboid coding depth map sequence coder. After outlining the experimental setup (Section V-A), effectiveness of BTBD partitioning is demonstrated with some examples (Section V-B), and coding and view-synthesis results of BTBD and some contemporary coding techniques are presented (Section V-C).

A. Experimental Setup

Experiments were carried out on eight MVD test sequences (Table II) recommended by the 3D video coding standardization group [43]. First six sequences were captured with linear camera arrangements of 7–12 cameras and the last two sequences were created with computer generated imagery (CGI). Texture frames were progressively scanned using 4:2:0



(a) (b) (c)

Fig. 4. BTBD partitioning of (a) 3D bitmap $\mathcal{M}_{\overline{\text{MVZ}}}$, shown with two 2D bitmaps (x-components (left) and y-components (right)); (b) 2D residual intmap $\mathcal{M}_{\text{res}}$, where blue/white, red, and yellow represent zero, non-zero, and don't-care (×), respectively; and (c) CTU divisions with CU prediction modes, where red, blue, white and green represents Intra, Skip, InterZ, and InterM modes, respectively, of *UndoDancer* sequence (view 1, frame 2, $Q = 1$). (best viewed in colour)

colour sampling. For both texture and depth, the sample precision was restricted to $8\,\text{bit}$. Views of all sequences were rectified.

Performance of the proposed BTBD depth coder was compared against the state-of-the-art 3D-HEVC depth coder, using the 3D-HEVC Test Model (3D-HTM) reference software version 7.0 [44]. Mono-view depth map sequences were encoded with BTBD and 3D-HEVC at different distortion levels by setting the scalar quantisation step $Q = 1, 3, \ldots, 15$ and the vector quantisation parameter $QP = 1, 5, 10, \ldots, 40$, respectively. The IPPPPPPPI group-of-picture (GOP) structure was used by both coders. With BTBD, P-frames were encoded through motion compensation using the diamond motion search [28] with search-width $\omega = 32$.

To evaluate near-lossless coding efficiency, depth maps of two selected views (Table II) in each sequence were encoded independently. Reported coding results were obtained by averaging the bitrate ($\text{bit per pixel (bpp)}$) and distortion (PSNR) of encoded depth maps of two coded-views.

3D-HEVC has not been designed for lossless compression as it is mostly based on explicit depth preservation with approximation. Thus, the following three alternatives were considered for conducting a comparative performance analysis of BTBD at lossless mode ($Q = 1$). Firstly, the pseudo-lossless 3D-HEVC ($QP = 1$), with the minimum possible quantisation noise, provides a baseline. Secondly, a simulated-lossless (Simul-LS) implementation of 3D-HEVC was considered. All edge-approximation coding modes were disabled and the residuals were encoded at frame-level similar to BTBD, i.e., without any frequency-domain transformation and using CAAC on ranked-residuals with BTBD context-model $\mathbb{C}_{\mathcal{M}_{\text{res}}}$ defined in (4). Thirdly, to justify motion-compensation used in BTBD, each frame of a depth map sequence was encoded independently with JPEG-LS using the JPEG-LS reference software version 2.2 [45]. To evaluate lossless coding efficiency, the average bitrate (bpp) of the encoded depth maps of the first selected view (Table II) and the corresponding compression-ratio $\text{cr} = 8/\text{bpp}$ are reported.

To evaluate view-synthesis quality, three intermediate equi-spaced virtual-views (Table II) were synthesised between the two coded-views using their decoded depth maps. Reported view-synthesis results were obtained by averaging the bitrate (bpp) of encoded depth maps of two coded-views and PSNR of rendered texture frames of three synthetic-views. The view synthesis reference software (VSRS) recommended by the 3D video coding standardization group [46] was used. While analysing view-synthesis quality from depth maps that were encoded at various levels of distortion, any undue interference from the decoded lossy texture frames was avoided by keeping the texture quantisation level fixed. Throughout the simulations, decoded texture frames of 3D-HEVC at a fixed intermediate quantisation level $QP = 25$ of two coded-views were used with both BTBD and 3D-HEVC for all quantisation levels. In absence of any reference for calculating PSNR of the rendered texture frames at the virtual-viewpoints of synthetic views, virtual-references were synthesised using the original uncompressed texture frames and depth maps of the two encoded-views.

To compare the overall performance of BTBD and 3D-HEVC in the lossy-compression domain, along with their applications in view-synthesis, the average Bjøntegaard delta (BD) metrics [47] for bitrate (BD-BR) and distortion (BD-PSNR) were calculated. The logarithmic bitrate ($\log_{10}\text{bpp}$) vs distortion (PSNR) curves were interpolated with cubic polynomials and the difference between their integrals over the common range was divided by the integration interval. Note that a negative BD-BR or a positive BD-PSNR means performance gain.

B. *Effectiveness of BTBD Partitioning*

The six data maps $\mathcal{M}_{\text{div}_{64}}$, $\mathcal{M}_{\text{div}_{32}}$, $\mathcal{M}_{\text{div}_{16}}$, $\mathcal{M}_{\text{mode}}$, $\mathcal{M}_{\overline{\text{MVZ}}}$, and $\mathcal{M}_{\text{res}}$ that BTBD uses differ in terms of map-dimension (2D vs 3D), data-range (binary vs integer), and degrees of clustering tendency (high vs moderate). Due to space limitation, 3D bitmap $\mathcal{M}_{\overline{\text{MVZ}}}$ and 2D intmap $\mathcal{M}_{\text{res}}$ of *UndoDancer* sequence (view 1, frame 2, $Q = 1$), covering all varieties, are selected to demonstrate the effectiveness of BTBD partitioning in exploiting the clustering tendency.

1) Zero/non-zero MV component bitmap: Fig. 4a presents BTBD partitioning of bitmap $\mathcal{M}_{\overline{\text{MVZ}}}$ of an arbitrary frame 2 in view 1 of *UndoDancer* sequence for lossless compression. For viewing convenience, the 3D bitmap is shown with two 2D bitmaps of x- and y-components. The partitioning tree has one 1D, 23 2D, and no 3D leaf nodes (cuboids) with maximum 663 and average 180 elements per cuboid. The number of zero, non-zero, and mixed leaf nodes are 1, 1, and 22, respectively. The number of splits orthogonal to x-, y-, and p-axis are 13, 6, and 4, respectively. The proposed BTBD technique encodes the bitmap with $2{,}794\,\text{bit}$ and the non-zero MV components with $11{,}632\,\text{bit}$, using arithmetic coding ($\bar{\text{PA}}$ coding mode),

TABLE III
LOSSLESS CODING PERFORMANCE OF BTBD AGAINST JPEG-LS AND 3D-HEVC (PSEUDO- AND SIMULATED-LOSSLESS) WHERE COMPRESSION-RATIO CR $= 8/$BPP

Seq	JPEG-LS		3D-HEVC ($QP = 1$)		3D-HEVC (Simul-LS)		BTBD ($Q = 1$)	
	bpp	cr	bpp	cr	bpp	cr	bpp	cr
1	0.345	×23.2	0.852	×9.4	0.250	×32.0	**0.230**	**×34.8**
2	0.489	×16.4	0.714	×11.2	0.271	×29.5	**0.259**	**×30.9**
3	0.440	×18.2	0.192	×41.6	0.105	×76.3	**0.091**	**×87.7**
4	0.269	×29.8	0.837	×9.6	**0.268**	**×29.8**	**0.268**	**×29.8**
5	0.676	×11.8	0.567	×14.1	**0.291**	**×27.4**	0.298	×26.8
6	0.137	×58.6	0.173	×46.4	0.144	×55.6	**0.097**	**×82.2**
7	0.324	×24.7	0.248	×32.3	0.089	×89.8	**0.074**	**×107.8**
8	0.270	×29.6	0.413	×19.4	0.231	×34.7	**0.199**	**×40.1**
Avg	0.369	×21.7	0.499	×16.0	0.206	×38.8	**0.190**	**×42.2**
Gain	−48.6 %		−62.0 %		−8.0 %			

in total 14,426 bit. Without the bitmap, encoding all (zero and non-zero) MV components with the same coding mode would have required 22,358 bit, resulting in −35.5 % coding gain for BTBD.

2) Residual intmap: Figs. 4b and 4c present BTBD partitioning of intmap $\mathcal{M}_{\text{res}}$ and the corresponding CTU divisions with CU prediction modes, respectively, of the same frame 2 in view 1 of *UndoDancer* sequence for lossless compression. The partitioning tree of the residual intmap has 554 1D and 1,446 2D leaf cuboids with maximum 65,216 and average 1,017 elements per cuboid. The number of zero, all SNZ, and mixed leaf nodes are 833, 49, and 1,118, respectively. CU prediction mode InterM is prominent as the sequence has high motion. For CU prediction mode Skip, the corresponding residuals are ignored, represented with the don't-care (×) symbols. Note that total 2,000 leaf nodes is only 6.1 % of the worst-possible CTU divisions when the frame is encoded with 32,640 CUs of size 8×8 pixel. Therefore, the complexity overheads of the BTBD decoder is of little concern.

C. Simulation Results

1) Lossless coding: Table III presents lossless coding performance results, depth bitrate (bpp) and corresponding compression-ratio, on all test sequences for JPEG-LS, 3D-HEVC (near- and simulated-lossless), and BTBD (lossless). Overall, average bitrate of 0.369, 0.499, 0.206, and 0.190 bpp was achieved by these four techniques, respectively. BTBD at lossless mode ($Q = 1$) outperformed other techniques with ×42.2 compression-ratio, on average, and −48.6 %, −62.0 %, and −8.0 % coding gain against JPEG-LS, 3D-HEVC ($QP = 1$), and 3D-HEVC (Simul-LS), respectively. On individual sequences, BTBD ($Q = 1$) outperformed 3D-HEVC (Simul-LS) in all but two, *Kendo* and *PoznanStreet*, where clustering tendency are not sufficiently strong to justify any partitioning, especially the overhead of encoding the binary tree. For these two sequences, however, BTBD performed very close to 3D-HEVC (Simul-LS) with negligible coding gain 0.0 % and 2.3 %, respectively.

Note that JPEG-LS outperformed 3D-HEVC ($QP = 1$) in five sequences despite being unable to exploit any temporal correlations. Unlike 3D-HEVC, JPEG-LS is capable of exploiting clustering tendency in some forms, e.g., compressing 1D-runs of very low values efficiently. This, however, is not surprising as performance gain by only intra-coding depth maps, based on geometric primitives, has been reported recently [48]. Nevertheless, the inter-coding path with motion compensation has the upper hand in lossless/near-lossless coding once a mechanism of exploiting clustering tendency in multi-dimensions (1D, 2D, and 3D), e.g., the proposed BTBD partitioning, is introduced.

2) Near-lossless coding: Table IV presents near-lossless coding performance results, depth bitrate (bpp), depth distortion (PSNR), and the average Bjøntegaard delta (BD) metrics BD-BR and BD-PSNR, for 3D-HEVC ($1 \le QP \le 40$) and BTBD ($3 \le Q \le 15$) on all test sequences. BTBD outperformed 3D-HEVC for all sequences with BD-BR −52.0 % to −97.9 % and BD-PSNR 5.39 dB to 10.04 dB, on average −79.4 % coding gain and 6.98 dB PSNR gain.

BTBD retained very high quality in decoded depth maps for all sequences. For $Q = 3, \ldots, 15$, on average, it achieved near-lossless PSNR 59.8 dB to 52.2 dB against low bitrate 0.100 bpp to 0.013 bpp with compression-ratio ×80.0 to ×608.8. To retain comparable near-lossless PSNR 59.0 dB to 51.5 dB, 3D-HEVC needed much higher bitrate 0.505 bpp to 0.096 bpp with compression-ratio ×15.8 to ×83.1 for $QP = 1, \ldots, 15$.

3) View-synthesis: Table V presents performance results in view-synthesis applications, depth bitrate (bpp), distortion in synthetic-texture (PSNR), and the average BD-BR and BD-PSNR metrics, for 3D-HEVC ($1 \le QP \le 40$) and BTBD ($1 \le Q \le 15$) on all test sequences. Compared to 3D-HEVC, BTBD rendered better quality synthetic-views for the first six sequences with BD-BR −8.2 % to −94.2 % and BD-PSNR 0.03 dB to 2.75 dB. For the remaining two CGI sequences *UndoDancer* and *GTFly*, BTBD performed mixed (BD-BR unfavourable; but BD-PSNR favourable) and unfavourably, respectively. On average, BTBD achieved −18.9 % coding gain and 0.43 dB PSNR gain against 3D-HEVC.

For natural sequences, depth maps are estimated indirectly from rectified texture frames at two or more views using stereo matching techniques. The larger the matching-window size, the smoother the depth map with less details. Depth maps of CGI sequences, however, are obtained directly from the 3D model of the scene geometry. They retain finer details with less smoothness, leading to exhibiting weaker clustering tendency. This explains why BTBD partitioning was less effective on CGI sequences. Nevertheless, subjective evaluations reveal that compared to 3D-HEVC, BTBD rendered views are visually more appealing due to lower shape deformation. Fig. 5 provides an example for subjective evaluation of synthetic-views of *Undodancer* sequence that were generated from the decoded depth maps of same quality by both techniques. Close scrutiny of the synthetic-view from 3D-HEVC depth maps reveals a number of shape deformations (encircled in red) due to use of edge-approximation modes. The synthetic-view from BTBD depth maps is free from such flaws.

TABLE IV
NEAR-LOSSLESS CODING PERFORMANCE OF BTBD AGAINST 3D-HEVC WITH AVERAGE BJØNTEGAARD DELTA METRICS FOR DEPTH BITRATE (BD-BR) AND DEPTH DISTORTION (BD-PSNR)

Seq	Rate	3D-HEVC ($QP=$)									BTBD ($Q=$)							BD-BR
	Dist	1	5	10	15	20	25	30	35	40	3	5	7	9	11	13	15	BD-PSNR
1	bpp	0.857	0.638	0.397	0.207	0.093	0.038	0.015	0.007	0.003	0.167	0.125	0.101	0.065	0.040	0.030	0.026	−76.3 %
	PSNR	61.4	58.1	54.3	51.0	47.6	44.6	41.5	39.2	36.3	59.9	56.9	53.8	51.9	50.3	49.2	48.8	6.42 dB
2	bpp	0.764	0.530	0.313	0.161	0.074	0.033	0.015	0.007	0.004	0.168	0.108	0.063	0.041	0.031	0.025	0.020	−84.2 %
	PSNR	57.4	52.7	49.3	46.9	44.4	41.7	39.7	38.2	36.1	57.3	53.3	51.3	49.9	49.2	48.8	48.1	8.01 dB
3	bpp	0.205	0.142	0.085	0.049	0.026	0.012	0.006	0.004	0.002	0.042	0.021	0.016	0.012	0.010	0.009	0.008	−88.8 %
	PSNR	55.8	54.6	53.4	52.0	50.5	48.7	47.0	45.3	42.4	58.5	55.9	54.8	54.0	53.5	52.8	52.4	5.72 dB
4	bpp	0.754	0.537	0.324	0.178	0.090	0.042	0.018	0.008	0.003	0.200	0.129	0.099	0.058	0.036	0.028	0.024	−69.8 %
	PSNR	60.8	58.3	55.1	52.1	49.1	46.0	42.8	39.3	34.8	59.5	57.1	52.7	52.4	52.1	50.8	50.5	5.54 dB
5	bpp	0.619	0.399	0.213	0.051	0.027	0.014	0.007	0.003	0.003	0.119	0.056	0.033	0.023	0.017	0.014	0.011	−79.6 %
	PSNR	59.4	56.5	53.8	49.1	46.6	44.7	43.0	40.8	40.8	56.6	54.3	53.4	52.7	52.2	51.9	51.5	5.98 dB
6	bpp	0.180	0.108	0.056	0.031	0.016	0.008	0.004	0.003	0.001	0.030	0.011	0.005	0.003	0.002	0.001	0.001	−52.0 %
	PSNR	63.8	62.4	60.3	57.1	54.3	51.5	48.9	46.1	43.0	63.4	58.8	58.3	57.9	57.7	57.5	57.3	8.67 dB
7	bpp	0.252	0.146	0.065	0.031	0.016	0.009	0.005	0.003	0.002	0.016	0.011	0.008	0.007	0.006	0.005	0.005	−97.9 %
	PSNR	55.5	54.9	53.7	51.0	50.7	49.7	48.7	46.4	43.3	62.9	60.9	59.3	58.1	57.4	56.4	55.8	10.04 dB
8	bpp	0.411	0.258	0.134	0.063	0.027	0.012	0.007	0.004	0.002	0.059	0.035	0.024	0.019	0.015	0.013	0.011	−86.8 %
	PSNR	58.2	57.0	55.2	53.1	50.5	48.8	47.7	46.0	41.6	60.3	57.5	55.7	55.0	54.2	54.1	53.0	5.39 dB
Avg	bpp	0.505	0.345	0.198	0.096	0.046	0.021	0.010	0.005	0.002	0.100	0.062	0.044	0.028	0.020	0.016	0.013	−79.4 %
	PSNR	59.0	56.8	54.4	51.5	49.2	47.0	44.9	42.7	39.8	59.8	56.8	54.9	54.0	53.3	52.7	52.2	6.98 dB

TABLE V
PERFORMANCE OF BTBD AGAINST 3D-HEVC IN VIEW-SYNTHESIS APPLICATIONS WITH AVERAGE BJØNTEGAARD DELTA METRICS FOR DEPTH BITRATE (BD-BR) AND SYNTHETIC-TEXTURE DISTORTION (BD-PSNR)

Seq	Rate	3D-HEVC ($QP=$)									BTBD ($Q=$)								BD-BR
	Dist	1	5	10	15	20	25	30	35	40	1	3	5	7	9	11	13	15	BD-PSNR
1	bpp	0.857	0.638	0.397	0.207	0.093	0.038	0.015	0.007	0.003	0.227	0.167	0.125	0.101	0.065	0.040	0.030	0.026	−30.0 %
	PSNR	45.2	45.1	45.0	44.7	44.1	43.5	43.0	42.4	41.8	45.2	45.0	44.7	44.2	43.9	43.7	43.6	43.4	0.14 dB
2	bpp	0.764	0.530	0.313	0.161	0.074	0.033	0.015	0.007	0.004	0.270	0.168	0.108	0.063	0.041	0.031	0.025	0.020	−94.2 %
	PSNR	41.5	40.7	40.1	39.5	39.2	38.3	37.8	37.7	37.5	43.3	42.4	41.7	41.4	41.3	41.1	40.9	40.8	2.75 dB
3	bpp	0.205	0.142	0.085	0.049	0.026	0.012	0.006	0.004	0.002	0.093	0.042	0.021	0.016	0.012	0.010	0.009	0.008	−8.2 %
	PSNR	42.9	42.8	42.5	42.2	41.8	41.5	41.3	41.1	40.6	43.0	42.2	41.8	41.5	41.4	41.3	41.2	41.1	0.03 dB
4	bpp	0.754	0.537	0.324	0.178	0.090	0.042	0.018	0.008	0.003	0.233	0.200	0.129	0.099	0.058	0.036	0.028	0.024	−49.4 %
	PSNR	45.9	45.8	45.8	45.6	45.4	45.2	45.0	44.6	44.0	45.9	45.8	45.7	45.6	45.5	45.5	45.4	45.4	0.17 dB
5	bpp	0.619	0.399	0.213	0.051	0.027	0.014	0.007	0.003	0.003	0.313	0.119	0.056	0.033	0.023	0.017	0.014	0.011	−29.8 %
	PSNR	41.9	41.6	41.1	40.2	39.8	39.5	39.3	38.8	38.8	42.1	40.8	40.3	40.0	39.9	39.8	39.8	39.7	0.20 dB
6	bpp	0.180	0.108	0.056	0.031	0.016	0.008	0.004	0.003	0.001	0.126	0.056	0.026	0.013	0.007	0.004	0.002	0.002	−26.5 %
	PSNR	44.1	43.9	43.8	43.5	43.3	43.0	42.7	42.4	42.1	44.2	43.7	43.4	43.3	43.1	43.0	42.9	42.7	0.18 dB
7	bpp	0.252	0.146	0.065	0.031	0.016	0.009	0.005	0.003	0.002	0.074	0.016	0.011	0.008	0.007	0.006	0.005	0.005	18.6 %
	PSNR	40.2	40.0	39.9	39.8	39.7	39.6	39.4	39.0	38.5	40.5	39.8	39.3	39.0	38.8	38.7	38.4	38.4	0.17 dB
8	bpp	0.411	0.258	0.134	0.063	0.027	0.012	0.007	0.004	0.002	0.197	0.059	0.035	0.024	0.019	0.015	0.013	0.011	68.7 %
	PSNR	42.4	42.3	42.1	41.9	41.7	41.5	41.5	41.4	41.1	42.5	42.0	41.5	41.3	41.0	40.7	40.4	40.1	−0.20 dB
Avg	bpp	0.505	0.345	0.198	0.096	0.046	0.021	0.010	0.005	0.002	0.191	0.103	0.064	0.045	0.029	0.020	0.016	0.013	−18.9 %
	PSNR	43.0	42.8	42.5	42.2	41.9	41.5	41.2	40.9	40.6	43.3	42.7	42.3	42.0	41.9	41.7	41.6	41.5	0.43 dB

VI. CONCLUSION

In this paper, a novel independent depth map sequence coder has been developed, which can efficiently exploit high spatio-temporal and inter-component correlations in CTU divisions, CU prediction modes, motion vectors, and residuals with 2D/3D binary/integer data maps. A simple but highly effective greedy heuristic based hierarchical decomposition technique has been introduced to adaptively partition the data maps into cuboids of data with skewed probability distribution that are encoded independently with context-adaptive arithmetic coding. The proposed BTBD technique achieved $\times 42.2$ compression-ratio on average for lossless coding to outperform the pseudo- ($QP = 1$) and simulated-lossless 3D-HEVC with average coding gain $-62.0\,\%$ and $-8.0\,\%$, respectively. For near-lossless coding, BTBD performed far superior compared to 3D-HEVC with average $-79.4\,\%$ coding gain and $6.98\,\mathrm{dB}$

Fig. 5. Synthetic-view 4 (frame 5) of *UndoDancer* sequence from the decoded depth maps of (left) 3D-HEVC ($QP = 40$), where shape deformations are encircled in red, and (right) BTBD ($Q = 15$), both rendered at the same synthetic-texture PSNR $\simeq$39 dB. (best viewed in colour)

PSNR gain. By restricting quantisation to spatial-domain, edges are preserved inherently. The synthetic-views rendered from the decoded depth maps from BTBD had superior visual quality with less shape deformations, achieving on average $0.43\,\mathrm{dB}$ PSNR gain against 3D-HEVC, which is perceptually significant. The proposed BTBD coding technique is elegant and it may have applications in other domains such as texture coding, hyperspectral image coding, image segmentation, and compressed-domain image processing that will be investigated in future.

REFERENCES

[1] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," in *Proc. SPIE, Stereoscopic Displays and Virtual Reality Systems XI*, Jan. 2004, pp. 93–104.

[2] G. J. Sullivan *et al.*, "Standardized extensions of high efficiency video coding (HEVC)," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1001–1016, 2013.

[3] G. Tech *et al.*, "Overview of the multiview and 3D extensions of high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, Jan 2016.

[4] P. Merkle, K. Müller, and T. Wiegand, "Coding of depth signals for 3D video using wedgelet block segmentation with residual adaptation," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME'13)*, Jul. 2013, pp. 1–6.

[5] K. Müller *et al.*, "3D high-efficiency video coding for multi-view video and depth data," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, 2013.

[6] M. Zamarin, M. Salmistraro, S. Forchhammer, and A. Ortega, "Edge-preserving intra depth coding based on context-coding and H.264/AVC," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME'13)*, Jul. 2013, pp. 1–6.

[7] S. Shahriyar, M. M. Murshed, M. Ali, and M. Paul, "Efficient coding of depth map by exploiting temporal correlation," in *Proc. IEEE Int. Conf. Digital Image Computing: Techniques and Applications (DICTA'14)*, Nov. 2014, pp. 1–8.

[8] M. Ali and M. Murshed, "An efficient predictive coding of integers with real-domain predictions using distributed source coding techniques," in *Advances in Multimedia Modeling*, ser. Lecture Notes in Computer Science, T.-J. Cham *et al.*, Eds. Berlin, Heidelberg: Springer, 2006, vol. 4351, pp. 227–236.

[9] S. Shahriyar, M. Murshed, M. Ali, and M. Paul, "Inherently edge-preserving depth-map coding without explicit edge detection and approximation," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW'14)*, Jul. 2014, pp. 1–6.

[10] M. J. Weinberger, G. Seroussi, and G. Sapiro, "From LOCO-I to the JPEG-LS standard," in *Proc. IEEE Int. Conf. Image Processing (ICIP'99)*, vol. 4, Oct. 1999, pp. 68–72.

[11] S. Shahriyar, M. Murshed, M. Ali, and M. Paul, "Cuboid coding of depth motion vectors using binary tree based decomposition," in *Proc. Data Compression Conf. (DCC'15)*, Apr. 2015, pp. 469–469.

[12] ——, "A novel depth motion vector coding exploiting spatial and inter-component clustering tendency," in *Proc. IEEE Int. Conf. Visual Communications and Image Processing (VCIP'15)*, Dec. 2015, pp. 1–4.

[13] ——, "Lossless depth map coding using binary tree based decomposition and context-based arithmetic coding," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME'16)*, Jul. 2016, pp. 1–6.

[14] B. Zhu *et al.*, "View synthesis oriented depth map coding algorithm," in *Proc. Asia-Pacific Conf. Information Processing (APCIP'09)*, vol. 2, Jul. 2009, pp. 104–107.

[15] S. Milani and G. Calvagno, "A depth image coder based on progressive silhouettes," *IEEE Signal Process. Lett.*, vol. 17, no. 8, pp. 711–714, 2010.

[16] F. Jager, "Contour-based segmentation and coding for depth map compression," in *Proc. IEEE Int. Conf. Visual Communications and Image Processing (VCIP'11)*, Nov. 2011, pp. 1–4.

[17] C. Lee and Y.-S. Ho, "Depth map coding using residual segmentation for 3D video system," *3D Research*, vol. 4, no. 2, pp. 1–9, 2013.

[18] Y. Morvan, P. H. N. de With, and D. Farin, "Platelet-based coding of depth maps for the transmission of multiview images," in *Proc. SPIE, Stereoscopic Displays and Applications XIII*, Jan. 2006, pp. 93–100.

[19] Y. Morvan, D. Farin, and P. H. N. de With, "Depth-image compression based on an R-D optimized quadtree decomposition for the transmission of multiview images," in *Proc. IEEE Int. Conf. Image Processing (ICIP'07)*, vol. 5, Sep. 2007, pp. 105–108.

[20] L. F. R. Lucas *et al.*, "Efficient depth map coding using linear residue approximation and a flexible prediction framework," in *Proc. IEEE Int. Conf. Image Processing (ICIP'12)*, Sep. 2012, pp. 1305–1308.

[21] M.-K. Kang, J. Lee, J. Y. Lee, and Y.-S. Ho, "Geometry-based block partitioning for efficient intra prediction in depth video coding," in *Proc. SPIE, Visual Information Processing and Communication*, vol. 7543, Jan. 2010, pp. 1–11.

[22] S. Liu, P. Lai, D. Tian, and C. W. Chang, "New depth coding techniques with utilization of corresponding video," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 551–561, 2011.

[23] P. Merkle *et al.*, "3D video: Depth coding based on inter-component prediction of block partitions," in *Proc. Picture Coding Symposium (PCS'12)*, May 2012, pp. 149–152.

[24] H. Oh and Y.-S. Ho, "H.264-based depth map sequence coding using motion information of corresponding texture video," in *Advances in Image and Video Technology*, ser. Lecture Notes in Computer Science, L.-W. Chang and W.-N. Lie, Eds. Berlin, Heidelberg: Springer, 2006, vol. 4319, pp. 898–907.

[25] S. Grewatsch and E. Miiller, "Sharing of motion vectors in 3D video coding," in *Proc. IEEE Int. Conf. Image Processing (ICIP'04)*, Oct. 2004, pp. 3271–3274.

[26] W. Su, D. Rusanovskyy, M. M. Hannuksela, and H. Li, "Depth-based motion vector prediction in 3D video coding," in *Proc. Picture Coding Symposium (PCS'12)*, May 2012, pp. 37–40.

[27] X. Liu, Y. Chang, Z. Li, and J. Huo, "Texture video-assisted motion vector predictor for depth map coding," *Optical Engineering*, vol. 50, no. 8, pp. 080 504–1–3, 2011.

[28] S. Zhu and K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, 2000.

[29] M. Maitre and M. N. Do, "Joint encoding of the depth image based representation using shape-adaptive wavelets," in *Proc. IEEE Int. Conf. Image Processing (ICIP'08)*, Oct. 2008, pp. 1768–1771.

[30] G. Shen *et al.*, "Edge-adaptive transforms for efficient depth map coding," in *Proc. Picture Coding Symposium (PCS'10)*, Dec. 2010, pp. 566–569.

[31] J. Duan *et al.*, "An improved video coding scheme for depth map sequences based on compressed sensing," in *Proc. Int. Conf. Multimedia Technology (ICMT'11)*, 2011, pp. 3401–3404.

[32] T. T. Do, X. Lu, and J. Sole, "Compressive sensing with adaptive pixel domain reconstruction for block-based video coding," in *Proc. Int. Conf. Image Processing (ICIP'10)*, 2010, pp. 3377–3380.

[33] S. Lee and A. Ortega, "Adaptive compressed sensing for depthmap compression using graph-based transform," in *Proc. Int. Conf. Image Processing (ICIP'12)*, 2012, pp. 929–932.

[34] K. Y. Kim, G.-H. Park, and D. Y. Suh, "Bit-plane-based lossless depth-map coding," *Optical Engineering*, vol. 49, no. 6, pp. 067 403–1–10, 2010.

[35] M. Zamarin and S. Forchhammer, "Lossless compression of stereo disparity maps for 3D," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW'12)*, Jul. 2012, pp. 617–622.

[36] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, 2003.

[37] J. Heo and Y.-S. Ho, "Improved context-based adaptive binary arithmetic coding over H.264/AVC for lossless depth map coding," *IEEE Signal Process. Lett.*, vol. 17, no. 10, pp. 835–838, 2010.

[38] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Comput.*, vol. 45, no. 4, pp. 437–444, 1997.

[39] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. Wiley, 2011.

[40] R. F. Rice, "Some practical universal noiseless coding techniques. Part III, Module PSI14,K+," Jet Propulsion Lab., Pasadena, USA, Tech. Rep. NASA-CR-188718, Nov. 1991.

[41] M. J. Weinberger, J. J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Process.*, vol. 5, no. 4, pp. 575–586, 1996.

[42] C. H. Kapadia and R. L. Thomasson, "On estimating the parameter of a truncated geometric distribution by the method of moments," *Annals of the Institute of Statistical Math.*, vol. 27, no. 1, pp. 269–272, 1975.

[43] ISO/IEC JTC1/SC29 WG 11, "Call for proposals on 3D video coding technology," Tech. Rep. MPEG2011/N12036, Mar. 2011.

[44] JCT-3V. (2013, May) 3D-HEVC Test Model (HTM 7.0). [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-7.0/

[45] The SPMG Lab. (1999, Jun.) JPEG-LS Reference Software Version 2.2. University of British Columbia, Canada. [Online]. Available: http://www.stat.columbia.edu/~jakulin/jpeg-ls/mirror.htm

[46] ISO/IEC JTC1/SC29 WG 11. (2011, Jul.) View Synthesis Reference Software (VSRS). [Online]. Available: http://www.merl.com/pub/avetro/3dv-cfp/

[47] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. VCEG-M33, ITU-T Q.6/SG16, April 2001.

[48] P. Merkle, K. Müller, D. Marpe, and T. Wiegand, "Depth intra coding for 3D video based on geometric primitives," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 570–582, 2016.