ARITHMETIC EDGE CODING FOR ARBITRARILY SHAPED SUB-BLOCK MOTION PREDICTION IN DEPTH VIDEO COMPRESSION

Ismael Daribo #, *Gene Cheung* #, *Dinei Florencio* o

National Institute of Informatics, o Microsoft Research

ABSTRACT

Depth map compression is important for compact representation of 3D visual data in "texture-plus-depth" format, where texture and depth maps of multiple closely spaced viewpoints are encoded and transmitted. A decoder can then freely synthesize any chosen intermediate view via depth-image-based rendering (DIBR) using neighboring coded texture and depth maps as anchors. In this work, we leverage on the observation that "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 sub-blocks along the dividing edge before performing separate motion prediction. While doing such arbitrarily shaped sub-block motion prediction can lead to very small prediction residuals (resulting in few bits required to code them), it incurs an overhead to losslessly encode dividing edges for sub-block identification. To minimize this overhead, we first devise an edge prediction scheme based on linear regression to predict the next edge direction in a contiguous contour. From the predicted edge direction, we assign probabilities to each possible edge direction using the von Mises distribution, which are subsequently inputted to a conditional arithmetic codec for entropy coding. Experimental results show an average overall bitrate reduction of up to 30% over classical H.264 implementation.

Index Terms— Depth-image-based rendering, depth map coding, arithmetic coding, edge coding

1. INTRODUCTION

Among many proposed formats for representing 3D visual data is "texture-plus-depth" [1], where texture maps (e.g., RGB images) and depth maps (per-pixel physical distances between capturing camera and the captured objects in the 3D scene) of multiple closely spaced viewpoints are encoded and transmitted. At the receiver, the decoded texture and depth maps, can then be used to freely synthesize any chosen intermediate view via a depth-image-based rendering technique like 3D warping [2]. Transmitting multiple large texture and depth maps from sender to receiver, however, incurs a high network transmission cost, which is not desirable. Thus, compression of 3D data in texture-plus-depth format is important. Compression of texture video is well studied during the past decades. We focus instead on compression of depth video in this paper. Towards the goal of efficient depth video coding, one can first observe that in general captured video, pixels of similar depth tend to belong to the same physical object and hence have similar motion [3]. In depth video coding, by definition per-pixel depth information is available. Thus, for a given code block, one can divide a code block containing two zones of distinct values (e.g., foreground and background) into two sub-blocks along the dividing boundary, before performing motion prediction (MP) separately for each of the sub-blocks. While doing such arbitrarily shaped sub-block motion prediction can lead to very small prediction residuals (resulting in few bits required to code them), it incurs an overhead to losslessly encode the dividing boundary for sub-block identification.

To minimize this overhead, in this paper we propose a lossless arithmetic edge coding (AEC) scheme for arbitrarily shaped sub-block motion prediction 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. Experimental results show an average overall bitrate reduction of up to 30% over classical H.264 implementation.

The outline of the paper is as follows. We first discuss related work in Section 2. We then discuss the arbitrarily shaped sub-block MP scheme for depth video in Section 3. We discuss our proposed arithmetic edge coding algorithm in Section 4. Experimental results and conclusions are presented in Section 5 and 6, respectively.

2. RELATED WORK

MP in H.264 [4] 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 coding overhead. 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 [5, 6] 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 often is 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 contour follows the object's boundary, we can easily segment a code block along object boundary with pixel-level accuracy. Moreover, the depth edge can be acquired cheaply via simple edge detection techniques.

The observation that pixels of similar depth have similar motion has been made previously [3], where unlikely coding modes are eliminated *a priori* for faster H.264 encoding. We focus instead on MP of arbitrarily shaped sub-blocks in depth maps for efficient depth video coding.

In our previous work [7], in order to efficiently perform MP in *texture* maps, we proposed to use available depth edges (detected in corresponding encoded depth maps of the same viewpoint) for sub-block partition and MP. While the notion of arbitrarily shaped sub-block MP is the same, our current work is more challenging, since the overhead in depth edge coding has to be accounted for during depth video coding. Further, our proposed edge coding scheme,

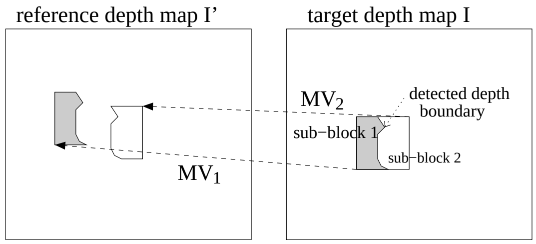


Fig. 1. Motion compensation of sub-blocks in depth map divided using detected edges.

which shows significant compression performance gain compared to existing schemes, can be used directly in recent popular image / video coding schemes based on edge-adaptive wavelets [8], graph-based transform [9] and graph-based wavelets [10], where edge coding is paramount in determining the overall compression performance.

3. SUB-BLOCK MOTION PREDICTION

We now overview *z-mode*, the arbitrarily shaped sub-block MP scheme we introduced in [7], as an additional MP mode in a H.264-like codec. The main idea behind *z-mode* is to partition a 16×16 coding block (macroblock or MB) into two arbitrarily shaped regions for separate motion estimation and compensation as illustrated in Fig. 1. 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 blocks in current frame.
3. For each sub-block, perform motion estimation; i.e., find the best-matched sub-block in reference frame to the current target sub-block.
4. Compute prediction residuals for the current MB given the two motion-compensated sub-blocks for residual encoding.

We briefly describe step 1 and 2 next. Step 3 and 4 are similar to standard MP performed in H.264.

3.1. Macroblock Partitioning

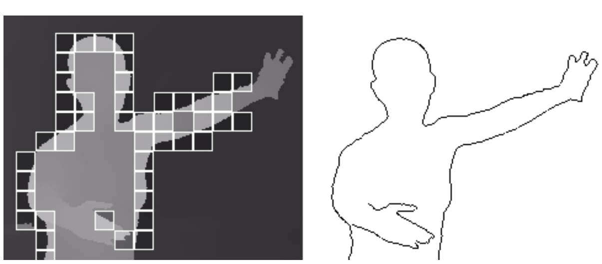


Fig. 2. Example of z-mode MBs for `Ballet` sequence: segmentation of depth MBs (left) around the boundary of dancer (right).

A given target depth code block is first partitioned using a simple edge detection technique: we first compute the arithmetic mean depth value of the block, and divide the block into sub-blocks with depth values above and below the mean. This statistical approach requires no thresholding (like the well used Canny edge detector), is robust, and has very low complexity. See Fig. 2 for an example of boundaries detected by our method in z-mode blocks (left), which matches well the actual object boundary (right).

3.2. Motion Vector Prediction

Leveraging again on the observation that pixels with similar depth have similar motion, we next compute the predicted motion vector of a sub-block using motion vectors of neighboring (sub-)blocks with similar depth values that have already been encoded. Specifically, PMV is the weighted sum of surrounding causal coded blocks' MVs, where the weights are computed using a Laplacian distribution, with argument being the difference in mean depth values between the target and predictor blocks.

We note that the encoded bitstream can only be correctly decoded if both encoder and decoder have the same block partition information. In the case of texture coding, we proposed to use the encoded depth information to divide a given MB, assuming that the depth data is first encoded and transmitted, so that the depth-based partitioning information is available to both encoder and decoder [7]. When encoding depth video, however, the block partition information has to be explicitly encoded. In the following we propose an arithmetic edge encoding scheme for this purpose.

4. ARITHMETIC EDGE CODING

In this section, we address the problem of losslessly encoding the boundary that separates a *z-mode* MB[1] into two sub-blocks. The overall coding scheme can be summarized in the following steps:

1. Given a depth MB with discontinuities (i.e. high block-wise depth variance), 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 [11].
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.

We discuss these steps in order next.

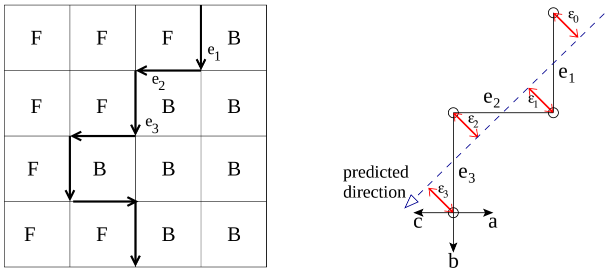
4.1. Differential Chain Code

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. 3(a) for an illustration of a boundary in a 4×4 block. As shown, the set of edges composing the boundary is a *4-connected chain code*; i.e., next edge e_{t+1} starts at the location where current edge e_t terminates, and e_{t+1} can only take on three possible directions: *forward* (continue the same direction as previous e_t), *left*, and *right* (as compared to previous e_t's direction). This boundary representation is also known as *differential chain code* (DCC), which belongs to the family of chain coding schemes pioneered by Freeman [11].

4.2. Edge Prediction

Given the definition of DCC, edges e_t's can be entropy-encoded consecutively, where each edge has an alphabet of three symbols representing the three possible edge directions. There are many options for entropy coding. One notable example is *prediction by partial*

[1] 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. 3. Detected edges in a 4×4 depth pixel block and linear prediction using previous three edges.

matching (PPM) [12], 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, unlike a generic text message, 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.

4.2.1. *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. 3(b) for an illustration where a line-of-best-fit is drawn to minimize squared error sum $\sum_{i=0}^{3} \epsilon_i^2$ given window of edges $\{e_1, e_2, e_3\}$.

The constructed line l provides a *predicted direction* $\vec{v}$. Given the three possible edge directions $\{\vec{v}_a, \vec{v}_b, \vec{v}_c\}$ of edge e_{t+1}, we can compute angles between $\vec{v}$ and each possible direction: $\{\alpha_a, \alpha_b, \alpha_c\}$. We next derive a procedure to assign direction probabilities to each of $\{\vec{v}_a, \vec{v}_b, \vec{v}_c\}$ using computed $\{\alpha_a, \alpha_b, \alpha_c\}$.

4.2.2. *Adaptive statistical model*

To derive a procedure to assign probabilities to edge directions $\{\vec{v}_a, \vec{v}_b, \vec{v}_c\}$, 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:

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

where $I_0\left(.\right)$ 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.

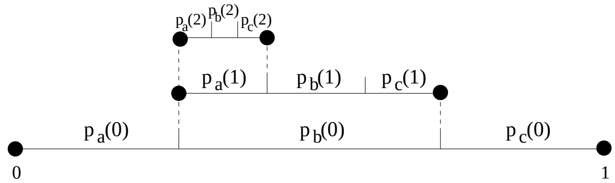


Fig. 4. Example of arithmetic edge coding, where the first two symbols to be encoded are b and a.

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 $\hat{\alpha}$:

$$\hat{\alpha} = \min\left(\alpha_a, \alpha_b, \alpha_c\right) \tag{2}$$

$\hat{\alpha} = 0$ 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 $\hat{\alpha}$,

$$\kappa = \rho \cdot \cos\left(2\hat{\alpha}\right) \tag{3}$$

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.

4.3. Adaptive Arithmetic Coding

Having estimated direction probabilities for each edge, we encode each edge using adaptive arithmetic coding. One important feature of arithmetic coding is that the actual encoding and modeling of the source can be completed separately. 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 $p_a(t+1), p_b(t+1), p_c(t+1)$ of next edge e_{t+1} given observation of previous K edges $e_t, \ldots, e_{t-K+1}$, as discussed previously, and encode the true direction of e_{t+1} by sub-partitioning into the corresponding interval, as shown in Fig. 4. Note that the derivation of direction probabilities for edge e_{t+1} can be mimicked at decoder, and hence no extra information needs to be sent for correct decoding.

5. EXPERIMENTATION

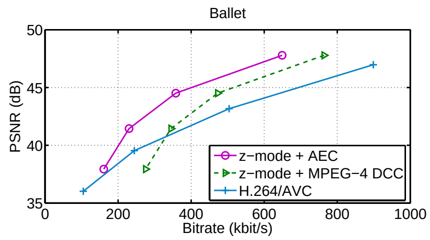
The performance of the proposed framework was evaluated using the multiview depth video sequences `Ballet` and `Breakdancers` (1024×768 @15 Hz) provided by Microsoft at the camera position 4. The depth video provided for each camera was estimated via a color-based segmentation algorithm.

5.1. Edge Coding Performance

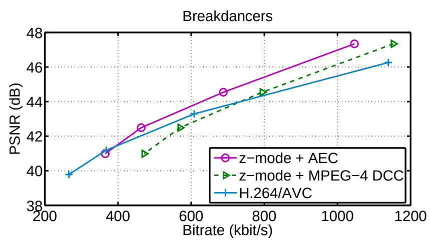
There have been various proposals on shape coding in MPEG-4 standard [13]. A notable lossless coding approach that relies on the boundary representation is the chain-code-based shape encoders [14]. Experiments conducted in MPEG-4 working group confirmed that DCC has higher efficiency lossless coding 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 [13].

Table 1. Average edge rate in bits/boundary pel (bpp)

	MPEG-4 DCC	iid AEC	non-iid AEC
Ballet	1.2 bpp	1.49 bpp	**0.29 bpp**
Breakdancers	1.2 bpp	1.57 bpp	**0.34 bpp**



(a)



(b)

Fig. 5. Objective PSNR results of depth coding.

In what follows, we compare our proposed AEC with the classical DCC utilized in MPEG-4 that 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. 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 1 our scheme clearly outperformed the current state-of-the-art MPEG-4 DCC by a factor of 4. In addition, the comparison with the assumption of an iid model confirmed the benefit of our proposed statistical model used in adaptive arithmetic coding. In this work, the spatio-temporal correlation of the starting point of each chain code has not been taken into consideration, and hence more gains can expected in the future.

5.2. Depth Video Coding Performance

The comparison of objective compression performance is illustrated in the rate-distortion (RD) curves plotted in Fig. 5, where the peak signal-to-noise ratio (PSNR) of the coded depth video was plotted against bitrate (kbits/s) over 100 frames. The RD results correspond to four QP quantization parameters: 27, 32, 37 and 42.

We implemented the proposed z-mode in JM 18.0. In motion estimation, only luminance component was considered. The maximum amplitude parameter ρ in our statistical model, as defined in (3), was set to 8. We see that the addition of our z-mode in H.264 resulted in significant compression gain. Specifically, an average bitrate reduction of up to 30% and 20% for the depth video sequence `Ballet` and `Breakdancers`, respectively, was observed.

6. CONCLUSION

In this paper we proposed a lossless arithmetic edge coding scheme for arbitrarily shaped sub-block motion prediction in depth video coding. We first partition a given depth block into two non-overlapping, arbitrarily shaped sub-blocks for separate motion prediction. The overhead from the boundary representation is encoded through a new arithmetic edge coding scheme. We designed a statistical model that captures the geometric correlation in the edges. The computed edge direction probabilities are inputted to an adaptive arithmetic coder. Experimental results show an overall bitrate reduction of up to 30% over classical H.264 implementation.

Acknowledgment

This work is partially supported by the Japan Society for the Promotion of Science (JSPS) Postdoctoral Program for Foreign Researchers.

7. REFERENCES

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

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

[3] 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.

[4] 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*, July 2003, vol. 13, no.7, pp. 560–576.

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

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

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

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

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

[10] E. Martinez-Enriquez, F. Diaz de Maria, and A. Ortega, “Video encoder based on lifting transforms on graphs,” in *IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011.

[11] Herbert Freeman, “On the encoding of arbitrary geometric configurations,” *IRE Transactions on Electronic Computers*, , no. 2, pp. 260–268, 1961.

[12] W. J. Teahan, “Probability estimation for PPM,” in *In the Proc. of the New Zealand Computer Science Research Students’ Conference*, 1995.

[13] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster, “MPEG-4 and rate-distortion-based shape-coding techniques,” vol. 86, no. 6, pp. 1126–1154, 1998.

[14] Murray Eden and Michel 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.