ARBITRARILY SHAPED SUB-BLOCK MOTION PREDICTION IN TEXTURE MAP COMPRESSION USING DEPTH INFORMATION

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

[#] National Institute of Informatics, [o] Microsoft Research

ABSTRACT

When transmitting the so-called "*texture-plus-depth*" video format, texture and depth maps from the same viewpoint exhibit high correlation. Coded bits from one map can then be used as side information to encode the other. In this paper, we propose to use the depth information to divide the corresponding block in texture map into arbitrarily shaped regions (sub-blocks) for separate motion estimation (ME) and motion compensation (MC). We implemented our proposed sub-block motion prediction (MP) method for texture map coding using depth information as a new coding mode (z-mode) in H.264. Nonetheless, in practical experiments one can observe either a misalignment between texture and depth edges, or an aliasing effect at the texture boundaries. To overcome this issue, z-mode offers two MC types: i) non-overlapping MC, and ii) overlapping MC. In the latter case, overlapped sub-blocks after ME are alpha-blended using a properly designed filter. Moreover, the MV of each sub-block in z-mode is predicted using a Laplacian-weighted average of MVs of neighboring blocks of similar depth. Experimental results show that using z-mode, coding performance of the texture map can be improved by up to 0.7dB compared to native H.264 implementation at high bitrate.

***Index Terms*—** Depth-image-based rendering, motion prediction, arbitrary shape, video compression

1. INTRODUCTION

Three dimensional (3D) displays have recently become widely available at a consumer level. The particular method of delivering visual information to those devices is, however, still a subject of intense research. The simple solution of sending stereo video (i.e., a left and a right view) suffices to provide the novelty aspects of the 3D experience but has significant limitations. Among more advanced solutions, a format known as *texture-plus-depth* has received significant attention due to its flexibility. Besides traditional texture maps (e.g., RGB images), this format sends corresponding depth maps (i.e., per-pixel physical distances between camera and the locations of the scene objects) of the same viewpoint. These depth maps provide the decoder with geometric information of the captured scene for view synthesis via depth-image-based rendering (DIBR) [1]. In particular, it allows the receiver to generate a second slightly shifted view using texture and depth maps of the first view for stereo viewing, resulting in a visual perception of depth.

Texture-plus-depth format can also be used for coding of multiview video, where texture and depth maps of multiple camera viewpoints are coded and transmitted. After decoding at receiver, any desired intermediate view can be synthesized using texture and depth maps of two neighboring camera-captured views via DIBR. This results in a smooth view transition from one camera-captured view to another, benefiting applications such as immersive video conferencing and free viewpoint TV [2]. Moreover, texture-plus-depth format provides information to assist in the creation of mixed content, e.g., when synthetic elements are added to a received video, at the correct depth and perspective. However, encoding and transmitting both texture and depth maps of captured views can incur a high transmission cost. In this paper, we address the problem of texture map compression in the context of texture-plus-depth format.

Because in texture-plus-depth format both texture and depth maps from the same viewpoint need to be compressed and transmitted, there exists strong correlation between them that can be exploited for coding gain. More specifically, coded bits from one map can be used as side information to encode the other, reducing the total required bitrate in the process. In this paper, we propose to use coded depth information to divide a given code block in texture map into sub-blocks for separate motion prediction (MP): motion estimation (ME) followed by motion compensation (MC). The key observation is that *pixels of similar depth have similar motion*. Thus, pixels on one side of a depth edge (detected in encoded depth map) in a texture block will typically have similar depth values and motion; representing their motion field with a single unique motion vector (MV) will lead to a smaller prediction residual, resulting in coding gain compared to coding the entire block using a single MV. The traditional solution is to divide a code block into many small rectangular blocks (as done in H.264 [3]). This may also result in a small prediction residual. However, besides never quite matching the boundary contour, these small blocks imply a much more complex representation (thus more required encoded bits).

We implemented our proposed sub-block motion prediction method for texture map coding using depth information as a coding mode (called z-mode) in H.264. In z-mode, each 16×16 macroblock is divided into two arbitrarily shaped regions (sub-blocks), formed by the pixels with depth value above (below) the mean-depth value for the block. Nonetheless, in practical experiments one can observe either a misalignment between texture and depth edges, or an aliasing effect at the texture boundaries. To overcome this issue, z-mode offers two MC types: i) non-overlapping MC, and ii) overlapping MC, while the ME utilizes only non-overlapping subregions. Overlapped sub-blocks are alpha-blended using a properly designed filter. The MV of each sub-block in z-mode is predicted using a Laplacian-weighted average of MVs of neighboring blocks of similar depth. Experimental results show that using our proposed z-mode, coding performance of the texture map can be improved by up to 0.7dB compared to native H.264 implementation at high bitrate, when intra mode is disabled.

The outline of the paper is as follows. We first overview related work in Section 2. We then discuss our proposal of sub-block motion compensation using depth edges in Section 3. Prediction of sub-block MVs is discussed in Section 4. Results and conclusion are presented in Section 5 and 6, respectively.

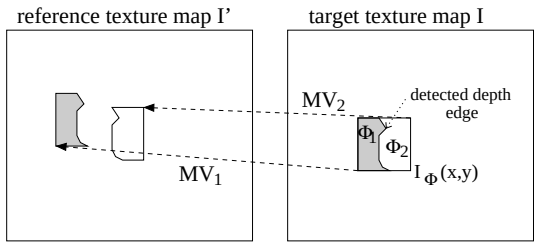


Fig. 1. Motion compensation of sub-blocks in texture map divided using detected edge in depth map.

2. RELATED WORK

There are several proposals in the literature [4, 5, 6] to exploit the inherent correlation between texture and depth maps of the same viewpoint in texture-plus-depth format for coding gain. [4] shared MVs between texture and depth maps to lower the overhead required for coding motion information of both maps. [5] extended the notion of motion information sharing further, by selecting a single MV that minimizes the energy of prediction residuals in texture and depth blocks simultaneously during ME. For edge-adaptive wavelet coding, [6] reused the encoded edge information in depth map for texture map as well to lower the overall bitrate required to code edges in both maps. We differ in that we use detected depth edges to segment a texture code block into arbitrarily shaped sub-blocks for separate ME and MC.

While H.264 [3] offers different block sizes (rectangular blocks from 16×16 down to 4×4) for MP, to accurately track the motion of an arbitrary-shaped object in a code block, many small sub-blocks are needed to accurately compose the object, resulting in a substantial overhead in coding a large set of MVs. In light of this problem, line-based segmentation schemes [7, 8] that divide a code block using any arbitrary straight line segment that cuts across the code block have been proposed. There are two problems, however: i) the shape of the moving object may not follow a straight line, resulting in shape-mismatch; and ii) the computation required to search for a RD-optimal dividing line segment can be prohibitively expensive. In our proposal, because the detected depth edge follows the shape of the object[1], we can segment a code block along object boundary with pixel-level accuracy. Moreover, the depth edge can be acquired simply without complex computation.

Note that the observation of pixels of similar depth having similar motion has been previous made in [9], where unlikely coding modes are eliminated a priori for faster H.264 encoding. We focus instead of MP of arbitrarily shaped sub-blocks in texture map divided by detected depth edges.

3. EDGE-BASED MOTION COMPENSATION

We first overview the system setup. We assume (one or more) pairs of texture and depth maps, one pair for each captured viewpoint, must be compressed across time for transmission. Texture and depth maps from the same viewpoint are of the same resolution. We assume depth maps are captured or estimated to be of sufficiently high quality, that a detected depth edge will roughly correspond to the boundary of a foreground object in the texture map.

[1]As discussed, unlike sharp depth edges, texture edges may be blurred due to out-of-focus camera capture. We discuss overlapping MC within our proposed z-mode to address this issue in Section 3.

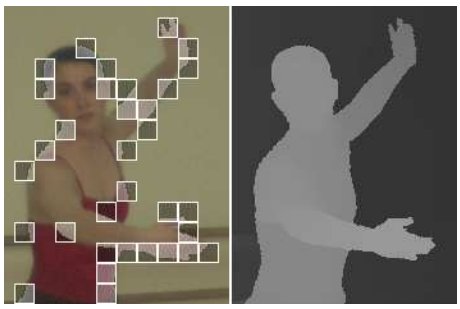


Fig. 2. Example of z-mode MBs for `Ballet` sequence in high bitrate: segmentation of texture MBs around boundary of dancer into sub-blocks (left) using corresponding depth data (right).

As discussed in the Introduction, we propose a new coding mode in H.264, called *z-mode*, to partition a 16×16 coding block (macroblock or MB) into two sub-blocks given an arbitrary boundary (detected using the block's depth information) for separate motion estimation and compensation. z-mode consists of the following parts:

- Partition the current MB into two sub-blocks.
- For each sub-block, perform motion estimation; i.e., find best-matched sub-block in reference frame to current sub-block.
- Compute prediction residuals for the current MB given the two motion-compensated sub-blocks for residual encoding.
- Compute a predicted motion vector (PMV) for each sub-block using MVs of neighboring blocks in current frame.

Before discussing the details in each step, we first define notations for ease of discussion. Considering a texture map I of a certain viewpoint and its corresponding per-pixel depth map Z. 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 texture and depth maps are $I_\Phi(x, y)$ and $Z_\Phi(x, y)$, respectively. 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 $\emptyset = \Phi_1 \cap \Phi_2$, so that $I_\Phi(x, y) = I_{\Phi_1}(x, y) \cup I_{\Phi_2}(x, y)$. See Fig. 1 for an illustration.

3.1. Macroblock Partitioning

The first 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 texture MB $I_\Phi(x, y)$ into foreground $I_{\Phi_1}(x, y)$ and background $I_{\Phi_2}(x, y)$. This statistical approach is robust and of very low complexity, with accurate empirical results as shown in Fig. 2.

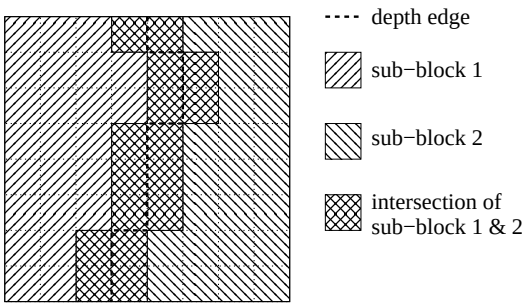


Fig. 3. Example of overlapping motion compensation.

3.2. Sub-block Motion Estimation

As similarly done in H.264, for each segmented sub-block $I_{\Phi_s}(x,y)$, $s \in \{1,2\}$, we search in the reference texture frame I' for a "best-matched" sub-block $I'_{\Phi_s}(m,n)$. By "best-matched", we mean a sub-block that minimizes a metric that measures the pixel difference between predictor sub-block $I'_{\Phi_s}(m,n)$ in reference frame I' and target sub-block $I_{\Phi_s}(x,y)$ in current frame I within a defined search space:

$$\min_{|x-m|\leq W, |y-n|\leq W} \left\| I'_{\Phi_1}(m,n) - I_{\Phi_1}(x,y) \right\|_1 + \lambda \cdot \text{Rate}\,(\text{MV} - \text{PMV}) \tag{2}$$

where W defines the size of search space of candidate sub-blocks in reference frame I', and Rate $(.)$ is the rate term for the motion vector prediction error. As done in H.264, either the l_1-norm (*sum of absolute difference* (SAD)) shown in (2), or the l_2-norm (*sum of square difference* (SSD)), can be used as the pixel difference metric.

3.3. Sub-block Motion Compensation

Having found a best-matched predictor sub-block $I'_{\Phi_s}(m,n)$ for each target sub-block $I_{\Phi_s}(x,y)$, we next perform motion compensation (MC) to compute the prediction residuals in the complete 16×16 MB for residual coding. We discuss two methods for MC in order: non-overlapping MC, and overlapping MC. z-mode offers both options, and we will encode one bit to indicate to the decoder which type of MC is used for this particular MB.

3.3.1. Non-overlapping Motion Compensation

Non-overlapping MC is the conventional MC used in H.26x, where for each target sub-block $I_{\Phi_s}(x,y)$, the pixel difference from the best-matched predictor sub-block, $I'_{\Phi_s}(m,n) - I_{\Phi_s}(x,y)$, is computed for residual coding. Transform coding of the residuals using Discrete Cosine Transform (DCT) is then performed, as typically done in H.264.

3.3.2. Overlapping Motion Compensation

For overlapping motion compensation, we extend the block $I'_{\Phi_s}(x,y)$ in the reference frame by one pixel across the depth edge boundary. Specifically, sub-block supports Φ_1 and Φ_2 are both expanded to include pixels on either side of the horizon and vertical boundaries, B_H and B_V. Formally, we define B_H and B_V as follows:

$$\begin{aligned} B_H &= \{(i,j) \in \Phi | Z(x+i,y+j) < \bar{z}_\Phi(x,y) \\ &\quad | \; Z(x+i,y+j\pm 1) \geq \bar{z}_\Phi(x,y)\} \end{aligned} \tag{3}$$

$$\begin{aligned} B_V &= \{(i,j) \in \Phi | Z(x+i,y+j) < \bar{z}_\Phi(x,y) \\ &\quad | \; Z(x+i\pm 1,y+j) \geq \bar{z}_\Phi(x,y)\} \end{aligned} \tag{4}$$

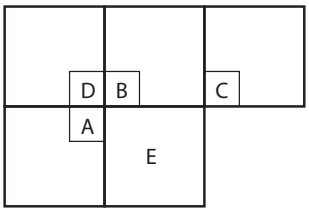


Fig. 4. Motion vector prediction for E is the median value of A, B and C or D depending on position.

The extended supports $\Phi_1^* = \Phi_1 \cup B_H \cup B_V$ and $\Phi_2^* = \Phi_2 \cup B_H \cup B_V$ will lead to overlapping sub-blocks $I'_{\Phi_1^*}(m,n)$ and $I'_{\Phi_2^*}(k,l)$ in $I_\Phi(x,y)$ after motion prediction, as shown in Fig. 3. We must hence perform *alpha blending* in the overlapped region.

To determine how alpha blending should be performed, we first perform 2D filtering around the depth edge using the following filter to determine the weights for pixel mixing:

$$A = \begin{bmatrix} 0 & 1/5 & 0 \\ 1/5 & 1/5 & 1/5 \\ 0 & 1/5 & 0 \end{bmatrix} \tag{5}$$

Having derived the weights, each pixel in the overlapped region will be the weighted sum of the two pixels in each sub-block.

4. MOTION VECTOR PREDICTION

Bits required to encode the MV $v_{\Phi_s}(x,y)$ for a sub-block $I_{\Phi_s}(x,y)$ can be reduced if a good PMV, $u_{\Phi_s}(x,y)$, can be estimated from MVs $v_\Phi(m,n)$'s of neighboring 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 $v_\Phi(m,n)$ is encoded and transmitted. In H.264, PMV is computed to be the median of the MVs of the neighboring blocks. In Fig. 4, PMV for block E is computed using MVs of block A (left), block B (top) and block C (top-right). If top-right block B is not available, then top-left block D is used instead.

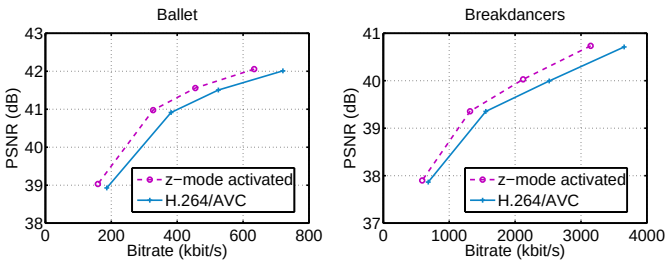
4.1. Depth-based Predicted Motion Vector

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 texture map given depth information, however, we have available depth values to evaluate the "trustworthiness" of motion information provided by neighboring blocks. In other words, assuming 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 texture sub-block $I_{\Phi_s}(x,y)$ as a weighted sum of MVs, $v_\Phi(m,n)$'s of neighboring blocks, $(m,n) \in \mathcal{N}(x,y)$, where the weights w's are proportional to the depth similarity between the neighboring blocks and the target block. Mathematically, we write:

$$u_{\Phi_s}(x,y) = \frac{1}{\bar{w}} \sum_{(m,n)\in\mathcal{N}(x,y)} w(\bar{z}_{\Phi_s}(x,y) - \bar{z}_\Phi(m,n)) \; v_\Phi(m,n) \tag{6}$$

where $\bar{z}_\Phi(m,n)$ is the mean of a neighboring depth block $Z_\Phi(m,n)$, and $\bar{w} = \sum_{(m,n)\in\mathcal{N}(x,y)} w(\bar{z}_{\Phi_s}(x,y) - \bar{z}_\Phi(m,n))$ is a scaling factor so that sum of weights w's divided by $\bar{w}$ is 1. The real-valued



(a) `Ballet` (b) `Breakdancers`

Fig. 5. Objective PSNR results of texture coding.

precision of the resulting PMV $u_{\Phi_s}(x,y)$ is then rounded to half-pel or quarter-pel precision to be H.264 compliant.

4.2. Finding Optimal Weights

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

$$w(x) = \frac{1}{2b} e^{-\frac{|x|}{b}} \tag{7}$$

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

$$b^* = \min_{b>0} \sum_{I_{\Phi_s}(x,y)\in\Theta} |u_{\Phi_s}(x,y) - v_{\Phi_s}(x,y)| \tag{8}$$

5. EXPERIMENTATION

The performance of the proposed framework is evaluated using the Multiview Video-plus-Depth (MVD) 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 [10], which gives a reasonable alignment between texture and depth edges. In the experiments, the original depth video has been utilized to partition each texture MB as described in Section 3.1. The objective compression performance of the proposed method is plotted as rate-distortion (RD) curves in Fig. 5: bitrate (kbit/s) averaged over 100 frames versus objective quality, measured in peak signal-to-noise ratio (PSNR). The RD results correspond respectively to four QP quantization parameters: 23, 25, 27 and 32.

We implemented the proposed z-mode in JM 18.0, and deactivated intra-modes in P-slices to provide a comparison among inter-modes only. The JM has been set up with the main profile and a full ME search. In motion estimation, only luminance component has been taking into account. The Laplacian parameter b that minimizes the motion vector prediction as defined in Eq. (6) has been set to 0.05. The activation of the z-mode in H.264 clearly indicates the benefit of arbitrarily shaped sub-block motion prediction, where a coding gain of 0.75 dB and 0.64 dB is achieved for the video-plus-depth sequence `Ballet` and `Breakdancers`, respectively. z-mode is selected, relative to other modes, at around 17% and 9% at high and low bitrate, respectively. In addition, the experiments results have shown that 75% of the selected z-mode used the overlapping MC mode.

6. CONCLUSION

Texture-plus-depth, where texture and depth maps from the same camera viewpoint are coded together, is an important video format for 3D and multiview video communication. Given the inherent correlation between texture and depth maps from the same viewpoint, in this paper we propose to use detected edges in a depth map to divide the corresponding block in texture map into sub-blocks for separate motion compensation. The key observation is that pixels of similar depth have similar motion, thus representing two sub-blocks each of similar depth (e.g., foreground and background) can lead to smaller prediction residual and coding gain. In addition, we introduce the notion of non-overlapping and overlapping sub-block in motion compensation to deal with possible texture-depth boundary misalignments. Experimental results show up to 0.7dB gain in PSNR over native H.264 implementation. Several issues remain that warrant further research, such as future studies on the depth coding impact on the overall framework performance.

Acknowledgment

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

7. REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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*, January 2009, vol. 2009 (2009).

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski, "High-quality video view interpolation using a layered representation," *Proc. of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 23, no. 3, pp. 600–608, Aug. 2004.