Arbitrarily Shaped Motion Prediction for Depth Video Compression using Arithmetic Edge Coding

Ismael Daribo *Member, IEEE*, Dinei Florencio *Senior Member, IEEE*, Gene Cheung *Senior Member, IEEE*

Abstract*—Depth image compression is important for compact representation of 3D visual data in "texture-plus-depth" format, where texture and depth maps from one or more viewpoints are encoded and transmitted. A decoder can then synthesize a freely chosen virtual view via depth-image-based rendering (DIBR) using nearby coded texture and depth maps as reference. Further, depth information can be used in other image processing applications beyond view synthesis, such as object identification, segmentation, etc. In this paper, we leverage on the observation that "neighboring pixels of similar depth have similar motion" to efficiently encode depth video. Specifically, we divide a depth block containing two zones of distinct values (e.g., foreground and background) into two arbitrarily shaped regions (sub-blocks) along the dividing boundary before performing separate motion prediction (MP). While such arbitrarily shaped sub-block MP can lead to very small prediction residuals (resulting in few bits required for residual coding), it incurs an overhead to transmit the dividing boundaries for sub-block identification at decoder. To minimize this overhead, we first devise a scheme called arithmetic edge coding (AEC) to efficiently code boundaries that divide blocks into sub-blocks. Specifically, we propose to incorporate the boundary geometrical correlation in an adaptive arithmetic coder in the form of a statistical model. Then, we propose two optimization procedures to further improve the edge coding performance of AEC for a given depth image. The first procedure operates within a code block, and allows lossy compression of the detected block boundary to lower the cost of AEC, with an option to augment boundary depth pixel values matching the new boundary, given the augmented pixels do not adversely affect synthesized view distortion. The second procedure operates across code blocks, and systematically identifies blocks along an object contour that should be coded using sub-block MP via a rate-distortion optimized trellis. Experimental results show an average overall bitrate reduction of up to 33% over classical H.264/AVC.

I. Introduction

The cost of consumer-level cameras has been rapidly decreasing, and video of a time-varying 3D scene can now be captured simultaneously by an array of multiple closely spaced cameras in a cost-effective manner [1]. The 3D geometry of the scene can be estimated from these multiple cameras, or, (also at a rapidly decreasing cost) acquired directly by depth-sensing cameras [2]. The addition of depth information to the traditional texture video creates a powerful combination, opening up a number of new applications. The acquired depth information may, for example, help in object identification, foreground / background segmentation, image re-lighting, mixed reality, and a number of novel processing tasks. If the processing is to be performed at a distant location from where the signals were acquired, there is a need for encoding these texture and depth maps for network transmission. Of particular interest in many immersive visual communication applications is a functionality called *free viewpoint*: the ability for a user to freely select any arbitrary virtual viewpoint, and have the system render the corresponding viewpoint image for observation. Example applications include free viewpoint TV [3], high-quality video conferencing [4], etc. In these scenarios, time-lag considerations typically demand the synthesis to be done at the viewer's site, making transmission of texture and depth maps a requirement.

Transmitting both texture and depth maps of one or more viewpoints from sender to receiver entails a large network cost, however, and hence compression of both texture and depth video to reduce data rate is important. Compression of texture video is well studied during the past decades, and is the focus of many past video coding standards like H.264/AVC [5]. Given the maturity of texture video coding, and the obvious correlation that exists between texture and depth video from the same camera viewpoint, it is natural to consider depth video coding using already coded texture video as side information (SI) to exploit the inherent correlation between them for bitrate reduction. This approach was taken in [6], for example, where edges in texture maps are detected and subsequently reused as edges in depth maps for intra-prediction (more details will be discussed in Section II). In this paper, we take the opposite approach and consider the compression of depth video independent from the texture video; texture video can be subsequently compressed instead using coded depth

Ismael Daribo was with National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan during this work, and is now with the European Patent Office (email: daribo.labs@gmail.com).

Dinei Florencio is with Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA (email: dinei@microsoft.com).

Gene Cheung is with National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430 Japan (email: cheung@nii.ac.jp).

This work is supported by Microsoft Research CORE program.

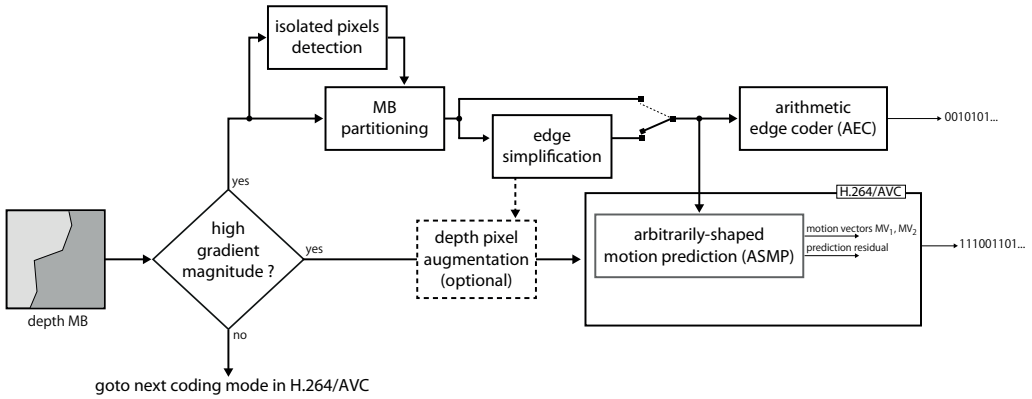


Fig. 1. Workflow of the proposed framework. ("Yes" to the "high graident magnitude" box means possibly two arrows need to be followed, if depth pixel segmentation is chosen to be performed.) A depth MB with high discontinuity is motion predicted using two non-overlapping arbitrarily shaped regions. The dividing boundary is side coded using an adaptive arithmetic coder.

video as SI [7]. The reason is the following key observation: in general captured video, *neighboring pixels of similar depth tend to belong to the same physical object and hence have similar motion* [8]. In depth video coding, by definition per-pixel depth information is available. Thus, if a given code block contains two zones of distinct depth values (e.g., foreground and background), then one can divide it into two arbitrarily shaped regions (sub-blocks) along the dividing boundary, before performing motion prediction (MP) separately for each of the sub-blocks. As we will demonstrate in our experiments, this method of *arbitrarily shaped sub-block motion prediction* (ASMP) can lead to very small prediction residuals, and hence very good depth video compression performance. Further, The dividing boundary can also be reused for similar ASMP in the corresponding block in the texture map of the same viewpoint as well [7]. The overall workflow of our proposed framework is illustrated in Fig. 1.

While ASMP can lead to very small prediction residuals, it incurs an overhead to—lossily or losslessly—encode the dividing boundary for sub-block identification. To minimize this overhead, we propose an *arithmetic edge coding* (AEC) scheme for ASMP in depth video coding. We first devise an edge prediction scheme based on linear regression to predict the next edge direction in a contiguous contour based on past observed edges. From the predicted edge direction, we assign probabilities to each possible edge direction using the von Mises distribution. The computed probabilities are subsequently inputted to a conditional arithmetic codec for entropy coding. This is possible with the arithmetic coder that encourages a clear separation between the statistical model for representing the data and the coding of information with respect to that model. To further optimize the performance of our proposed AEC scheme in the context of depth video coding, we perform two optimization procedures, within-block and across-block, respectively. For within-block optimization, we allow lossy compression of the detected block boundary to lower the cost of AEC—i.e., code a simpler dividing

Fig. 2. Example of depth irregularities along boundaries, which make harder the prediction of the geometrical structure of the boundary.

boundary that can be more easily predicted by our statistical model. Optionally, we can then augment *imprecise depth pixels*—boundary pixels along the new boundary that when converted from background pixels to foreground (or vice versa) will not adversely affect synthesized view distortion—to match the new boundary, in order to reduce prediction residual energy. Due to the limitation of depth sensors and depth estimation algorithms, object boundaries commonly contain a certain amount of localized irregularities as shown in Fig. 2. How to detect and eliminate those irregularities along the detected block-dividing boundary is crucial to improve the boundary coding performance. Second, given contiguous boundaries across blocks along an object's contour can be efficiently coded using AEC, we identify which blocks along an object's contour should encode their block-dividing boundaries to optimize overall rate-distortion (RD) performance, by finding the shortest path in a trellis.

We implemented our proposal as a new coding mode called *z-mode* for coding of 16×16 pixel blocks in H.264/AVC. Experimental results show that our proposal can reduce overall bitrate by up to 33% compared to

classical H.264/AVC without the proposed z-mode, with respect to Bjontegaard metric [9].

The outline of the paper is as follows. We first discuss related work in Section II. We then discuss the arbitrarily shaped sub-block MP scheme for depth video in Section III. We discuss the basic AEC engine in Section IV, and how we can optimally apply AEC in a depth video frame in Section V. Experimental results and conclusions are presented in Section VI and VII, respectively.

II. Related Work

We divide our discussion of related work into three sections. We first discuss existing work in the literature for depth map coding. We then discuss previous work in motion field and lossless edge coding in video compression. Finally, we juxtapose our new contributions in this paper to our previous publications on the same topic.

A. Depth Map Coding

Because of the popularity of texture-plus-depth representation of 3D visual data [10], there is a growing interest in the compression of depth maps. A large portion of this work focuses on exploiting the unique characteristics of depth maps that are different from texture maps: sharp edges and smooth interior surfaces, as illustrated in Fig. 2. [11], [12] proposed to use edge-adaptive *graph Fourier transform* (GFT) for transform coding of rectangular pixel blocks in a depth image, so that filtering across detected edges is avoided, which would otherwise result in large high-frequency components. [13] proposed to encode the detected edges losslessly, then use *partial differential equations* (PDE) to interpolate the pixels inside the edges, given the interior surfaces are likely smooth (i.e., low frequency content only). Computing adaptive transforms and solving PDEs mean a non-trivial computation burden at the decoder, however. Further, while these works exploit the spatial characteristics of depth signals for intra-coding, we focus instead on the temporal aspect of depth video coding using our proposed arbitrarily shaped sub-block motion prediction (ASMP) scheme.

Alternative representations of depth maps have also been proposed. [14] used *platelets*—piecewise linear functions—to code depth images. Assuming that texture maps are first coded independent of depth maps, [6] proposed to identify edges in the texture maps that would also appear in corresponding depth maps of the same viewpoints. [6] then performed intra-prediction to estimate a constant depth value for pixels on one side of a code block divided by a contiguous boundary of detected edges, using causal neighboring depth blocks that have already been coded and on the same side of the boundary. However, because depth maps are piecewise smooth and not necessarily piecewise linear or constant, [14] and [6] have fixed non-zero representation errors at any coding rate. [15] proposed to encode a low spatial resolution of depth maps at the encoder, so that at the decoder the decoded depth maps can be up-sampled and interpolated using *weighted mode filtering* (WMF). While WMF performs reasonably well in recovering high-resolution edge details, it is not a perfect reconstruction scheme, and thus there are also non-zero reconstruction errors even at high rate. In contrast, our proposed AEC can lead to perfect edge reconstruction at decoder, if coding rate can be afforded.

Assuming depth maps are encoded for the sole purpose of view synthesis at decoder via depth image-based rendering (DIBR), [16], [17] observed that depth maps are then merely a means to the end of view synthesis, and are not directly viewed themselves. Thus, [16], [17] proposed to use a computation-efficient model to estimate the resulting synthesized distortion during rate-distortion (RD) optimized mode selection in depth video coding, rather than the distortion of the depth signal itself. [18]–[20] made a similar observation that errors in different depth pixels have different degrees of adverse effect in the synthesized view, and proposed to characterize the synthesized view distortion sensitivity to errors in a depth map using per-pixel *don't care region* (DCR). Our proposed ASMP—implemented as an additional z coding mode for the coding of depth maps—is orthogonal to these works; a mode optimization scheme that uses synthesized view quality as criteria can subsequently be used to select the most suitable modes for a group of blocks.

Another line of attack for depth map coding is to exploit the inherent correlation between texture and depth maps from the same viewpoint during joint compression. [21] first proposed to reuse motion vectors (MVs) in texture maps for depth maps as well, lowering overall bitrate. As a follow-up investigation, [22] proposed to find a single set of MVs that minimize a weighted sum of residual prediction energy of both texture and depth video. [23] proposed to encode detected edges in texture and depth maps only once, where the detected edges are used during edge-adaptive wavelet coding of both texture and depth maps. Our proposal is different in that we do not rely on the inter-correlation between texture and depth maps for coding gain. Instead, given the observation that "neighboring pixels of similar depth have similar motion"[1], we encode depth maps so that a compact representation of the motion field (one MV for each divided sub-block of pixels) can be found. If texture video needs to be encoded as well, the encoded edges can then be shared for similar ASMP [7]. We note that some applications require the sender to convey only the dynamic 3D geometry of the scene to the receiver, in which case the encoded depth video alone is sufficient without any texture video compression [24], [25]. Our proposal clearly can be applicable to these applications as well.

It is observed [26], [27] that depth maps often are

[1]The observation that pixels of similar depth have similar motion has been made previously [8], where unlikely coding modes are eliminated *a priori* for faster H.264/AVC encoding.

corrupted with noise during acquisition, and given coding noise incurs coding cost but does not improve performance, one should denoise depth maps prior to the coding process. We believe that as depth sensing technologies improve, depth acquisition noise will become less of a problem in the future. Further, preliminary investigation in [28] has shown that a joint denoising / coding approach can lead to better performance than a separate approach where denoising and coding are performed in individual steps in stages. While the direction in [28] is promising, for the sake of simplicity, we will focus only on the depth video compression aspect and assume in this work that the input depth maps are not corrupted by severe noise.

B. Motion Field and Lossless Edge Coding

MP in H.264/AVC [5] offers different block sizes (rectangular blocks from 16×16 down to 4×4) as different coding modes during video encoding. However, to accurately track the motion of an arbitrarily shaped object inside a code block, many small sub-blocks along the object boundary are needed, resulting in a large overhead in coding many MVs for the many small sub-blocks. Further, searching for the most appropriate block sizes by evaluating the rate-distortion (RD) costs of possible coding modes is computationally expensive. Alternatively, line-based segmentation schemes [29], [30] divide a code block using an arbitrary line segment that cuts across the code block. There are two problems to this approach: i) an object's boundary is often not a straight line, resulting in shape-mismatch; and ii) it is still computationally expensive to search for a RD-optimal dividing line segment. In our proposal, because the detected depth edges follow the object's contour, we can easily segment a code block along foreground / background boundary with pixel-level accuracy. Moreover, the depth edge can be acquired cheaply via simple edge detection techniques.

Shape coding in images has a long history [31], and more recently has been studied in the context of MPEG4 for the coding of video object plane (VOP) [32]. Our study of edge coding is driven by the necessity of depth video coding, and leverages on the ease of context model in arithmetic coding. Note that our proposed AEC scheme, which shows significant compression performance gain compared to existing schemes, can also be used directly in recent popular image / video coding schemes based on edge-adaptive wavelets [23] and graph Fourier transform [11], [12], where edge coding is paramount in determining the overall compression performance.

C. Comparison with Our Previous Work

In our first previous work [7] on the same topic, we assumed that the depth video is first independently coded and the texture video is coded thereafter using coded depth video as side information (SI). With this assumption, we can use detected boundary in a depth code block to divide a corresponding block in texture map of the same viewpoint into two arbitrarily shaped sub-blocks for separate MP. In our follow-up work [33], we considered compression of depth video independent from texture video, also using the idea of ASMP. However, because depth video is independently coded (from texture video), the dividing boundary in a depth code block needs to be explicitly encoded and signaled to the decoder for correct decoding. We thus devised an AEC scheme for this purpose. This paper is a non-trivial extension of our work in [33]. First, we reassign isolated depth pixels to appropriate sub-blocks during macroblock partitioning, which improves the performance of the subsequent motion prediction procedure. Second, we extend AEC to include lossy coding of block boundaries in the proposed framework. Correspondingly, we can optionally identify and augment imprecise depth pixels (from foreground to background (or vice versa)) to match the new encoded boundary with marginal increase in synthesized view distortion. Finally, we propose a trellis-based optimization to decide which blocks along a detected object contour in a depth image should be encoded using ASMP in an RD optimal fashion. Experimental results show that these optimizations can bring about substantial gain over our previous work [33].

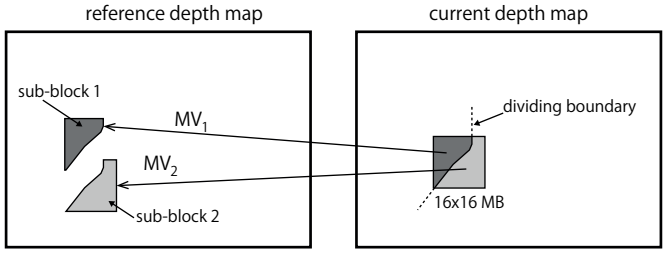


Fig. 3. Motion compensation of sub-blocks in depth map divided using detected edges.

III. Sub-block Motion Prediction

To show the effectiveness of our two proposed techniques—-arbitrarily shaped sub-block motion prediction (ASMP) and arithmetic edge coding (AEC)—for depth video coding, we implemented them together into a new additional coding mode called *z-mode* for encoding of a 16×16 code block in H.264/AVC [5]. While smaller blocks can also be considered, using smaller blocks results in more motion vectors (MV) that require encoding, leading to a larger overhead. Experimentally we found 16×16 block size to be sufficient. The main idea behind z-mode is to partition a 16×16 code block (macroblock or MB) into two non-overlapping arbitrarily shaped regions for separate motion estimation and compensation as illustrated in Fig. 3. This new coding mode involves the following steps:

1) Partition the current MB into two sub-blocks using a chosen edge detection scheme.

2) Compute a predicted motion vector (PMV) for each sub-block using MVs of neighboring causal blocks of similar depth in the current frame.
3) For each sub-block, perform motion estimation; i.e., find the best-matched sub-block in a reference frame for the current target sub-block.
4) Compute prediction residuals in the current MB given the two motion-compensated sub-blocks for residual transform coding in Discrete Cosine Transform (DCT).

We describe step 1 and 2 in details next. Step 3 and 4 are similar to standard MP performed in H.264/AVC, and we will only highlight the differences when employed in our proposed z-mode.

A. Macroblock Partitioning

We first define notations for ease of discussion. Consider a depth map Z of a certain viewpoint. A 16×16 MB support is denoted by a set of pixel offsets $\Phi = \{(0,0),(0,1),\ldots(15,15)\}$. Thus, a 16×16 MB with top-left corner at pixel (x,y) in the depth map is $Z_\Phi(x,y)$. MB support Φ can be partitioned into two non-overlapping sub-block supports Φ_1 and Φ_2, where $\Phi = \Phi_1 \cup \Phi_2$ and $\Phi_1 \cap \Phi_2 = \emptyset$, so that $Z_\Phi(x,y) = Z_{\Phi_1}(x,y) \cup Z_{\Phi_2}(x,y)$. See Fig. 3 for an illustration.

We first identify potential MB candidates for encoding in our proposed z-mode. As previously mentioned, our proposed z-mode works well for the case when a MB contains exactly two objects, which entails a depth discontinuity at the boundary between the two objects in the MB. Thus, a MB with an average gradient magnitude larger than a certain threshold is identified as possible candidate for our proposed z-mode.

The next step of our proposed z-mode is MB partitioning. Given depth block $Z_\Phi(x,y)$, we divide MB support Φ into two non-overlapping sub-block supports Φ_1 and Φ_2 as follows:

$$\begin{aligned} \Phi_1 &= \{(i,j) \in \Phi \mid Z(x+i,y+j) < \bar{z}_\Phi(x,y)\} \\ \Phi_2 &= \{(i,j) \in \Phi \mid Z(x+i,y+j) \geq \bar{z}_\Phi(x,y)\} \end{aligned} \quad (1)$$

where $\bar{z}_\Phi(x,y)$ is the arithmetic mean of depth values in depth block $Z_\Phi(x,y)$, and $Z(x,y)$ is the depth value at pixel (x,y) in depth map Z. In other words, $Z_{\Phi_1}(x,y)$ and $Z_{\Phi_2}(x,y)$ are the sets of depth pixels with values smaller than, or larger than and equal to, block-wise depth value mean $\bar{z}_\Phi(x,y)$, respectively. Assuming MB $Z_\Phi(x,y)$ contains only one foreground object (small depth) in front of a background (large depth), (1) can segment pixels in MB $Z_\Phi(x,y)$ into foreground $Z_{\Phi_1}(x,y)$ and background $Z_{\Phi_2}(x,y)$. This statistical approach is robust and of very low complexity. A similar approach was also taken in [6] to divide a texture block into two sub-blocks. It is important to note that at the termination of the MB partitioning procedure, there is no T-junction in the dividing boundary. This observation has important implication to AEC to be discussed in Section IV.

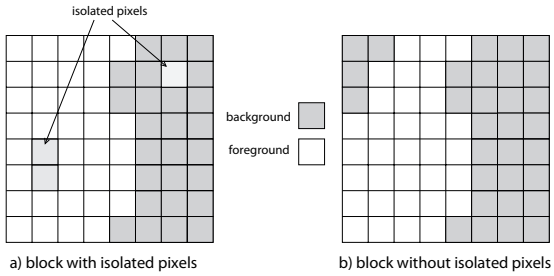


Fig. 4. Example of depth MBs where, in the presence of isolated pixels, the block partitions do not take into account these pixels to ensure connected pixel components as sub-blocks.

The above procedure to partition an original pixel support Φ into two smaller supports, Φ_1 and Φ_2, works well in dividing pixels into foreground and background most of the time. However, there are rare occasions when pixels in a sub-group Φ_i do not live in a single connected region. This happens, for example, if a very small number of *interior* pixels in a support Φ_i were corrupted by noise during acquisition/estimation, so that the corrupted pixels were assigned to the sub-group disconnected to them. See Fig. 4(a) for an illustration. It is clear that in this case, a single MV for a non-connected sub-group would unlikely capture well the motion for all pixels in the sub-group. Note that this observation applies only to pixels interior to the block. As shown in Fig. 4(b), there can be regions that are not connected in the block but connected in the large image. A single MV in this case can still capture motion well for pixels in the same sub-group but in non-connected regions. Thus, we detect these isolated interior group of pixels and merge them with the surrounding support as described below. We perform the following procedure to reconstruct supports Φ_1 and Φ_2, so that when the procedure terminates, we are guaranteed to have a single connected region for pixels in a support Φ_i, excluding pixel regions that are at the boundary of the block:

1) Initially partition the block as described by Eq. (1).
2) Identify isolated interior group of pixels as region with a closed boundary.
3) For each detected isolated interior group pixels, we reassign their identification (background or foreground) to the one of the surrounding region.

B. Motion Vector Prediction

Bits required to encode the MV $v_{\Phi_s}(x,y)$ for a sub-block $Z_{\Phi_s}(x,y)$, $s \in \{1,2\}$, can be reduced if a good PMV, $u_{\Phi_s}(x,y)$, can be estimated from MVs $v_\Phi(m,n)$'s of neighboring causal code blocks, $(m,n) \in \mathcal{N}(x,y)$. Thus, only the motion vector difference (MVD) between the PMV $u_{\Phi_s}(x,y)$ and the sub-block true MV $v_{\Phi_s}(x,y)$ is encoded and transmitted. In H.264/AVC [5], PMV is computed to be the median of the MVs of neighboring blocks.

1) Depth-based Predicted Motion Vector: The reason median filter is applied to the neighboring blocks' MVs is to

eliminate outliers that have motion uncorrelated to the motion of the target block, which happens, for example, when a foreground object has different motion than the background. When coding depth map, however, we have available depth values to evaluate the "trustworthiness" of motion information provided by neighboring blocks. In other words, assuming neighboring pixels of similar depth have similar motion, we can discredit a neighboring block's MV if it has depth very different from our target block.

Specifically, we propose to compute the PMV $u_{\Phi_s}(x,y)$ for sub-block $Z_{\Phi_s}(x,y)$ as a weighted sum of MVs, $v_\Phi(m,n)$'s of causal neighboring blocks, $(m,n) \in \mathcal{N}(x,y)$, where the weights w's are proportional to the depth similarity between the neighboring block averages $\bar{z}_\Phi(m,n)$'s and the *estimated* target sub-block average $\tilde{z}_{\Phi_s}(x,y)$—estimated using causal pixels just across target sub-block boundaries. Mathematically, we write:

$$u_{\Phi_s}(x,y) = \frac{1}{\bar{w}} \sum_{(m,n)\in\mathcal{N}(x,y)} w(\tilde{z}_{\Phi_s}(x,y) - \bar{z}_\Phi(m,n))\ v_\Phi(m,n) \quad (2)$$

where $\bar{w} = \sum_{(m,n)\in\mathcal{N}(x,y)} w(\bar{z}_{\Phi_s}(x,y) - \bar{z}_\Phi(m,n))$ is a scaling factor so that the sum of weights w's divided by $\bar{w}$ is 1. The real-valued precision of the resulting PMV $u_{\Phi_s}(x,y)$ is then rounded to half-pel or quarter-pel precision to be H.264/AVC compliant.

2) Finding Optimal Weights: To find the optimal weights w's, we first assume it follows a Laplacian distribution with parameter $b > 0$:

$$w(x) = \frac{1}{2b} e^{-\frac{|x|}{b}} \quad (3)$$

The reason we use a Laplacian distribution as opposed to Gaussian, is because Laplacian has a sharper decrease, and we prefer to quickly rule out MVs of blocks that are even moderately different from the target block in the PMV computation.

We find the optimal parameter b^* that minimizes the sum of absolute errors between prediction $u_{\Phi_s}(x,y)$ and true MV $v_{\Phi_s}(x,y)$ for each sub-block $Z_{\Phi_s}(x,y)$ in a set Θ of training data:

$$b^* = \arg\min_{b>0} \sum_{Z_{\Phi_s}(x,y)\in\Theta} \|u_{\Phi_s}(x,y) - v_{\Phi_s}(x,y)\|_1 \quad (4)$$

where $\|.\|_1$ denotes the l_1-norm.

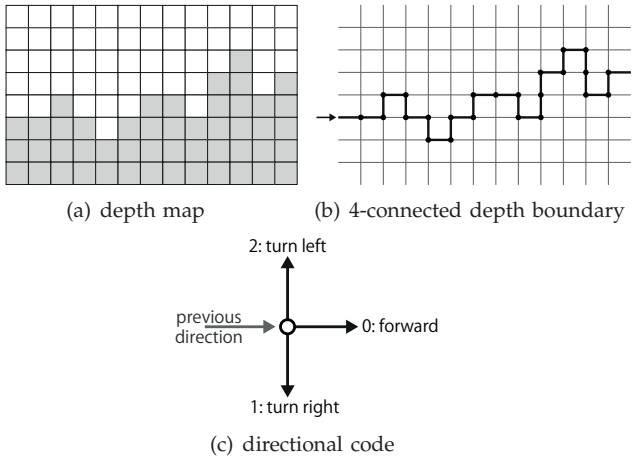
We note that the encoded bitstream can only be correctly decoded if both encoder and decoder have the same block partition information. In the following section, we propose an efficient arithmetic edge coding scheme encoding the boundary that divides a block into two sub-blocks.

IV. Arithmetic Edge Coding

In this section, we address the problem of losslessly encoding the boundary that separates a z-mode MB[2] into two sub-blocks. The overall coding scheme can be summarized in the following steps for each MB:

1) Given a depth MB with discontinuities (i.e., large block-wise average gradient magnitude), represent the two partitions by their common boundary (a series of between-pixel edges).
2) Map the boundary into a directional 4-connected chain code, also known as Freeman chain [31].
3) Given a window of consecutive previous edges, predict the next edge by assigning probabilities to possible edge directions.
4) Encode each edge in the boundary by inputting the assigned direction probabilities to a conditional arithmetic coder. Edges of a series of neighboring blocks that form a contiguous contour in the frame can be coded as a single arithmetic codeword for better coding efficiency (to be discussed in more details in Section V-C).

We discuss these steps in order next.



(a) depth map (b) 4-connected depth boundary (c) directional code

Fig. 5. Depth map boundary represented by a directional 4-connected chain code. In this example, the corresponding directional code is: '0−0−2−1−1−2−1−2−2−1−2−1−0−1−2−2−0−1−2−1−1−0−1−1−2'.

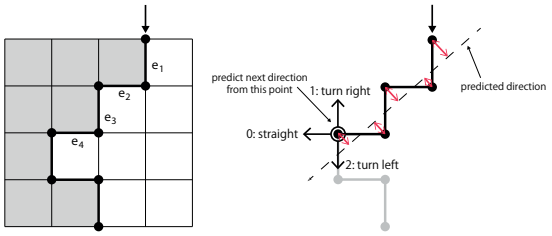
A. Differential Chain Code

A MB boundary divides pixels in the MB into two sub-blocks. Note that the boundary exists *in-between* pixels, not *on* pixels. See Fig. 5(b) for an illustration of a boundary. As shown, the set of edges composing the boundary is a *4-connected chain code*. This boundary representation is also known as *directional chain code* (DCC), which belongs to the family of chain coding schemes pioneered by Freeman [31]. At each boundary pixel only three possible directions can be taken: "forward", "turn right" and "turn left" with respect to the previous direction, where the code 0, 1, 2 is assigned to each direction, respectively (see Fig.5(c)). Let us note that T-junctions are not possible using the partition boundary detection scheme discussed in Section III-A.

B. Edge Prediction

Given the definition of DCC, edges e_t's can be entropy-encoded consecutively, where each edge has an alphabet

[2]By *z-mode* MB, we mean a MB that has been encoded in the aforementioned *z-mode*.



(a) edges for 4 × 4 block (b) linear regression

Fig. 6. Detected edges in a 4×4 depth pixel block and linear prediction using previous edges.

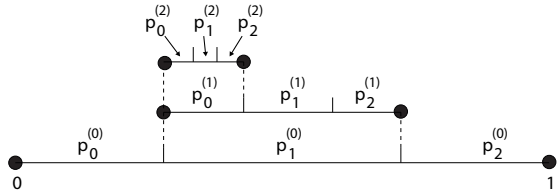


Fig. 7. Example of arithmetic edge coding, where the first two symbols to be encoded are 1 and 0.

of three symbols representing the three possible edge directions. There are many options for entropy coding. One notable example is *prediction by partial matching* (PPM) [34], which predicts the next symbol given observations of the previous ones in the symbol stream. PPM model is typically designed based on statistics of previous symbols.

In our case, rather than the repeated patterns of text, a boundary often follows the contour of a physical object in the 3D scene, and hence possesses geometrical structures that can be exploited to more accurately predict the most likely direction for the next edge. Based on this observation, we propose a geometrical prediction model to estimate probabilities of the three possible directions for the next edge, given observation of a window of previous edges. The estimated direction probabilities are subsequently inputted into an adaptive arithmetic coder for entropy coding.

1) Linear prediction: We predict the direction of the next edge e_{t+1} by first constructing a line-of-best-fit using a window of previous edges via *linear regression*. Specifically, given end points $p_{t-K}, \ldots, p_t$ of a window of K previous edges $e_{t-K+1}, \ldots, e_t$, we construct a line l that minimizes the sum of squared errors $\sum_{i=t-K}^{k} \epsilon_i^2$, where ϵ_i is the minimum distance between line l and end point p_i. See Fig. 6(b) for an illustration where a line-of-best-fit is drawn to minimize squared error sum $\sum_{i=0}^{4} \epsilon_i^2$ given window of given edges $\{e_1, e_2, e_3, e_4\}$.

The constructed line l provides a *predicted direction* $\vec{v}$. Given the three possible edge directions $\{\vec{v}_0, \vec{v}_1, \vec{v}_2\}$ of edge e_{t+1}, we can compute angles between $\vec{v}$ and each possible direction: $\{\alpha_0, \alpha_1, \alpha_2\}$. We next derive a procedure to assign direction probabilities to each of $\{\vec{v}_0, \vec{v}_1, \vec{v}_2\}$ using computed $\{\alpha_0, \alpha_1, \alpha_2\}$.

2) Adaptive statistical model: To derive a procedure to assign probabilities to edge directions $\{\vec{v}_0, \vec{v}_1, \vec{v}_2\}$, we first consider the following. Intuitively, a closer edge direction to the predicted one (smaller angle α_i) should be assigned a higher probability than a further edge direction (larger angle). To accomplish that, we use the von Mises probability distribution, defined below, to assign probability to angle α:

$$p(\alpha|\mu, \kappa) = \frac{1}{2\pi \cdot I_0(\kappa)} \cdot e^{\kappa \cos(\alpha - \mu)} \tag{5}$$

where $I_0(.)$ is the modified Bessel function of order 0. The parameters μ and $1/\kappa$ are respectively the mean and variance in the circular normal distribution; we set $\mu = 0$ in our case. The von Mises distribution is the natural Gaussian distribution for angular measurements. We argue this is an appropriate choice because: i) it maximizes the probability when the edge direction is the same as predicted direction ($\alpha_i = 0$), and ii) it decreases symmetrically in left / right directions as the edge direction deviates from the predicted direction.

The parameter κ can be interpreted as a confidence measure: κ is larger when the predicted direction is considered more trustworthy. To quantify the confidence of a predicted direction, we first define the minimum angle $\hat{\alpha}$:

$$\hat{\alpha} = \min(\alpha_0, \alpha_1, \alpha_2) \tag{6}$$

$\hat{\alpha} = 0$ corresponds to the case when the predicted direction falls exactly on the grid, while $\hat{\alpha} = \pi/4$ corresponds to the case when the predicted direction falls in-between two edge directions.

To assign appropriate value of κ, we made the following design choice: define κ as function of $\hat{\alpha}$,

$$\kappa = \max\left(\kappa_{\min}, \rho \cdot \cos(2\hat{\alpha})\right) \tag{7}$$

where the parameter ρ is the maximum amplitude at angle 0. The intuition behind our design choice is that the predicted direction is likely more accurate when it is more aligned with the axes of the grid. When the predicted direction falls in-between two edge directions, which of the two edge directions is more likely becomes ambiguous. In this case, the positive $\kappa_{\min}$ prevents κ from becoming zero.

C. Adaptive Arithmetic Coding

Having estimated direction probabilities for each edge, we encode each edge in the MB using adaptive arithmetic coding. One important feature of arithmetic coding is that the actual encoding and modeling of the source can be completed separately [35]. Thus, we can design our own statistical model that fits our particular application and use arithmetic coding in a straight-forward manner.

In particular, for our application of lossless edge coding, we compute the direction probabilities $p_0^{(t+1)}, p_1^{(t+1)}, p_2^{(t+1)}$ of next edge e_{t+1} given observation of previous K edges $e_t, \ldots, e_{t-K+1}$, as discussed previously,

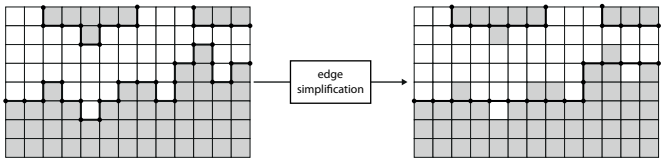


Fig. 8. Edge simplification by singularities detection. (left) Original depth boundary where localized irregularities can be seen at the pixel level. (right) Edges have been coarsely approximated by removing irregularities at 1-pixel-distance.

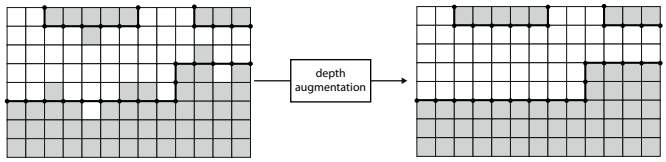


Fig. 9. Depth map has been augmented to match the coarse approximation of the boundary of Fig. 8.

and encode the true direction of e_{t+1} by sub-partitioning into the corresponding interval, as shown in Fig. 7. One approach is to encode all the detected edges in one MB as one AEC codeword. However, edges often follow an object's contour in the 3D scene, thus the entire contour can be coded into a single AEC codeword for better coding efficiency. On the other hand, z-mode may not be the best performing coding mode for all the MBs with edges along the object contour. This means modes along the object contour must be chosen jointly for RD-optimal coding performance. We postpone this discussion to Section V-C.

V. Optimizing Edge Coding

After describing the core AEC coding engine for detected sub-block boundary in a MB, in this section we discuss how AEC can be lossily encoded and optimally used in block-based depth video coding. We first discuss an edge modification strategy that slightly alters detected edges to improve edge coding efficiency. We then describe an optimization framework where z-mode in a sequence of blocks along an object's contour are selected in an RD-optimal manner.

A. Edge Simplification by Irregularities Detection

Always faithfully encoding the detected boundary losslessly may be too costly, especially at low-bitrate. When minimizing the coding rate is more a concern than signal distortion, it may be more efficient to encode a coarse approximation of the boundary, i.e., a simpler boundary that can be more easily predicted by our statistical model and requires fewer coding bits. Because of the mismatch between the original detected sub-block boundary and the actual encoded boundary, this will lead to larger prediction residuals than lossless encoding of the original boundary. However, if the larger prediction residuals are more coarsely quantized at low bitrate anyway, then the resulting increase in distortion may be small relative to the gain in boundary coding efficiency.

As discussed in Section IV, bits required to encode edges can be reduced if the geometrical structure of the boundary is well predicted. However, the boundary can contain a certain amount of localized irregularities that are hard to predict. To tackle this issue, we propose to identify those irregularities and encode the boundary disregarding them. In other words, we seek to locally reduce the variation of the boundary, while maintaining a coarse approximation close to the original boundary.

We determine how to simplify the boundary as follows. Given a sub-block dividing boundary, we identify irregularities in the boundary that temporally deviates from the current direction and return soon after. Fig. 8 illustrates such behavior where we can see one pixel or a group of two pixels at the wrong side of the boundary after simplification. As a boundary is represented through its directional chain code, we identify an irregularity as a small set of consecutive directions that temporally deviates from the current direction. We empirically limit the deviation criterion to a 1-pixel-distance, and the number of consecutive directions to five. Such limitation ensures the approximation to stay close to the original boundary. The problem of identifying the irregularities can then be casted as the problem of *pattern matching* in a sequence of symbols. According to the directional code shown in Fig. 5(c) and the described irregularity, patterns to be found in the directional chain code are: '1 – 2 – 2', '2 – 1 – 1', '1 – 2 – 0 – 2', '2 – 1 – 0 – 1'. Finally, the detected irregularities are disregarded during construction of the new boundary as illustrated in Fig. 8.

B. Depth Pixel Augmentation for View Synthesis

As previously discussed, changes in encoded sub-block boundary can lower AEC cost, but can also result in large prediction residuals. If the intended application of depth video coding is solely view synthesis at decoder, it is possible to augment the depth signal itself to match the new simpler sub-block boundary, resulting in smaller prediction residuals after MP[3]. In other words, when depth maps are intended solely for view synthesis, depth information is essentially an *auxiliary information* that assists in the processing of target images, but is not itself directly observed. That means one can pro-actively augment the ground truth depth maps as long as the augmentation does not lead to noticeable increase in target synthesized image distortion[4]. To avoid the aforementioned larger prediction residuals due to encoding of a coarse approximation of the original boundary, we

[3]We emphasize the point that we only pro-actively augment depth values when the intended application is view synthesis. For other applications where original depth maps must be faithfully represented, we perform coarse coding of edges but not depth pixel augmentation.

[4]Pro-active depth pixel augmentation was also done in [18], [19] via transform domain sparsification.

propose to augment a subset of pixels along the new sub-block boundary so that the augmented signal matches the coarse encoded version of the boundary. The pixels we deem amenable to such augmentation are termed *imprecise depth pixels*.

Unlike texture map, where edges can be blurred due to alpha-blending of foreground and background (e.g., a consequence of an out-of-focus capturing camera at that spatial location), edges in depth maps tend to be sharp. One reason can be that, during acquisition, a time-of-flight camera like [2] will return the per-pixel distance between the capturing camera and either a foreground object or background, but not a combination of both. However, in reality it is indeed possible for a depth pixel square to cover part foreground object and part background—i.e., the true foreground / background boundary is at a sub-pixel level—in which case the depth pixel suffers from imprecision no matter if it is counted as foreground or background pixel. These *imprecise pixels*, each with corresponding texture pixel that is alpha-blended, are the ones that we can optionally augment towards better edge coding efficiency.

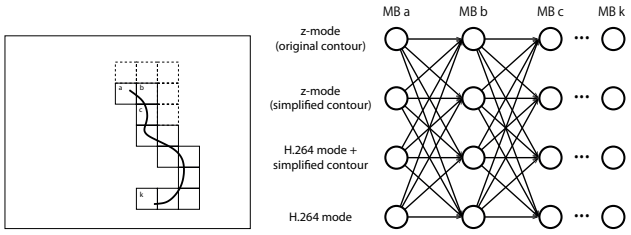
We exploit this imprecision of depth pixels to improve coding performance as follows. As the chain code representation of a simplified edge is being encoded into an arithmetic codeword, we augment an imprecise depth pixel from foreground to background (or vice versa) to lower the prediction residuals. We can classify a pixel as "augmentable" empirically by mapping the corresponding texture pixel in the same view to the corresponding location in a neighboring texture map, and calculate the resulting pixel intensity difference. As an example, in Fig. 9 several boundary depth pixels have been converted from background to foreground (and vice versa).

In more details, we perform the following procedure after a contiguous boundary of edges dividing a block into two sub-blocks has been determined, as described in the previous section. We first assume that the texture map T^v from the same viewpoint v as the target depth map Z^v (coded view) is also available at encoder. Further, texture map T^u of another nearby viewpoint u (projected view) is also available—availability of texture and disparity maps of multiple nearby viewpoints is typical in MVC setup, as described in the Introduction. For simplicity of explanation, we will assume also that synthesized camera viewpoint u is a pure horizontal shift of camera viewpoint v. That means a texture pixel $T^v(x, y)$ in view v will map to a corresponding pixel $T^u(x + \frac{t_x \cdot f}{Z^v(x,y)}, y)$ in view u, where the amount of horizon shift $\frac{t_x \cdot f}{Z^v(x,y)}$ is determined by the depth value $Z^v(x, y)$ in view v, a measure t_x of the distance between the two camera viewpoints, and the focal length f of the reference camera v.

Let pixel $Z^v(x, y)$ in view v be a *boundary pixel*; i.e., a pixel on either side of a detected edge in a code block. Suppose we reassign it the depth value of pixel $Z^v(x', y')$ on the other side of the detected edge. We can evaluate the change in synthesized distortion $d^{u,v}((x, y), (x', y'))$ due to the depth value reassignment as follows:

$$d^{v,u}((x,y),(x',y')) = |T^v(x,y) - T^u(x + \frac{t_x \cdot f}{Z^v(x,y)}, y)| - |T^v(x,y) - T^u(x + \frac{t_x \cdot f}{Z^v(x',y')}, y)| \quad (8)$$

If $d^{v,u}((x, y), (x', y'))$ is smaller than a threshold δ, then reassigning the boundary pixel from foreground to background (or vice versa) will have a minimal effect on synthesized distortion, and we can safely declare the boundary pixel as an imprecise pixel. All boundary pixels are tested in this manner to see they can be classified as imprecise pixels. After, for each imprecise pixel along the coarse approximation of the boundary, we apply a median filter including the 8-neighbor pixels on the same side of the boundary to compute its new depth value. In that way, imprecise pixels are augmented from foreground to background (or vice versa) to lower prediction residual energy.



(a) blocks along contour (b) trellis-based mode selection

Fig. 10. Coding modes for blocks along a detected contour are selected together as a group.

C. Trellis-based Optimization

As discussed in Section IV, we can code edges belonging to an object's contour that divide a block into two sub-blocks on a MB-by-MB basis (i.e., one AEC codeword for each MB), or we can code edges for a sequence of neighboring MBs along an object's contour into the same AEC codeword, as shown in Fig. 10(a). In the latter case, though the AEC coding efficiency is improved when coding a long contour of edges into one AEC codeword, z-mode may not be the RD-optimal coding mode for all the MBs along the contour. That means that the coded edges are not needed for ASMP, and are coded only to maintain continuity of the contour. On the other hand, if edges are only coded for blocks selected with z-mode, then continuity of the contour is lost, and each reinitialization of arithmetic codeword will cause coding inefficiency. In this section, we discuss an optimization strategy to optimally select the right set of edges along an object contour for AEC in a depth image for compression gain.

There are two basic ideas to our strategy. The first idea is to allow a code block that does not select z-mode for ASMP to nonetheless encode edges in the block

for the sake of contour continuity (for efficient coding of edges in the following block(s) along the contour). The second idea is to make coding mode decisions for a sequence of blocks along a contour in a dependent manner. Specifically, we make mode decisions on an image basis in a two-pass process as follows. In the first pass, we first make individual block mode decision for non-contour blocks in a conventional raster scan order, assuming each contour block (block containing edges that belong to an object's contour) is encoded in z-mode using either the original and simplified contour. This is a reasonable assumption for non-contour block mode selection, since statistically speaking, z-mode is selected for contour blocks majority of the time at medium and high bitrate.

In the second pass, the sequence of contour blocks along one detected contour are re-examined to see if better mode decisions can be made as a group of blocks. In particular, we use a trellis to formalize our block mode selections as follows. Each state represents a particular selection of coding mode—z-mode using original contour, z-mode using simplified contour (as described in Section V-A), conventional H.264/AVC mode plus simplified contour coding in the block, or H.264/AVC mode without contour coding—for a particular block x. The four states for a given block x are then stacked in a vertical column called *stage*, and the stages for different blocks are organized in the order of appearance along the contour. See Fig. 10(b) for an illustration.

To find the optimal set of modes, we find the shortest path from any state in the first stage of the trellis to any state in the last stage of the trellis, where the cost of the path is expressed in terms of a weighted sum of rate and distortion. Specifically, let $L(s_t)$ be the smallest cost sub-path from any state s_1 of the first stage to state s_t of stage t. Initially, $L(s_1)$ of a state s_1 in the first stage is simply the cost of coding the first block using a coding mode indicated by s_1:

$$L(s_1) = D(s_1) + \lambda R(s_1) \tag{9}$$

where $D(s_1)$ and $R(s_1)$ are respectively the distortion and rate when the first block is coded using mode indicated by s_1. λ is the Lagrange multiplier that trades off distortion and rate.

For state s_t in each subsequent stage t, shortest sub-path cost $L(s_t)$ is found by examining each combination of sub-path cost $L(s_{t-1})$ of the previous stage, *plus* the additional cost of coding MB t in mode indicated by s_t, for all possible s_{t-1}:

$$L(s_t) = \min_{s_{t-1}} \Big\{ L(s_{t-1}) \ + \ D(s_t) + \lambda R(s_t|s_{t-1}) \Big\} \tag{10}$$

Note that in (10), the distortion term $D(s_t)$ is independent of previous mode selected in state s_{t-1} of previous stage $t-1$. This is accurate, since the distortion depends only on the type of motion prediction (or intra coding) performed given the coding mode, each resulting in a different prediction residual, from which distortion $D(s_t)$ is computed. Rate $R(s_t|s_{t-1})$, on the other hand, depends on the mode s_{t-1} chosen in stage $t-1$. This is to capture the dependency we discussed early about AEC; if edges in block $t-1$ were coded, edges in block t can be predicted, and thus coded with higher efficiency. Otherwise, edges in block t will be encoded as a new AEC codeword, which is more coding expensive.

The complexity of the trellis-based optimization can be analyzed simply as follows. There are four states in each stage, and there are as many stages as there are MBs along an object's contour, say N. For each state s_t, the shortest sub-path cost $L(s_t)$ is computed using (10), which is $O(4)$. Hence the total complexity of the optimization is $O(4^2 \cdot N)$.

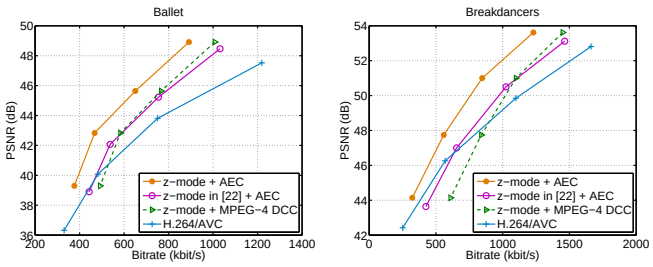
VI. Experimentation

In this section, we evaluate the performance of our proposed framework using the multiview depth video sequences `Ballet` and `Breakdancers` (1024×768 @15 Hz), provided by Microsoft [36], at the camera position 4, and sequences `bookArrival` (1024×768 @16.67 Hz) and `Undo_Dancer` (1280×720 @25 Hz) at camera position 8 and 1 respectively. For the first three sequences, the depth video provided for each camera was estimated via a color-based segmentation algorithm [37], which resulted in sharp depth boundaries. `Undo_Dancer` is a synthetic sequence with no depth estimation errors.

We implemented the proposed z-mode in the motion prediction unit of JM 18.0, where only the luminance component has been considered. The JM has been set up with the main profile and a full ME search. The maximum amplitude parameter ρ in our statistical model, as defined in (7), is set to 8. In the following, the comparison of depth compression performance is illustrated in different rate-distortion (RD) curves, where peak signal-to-noise ratio (PSNR) (of either the depth signal itself or the virtual view image synthesized using our compressed depth maps) is plotted against the bitrate (kbits/s) over 30 frames. The RD results correspond to four QP quantization parameters (QP): 32, 37, 42 and 47. Further, to measure the relative gain we used the Bjontegaard metric [9]. Synthesized views were generated using MPEG standard view synthesis software VSRS version 3.5. We note that all RD curves related to z-mode include the extra rate of sending the edges as side information. Specifically, Fig. 11 illustrates the objective PSNR of the coded depth video, while Fig. 14 and Fig. 15 show the evaluation of one synthesized view that uses the coded depth video as discussed in Section V-B. We see that in both results the addition of our z-mode in H.264/AVC results in significant compression gain. Details are discussed in next sub-sections.

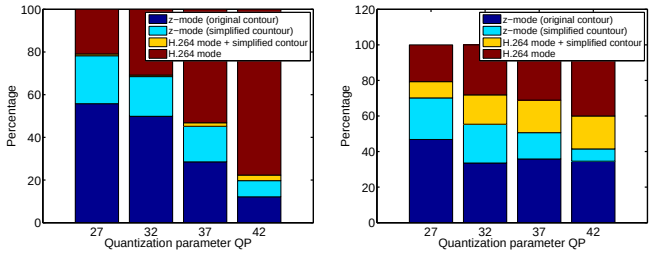
A. Depth Video Coding Evaluation

In this section, we evaluate the RD performance of depth video coding using our proposed z-mode against our earlier work in [33], edge coding scheme using



(a) (b)

Fig. 11. RD results of depth video coding. The classical H.264/AVC codec is used as baseline to evaluate the performance of the proposed framework: z-mode and the adaptive edge coding (AEC).



(a) Ballet (b) Breakdancers

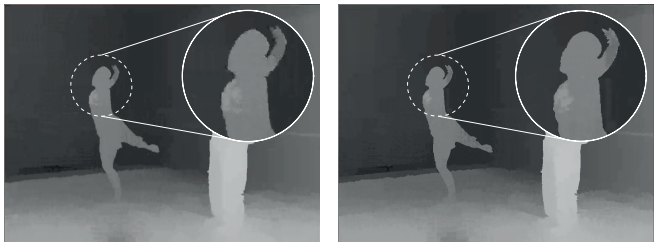
Fig. 12. Percentage of selected z-mode MBs along depth discontinuities.

MPEG-4 DCC (see details in Section VI-C) and classical modes in H.264/AVC. With respect to the Bjontegaard metric, we observe a PSNR gain up to 2.9dB and bitrate reduction up to 33.7% over the native H.264/AVC for the depth video sequence Ballet. An average PSNR gain up to 1.8dB and bitrate reduction up to 32.2% is observed for the sequence Breakdancers. Against our earlier work [33], we observe an average of 2.1dB PSNR gain and 19.7% bitrate reduction for the Ballet sequence. An average 1.9dB PSNR gain and 26.5% bitrate reduction for the Breakdancers sequence.

In particular, we observe major improvements over our earlier work at low bitrates as shown in Fig. 11. While in our earlier work we did not incorporate the edge encoding cost into the RD block mode decision, in this work we devise: i) a lossy coding of the edges, and ii) a trellis-based framework on a block-by-block basis. As a result, our proposed framework is competitive at both low and high bitrates, while our earlier work was only competitive at high bitrates.

In Fig. 12, we illustrate the percentages of selected modes for different QPs using our modified H.264/AVC software with the new z-mode implementation. We observe that our z-mode is selected more frequently at high and middle bitrates, while the cost of sending edges reduces the number of selected z-mode at low bitrate. Note also that the fraction of selected z-mode using simplified contour out of all selected z-mode becomes larger as the QP becomes coarser.

In addition to the RD performance gain, we can visually confirm the benefit of using our proposed arbitrarily



(a) classical H.264/AVC coding (b) proposed framework

Fig. 13. Example of coded depth map at QP=37.

shaped motion prediction (ASMP) in Fig. 13. We observe that the block partitioning into two non-overlapping arbitrarily shaped regions entails a much better preservation of object boundaries. Our framework then possesses an inherent edge-preserving property that will improve the performance of many depth-image-based applications, such as object detection, object tracking and DIBR-based sview synthesis, which we will demonstrate in the next section.

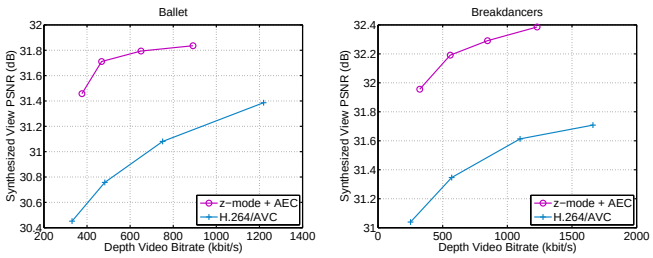
We have demonstrated the effectiveness of our proposed work against both our previous work [7] and classical prediction modes in H.264/AVC with respect to depth video coding. Because better depth map quality in general can lead to better performance for applications at decoder that use depth maps as input, by showing that we have non-negligible PSNR improvement in decoded depth maps, we have also demonstrated we have non-negligible performance improvement in the applications at decoder that depends on the quality of the decoded depth maps. In what follows, we evaluate also the impact of our scheme with respect to view synthesis quality and the performance of our arithmetic edge codec.

B. Synthesis View Quality Evaluation

In the case when the intended application is for view synthesis via DIBR, we evaluate the rendering quality of the synthesized view using the decoded depth video against the case where the depth video is encoded using the original H.264/AVC codec. Further, the evaluation is done without consideration for disoccluded regions (e.g., white regions in Fig. 16) and we assume the texture video is losslessly encoded. The purpose is to seek to evaluate the impact of depth video coding on the synthesized views, regardless of the hole-filling procedure and the texture map encoding algorithm used. Otherwise, issues such as bitrate allocation between texture and depth video should also be considered, which is an important but orthogonal research topic [38] and is outside the scope of this paper.

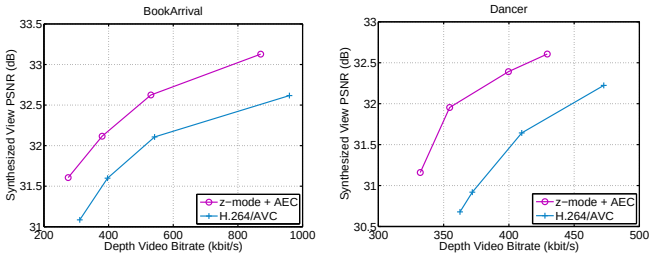
In addition to the pure depth coding gain shown in Fig. 11, the aforementioned edge-preserving property of our scheme directly reflects on the quality of the synthesized view[5] as shown in Fig. 16. Our proposal clearly

[5]It has been previously reported in [39] that coding-errors located at depth discontinuity impact on the quality of the synthesized view.



(a) (b)

Fig. 14. RD results of the synthesized video using different decoded depth video. The view synthesis is done by the projection of camera 4 onto camera 3 of the Microsoft multiview sequence.



(a) (b)

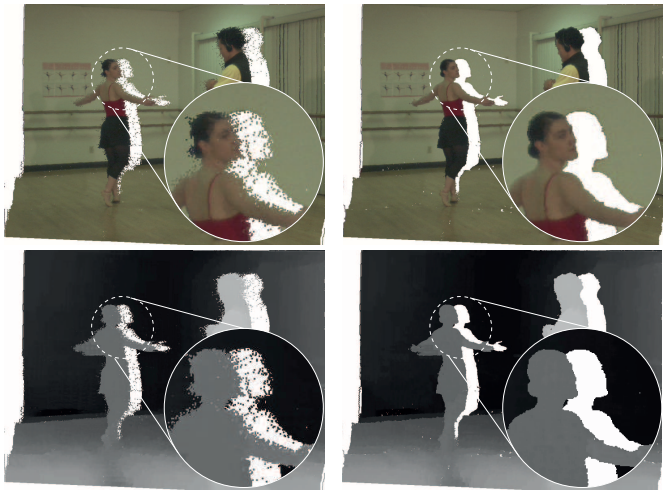
Fig. 15. RD results of the synthesized video using different decoded depth video. For `BookArrival`, we projected from camera 8 onto 10, and for `Dancer`, we projected from camera 1 to 5.

can limit depth-coding-induced-errors in the synthesized view, while a standard depth video coding fails at low bitrates. As a result of sharp edge preservation in our proposal, objectively we observe about 0.8dB PSNR gain for both sequence `Ballet` and `Breakdancers` as shown in Fig. 14, and about 0.7dB and 1.2dB gain respectively for sequences `BookArrival` and `Undo_Dancer` as shown in Fig. 15. We conjecture that the better results obtained for `Undo_Dancer` is due to its better quality depth video. This means that as technologies of depth sensor improve over time resulting with cleaner depth maps, the gain of our proposal can be expected to be more significant.

C. Edge coding evaluation

There have been various proposals on shape coding in MPEG-4 standard [40]. A notable lossless coding approach that relies on the boundary representation is the chain-code-based shape encoders [41]. Experiments conducted in MPEG-4 working group confirmed that DCC has higher efficiency lossless coding than a normal chain coding, with an average of 1.2 bits/boundary pel and 1.4 bits/boundary pel for a 4- and 8-connected chain, respectively [40].

In what follows, we compare our proposed AEC with the classical DCC utilized in MPEG-4, which can be considered as the current state-of-the-art for lossless boundary coding. In addition, we compare our AEC implementation with the iid and non-iid assumption of the source. With the iid case, one pmf is computed for all the edges and transmitted as overhead to the decoder.



(a) H.264/AVC (b) proposed framework

Fig. 16. Example of projected texture and depth map from camera 4 to camera 3, where the reference depth map has been coded using (a) H.264/AVC with visible depth-coding-induced-errors and (b) the proposed edge-preserving framework. Disoccluded regions are represented in white color.

TABLE I
AVERAGE EDGE RATE IN BITS/BOUNDARY PEL (BPP)

	MPEG-4 DCC	iid AEC	non-iid AEC
`Ballet`	1.2 bpp	1.36 bpp	**0.27 bpp**
`Breakdancers`	1.2 bpp	1.43 bpp	**0.31 bpp**
`Bookarrival`	1.2 bpp	1.26 bpp	**0.33 bpp**
`Undo_Dancer`	1.2 bpp	1.17 bpp	**0.24 bpp**

Non-iid AEC corresponds to the scheme proposed in this paper. Entire contour in a frame was encoded, which was not a significant loss of coding efficiency since our *z-mode* was very often selected as the optimal mode for MB encoding at the boundary of objects.

As shown in Table I our scheme clearly outperformed the current state-of-the-art MPEG-4 DCC by a factor of 4 for all four test sequences. In addition, the comparison with the assumption of an iid model confirmed the benefit of our proposed statistical model used in adaptive arithmetic coding.

VII. Conclusion

In this paper we proposed two main contributions: i) a novel, precise and compact motion prediction method that partitions a block into two non-overlapping sub-blocks for separate motion estimation and compensation, which we denoted as arbitrarily shaped motion prediction (ASMP), and ii) an entropy coding scheme to compress the dividing boundary as side information needed at the decoder side. Experimental results show that these two contributions can bring about substantial coding gain. In addition, our scheme better preserves ob-

jects boundaries at both high and low bitrate. This edge-preserving property enhances the performance of depth-based image processing applications, such as DIBR view synthesis. Further, we demonstrated that our arithmetic edge coding scheme is more coding-efficient than current state-of-the-art.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multi-view imaging and 3DTV," in *IEEE Signal Processing Magazine*, vol. 24, no.6, November 2007.

[2] S. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor—system description, issues and solutions," in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, Washington, DC, June 2004.

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

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

[5] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no.7, July 2003, pp. 560–576.

[6] P. Merkle, C. Bartnik, K. Muller, D. Marpe, and T. Weigand, "3D video: Depth coding based on inter-component prediction of block partitions," in *2010 Picture Coding Symposium*, Krakow, Poland, May 2012.

[7] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped sub-block motion prediction in texture map compression using depth information," in *29th Picture Coding Symposium*, Krakow, Poland, May 2012.

[8] G. Cheung, A. Ortega, and T. Sakamoto, "Fast H.264 mode selection using depth information for distributed game viewing," in *IS&T/SPIE Visual Communications and Image Processing (VCIP'08)*, San Jose, CA, January 2008.

[9] G. Bjontegaard, "Calculation of average PSNR differences between RD curves," Austin, TX, USA, Apr. 2001, ITU SC16/Q6, 13th VCEG Meeting, Austin, Texas, USA, VCEG-M33 doc.

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

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

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

[13] J. Gautier, O. L. Meur, and C. Guillemot, "Efficient depth map compression based on lossless edge coding and diffusion," in *29th Picture Coding Symposium*, Krakow, Poland, May 2012.

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

[15] V.-A. Nguyen, D. Min, and M. N. Do, "Efficient techniques for depth video compression using weighted mode filtering," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no.2, February 2013, pp. 189–202.

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

[17] ——, "Depth map coding with distortion estimation of rendered view," in *SPIE Visual Information Processing and Communication*, San Jose, CA, January 2010.

[18] G. Cheung, A. Kubota, and A. Ortega, "Sparse representation of depth maps for efficient transform coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

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

[20] G. Cheung, W. s. Kim, A. Ortega, J. Ishida, and A. Kubota, "Depth map coding using graph based transform and transform domain sparsification," in *IEEE International Workshop on Multimedia Signal Processing*, Hangzhou, China, October 2011.

[21] H. Oh and Y.-S. Ho, "H.264-based depth map sequence coding using motion information of corresponding texture video," in *The Pacific-Rim Symposium on Image and Video Technology*, Hsinchu, Taiwan, December 2006.

[22] I. Daribo, C. Tillier, and B. Pesquet-Popescu, "Motion vector sharing and bit-rate allocation for 3D video-plus-depth coding," in *EURASIP: Special Issue on 3DTV in Journal on Advances in Signal Processing*, vol. 2009 (2009), January 2009.

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

[24] H. A. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, May 2014.

[25] Y. Gao, G. Cheung, T. Maugey, P. Frossard, and J. Liang, "3D geometry representation using multiview coding of image tiles," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, May 2014.

[26] H. Helgason, H. Li, and M. Flierl, "Multiscale framework for adaptive and robust enhancement of depth in multi-view imagery," in *IEEE International Conference on Image Processing*, Orlando, FL, October 2012.

[27] R. Li, D. Rusanovskyy, M. Hannuksela, and H. Li, "Joint view filtering for multiview depth map sequences," in *IEEE International Conference on Image Processing*, Orlando, FL, October 2012.

[28] W. Sun, G. Cheung, P. Chou, D. Florencio, C. Zhang, and O. Au, "Rate-distortion optimized 3d reconstruction from noise-corrupted multiview depth videos," in *IEEE International Conference on Multimedia and Expo*, San Jose, CA, July 2013.

[29] E. Hung, R. D. Queiroz, and D. Mukherjee, "On macroblock partition for motion compensation," in *IEEE International Conference on Image Processing*, Atlanta, GA, October 2006.

[30] R. Ferreira, E. Hung, R. D. Queiroz, and D. Mukherjee, "Efficiency improvements for a geometric-partition-based video coder," in *IEEE International Conference on Image Processing*, Cairo, Egypt, November 2009.

[31] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, no. 2, pp. 260–268, 1961.

[32] A. Katsaggelos, L. Kondi, F. Meier, J. Ostermann, and G. Schuster, "MPEG-4 and rate-distortion-based shape-coding techniques," in *Proceedings of the IEEE*, vol. 86, no.6, June 1998, pp. 1126–1154.

[33] I. Daribo, G. Cheung, and D. Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video coding," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[34] W. J. Teahan, "Probability estimation for PPM," in *In the Proc. of the New Zealand Computer Science Research Students' Conference*, 1995.

[35] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no.7, July 2003, pp. 620–635.

[36] "Microsoft sequence ballet and breakdancers," 2004, [Online] Available: http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/.

[37] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *The 31st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 23, no. 3, pp. 600–608, Aug. 2004.

[38] G. Cheung, V. Velisavljevic, and A. Ortega, "On dependent bit allocation for multiview image coding with depth-image-based rendering," in *IEEE Transactions on Image Processing*, vol. 20, no.11, March 2011, pp. 1109–1126.

[39] I. Daribo and H. Saito, "Influence of wavelet-based depth coding in multiview video systems," in *28th Picture Coding Symposium*, 2010, pp. 334–337.
[40] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster, "MPEG-4 and rate-distortion-based shape-coding techniques," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1126–1154, 1998.
[41] M. Eden and M. Kocher, "On the performance of a contour coding algorithm in the context of image coding part I: Contour segment coding," *Signal Processing*, vol. 8, no. 4, pp. 381–386, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0165168485900015

Ismael Daribo (M'10) received B.Sc., M.Sc./M.Eng. and Ph.D. degrees in specialty of Multimedia, Computer Science, Image and Signal Processing in 2002, 2005 and 2009, respectively.From 2010 to 2013, he has been a specially appointed Assistant Professor and Visiting Research Scientist in Japan, where he became a specialist in 3D Video Communication Systems and related Signal Processing.

His research interests include all technical aspects of 3D video communication end-to-end services, including multiple viewpoint camera acquisition, 3D video data representation, and 3D video coding and image-based rendering on 3D displays.

Dinei Florencio (M'97—SM'05) has been a Researcher with Microsoft Research, Redmond, WA, USA, since 1999. He received the B.S. and M.S. degrees from the University of Brasilia, Brasilia, Brazil, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA, USA, all in electrical engineering. Before joining Microsoft, he was a Member of the Research Staff at the David Sarnoff Research Center from 1996 to 1999. He was also a Co-Op Student with the AT&T Human Interface Laboratory (now part of NCR) from 1994 to 1996, and a Summer Intern at Interval Research Corporation, Palo Alto, CA, USA, in 1994. He has authored over 70 papers, and holds 54 granted U.S. patents.

Dr. Florencio was the General Co-Chair of MMSP 2009, WIFS 2011, Hot3D 2010 and 2013, and Technical Co-Chair of WIFS 2010, ICME 2011, and MMSP 2013. He is the Chair of the IEEE SPS Technical Committee on Multimedia Signal Processing (from 2014 to 2015). He is also an Elected Member of the IEEE SPS Technical Committee on Information Forensics and Security, and an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.

Gene Cheung (M'00—SM'07) received the B.S. degree in electrical engineering from Cornell University in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1998 and 2000, respectively.

He was a senior researcher in Hewlett-Packard Laboratories Japan, Tokyo, from 2000 till 2009. He is now an associate professor in National Institute of Informatics in Tokyo, Japan.

His research interests include image & video representation, immersive visual communication and graph signal processing. He has published over 140 international conference and journal publications. He has served as associate editor for IEEE Transactions on Multimedia from 2007 to 2011 and currently serves as associate editor for DSP Applications Column in IEEE Signal Processing Magazine, APSIPA Journal on Signal & Information Processing and SPIE Journal of Electronic Imaging, and as area editor for EURASIP Signal Processing: Image Communication. He currently serves as member of the Multimedia Signal Processing Technical Committee (MMSP-TC) in IEEE Signal Processing Society (2012-2014). He has also served as area chair in IEEE International Conference on Image Processing (ICIP) 2010, 2012-2013, technical program co-chair of International Packet Video Workshop (PV) 2010, track co-chair for Multimedia Signal Processing track in IEEE International Conference on Multimedia and Expo (ICME) 2011, symposium co-chair for CSSMA Symposium in IEEE GLOBECOM 2012, and area chair for ICME 2013. He is a co-author of best student paper award in IEEE Workshop on Streaming and Media Communications 2011 (in conjunction with ICME 2011), best paper finalists in ICME 2011 and ICIP 2011, best paper runner-up award in ICME 2012, and best student paper award in ICIP 2013.