# Depth-based Block Partitioning for 3D Video Coding

Fabian Jäger
Institut für Nachrichtechnik
RWTH Aachen University, GERMANY
jaeger@ient.rwth-aachen.de

***Abstract*—3D video is an emerging technology that bundles depth information with texture videos to allow for view synthesis applications at the receiver. Depth discontinuities define object boundaries in both, depth maps and the collocated texture video. Therefore, segmentation of the depth map can be used for more accurate partitioning of the corresponding texture component.**

**In this paper, already coded depth maps are used to increase coding efficiency for texture videos by deriving an arbitrarily shaped partitioning of a texture block based on the collocated depth block. By applying motion compensation to each partition and combining them afterwards by a depth-based segmentation mask, highly accurate prediction signals can be formed that reduce the remaining residual signal significantly. Simulation results show bitrate savings of up to $5.7\%$ for the dependent texture views and up to $2.3\%$ with respect to the total bitrate.**

***Index Terms*—3D video coding, depth map, motion partition, High Efficiency Video Coding (HEVC)**

## I. Introduction

3D video technology extends conventional stereo videos by the addition of corresponding depth maps. Nowadays, stereoscopic displays require special glasses to provide a depth illusion to the views when displaying stereoscopic content. Emerging technologies, such as auto-stereoscopic and depth-adapting stereoscopic displays, require more than two views of the same scene to allow for their enhanced 3D capabilities. The Multi-View plus Depth (MVD) data format bundles multi view texture videos with their corresponding depth maps and is designed for the described display technologies.

Consequently, combined coding of texture and depth videos becomes an important research field with the goal to exploit inter-component dependencies to increase overall coding performance. Both directions (texture-to-depth and depth-to-texture) show advantages and disadvantages, but using depth information to reduce the texture bitrate seems more appropriate as the texture bitrate is significantly higher than the bitrate for the depth component. Several approaches to improve overall coding efficiency by utilizing inter-component dependencies were recently investigated.

In [1] coded texture information of the same view is used to generate a segmentation mask, which is used to predict the collocated depth block. For each of the two segments of the resulting binary segmentation mask a DC prediction value is derived. This shape prediction from texture to depth can improve the prediction quality and especially the location accuracy of depth discontinuities.

While the aforementioned method is designed for intra-predicted blocks in depth map videos, motion information shows high correlation between texture and depth components, too. Reusing already coded motion information of the texture view to reduce the required bitrate of the same view's depth component is proposed in [2]. In that approach, the motion vector information and also the partitioning of the prediction units can be inherited from the collocated texture block when coding a depth block. A single flag signals the usage of the *Motion Parameter Inheritance*.

The inheritance of coded information from texture to depth can be taken even further. In [3] the authors propose to limit the depth of the coding quad-tree for the depth component to the corresponding texture quad-tree. On the one hand, this limitation makes it possible to save some bitrate for the splitting flag in the depth component, but on the other hand it introduces a parsing dependency between the two components. Consequently, it is impossible to decode the depth component without having decoded the texture frame before.

All three described approaches utilize the texture video to improve coding efficiency of the depth map. Therefore, the coding order needs to be texture-first coding for all views. Moreover, the resulting bitrate reduction for the depth component ends up in a very minor improvement of coding efficiency with respect to the overall bitrate of the 3D video bitstream. This is due to the fact that the depth bitrate is relatively low (approx. $7\%$) compared to the overall bitrate including texture and depth. Thus, utilizing depth information to efficiently code the texture component seems to be the more appropriate choice.

Synthesizing an additional prediction signal for the dependent texture views based on coded depth information is proposed by [4] and [5]. In the latter method, the resulting residual is always skipped and no residual is coded. This approach is based on the assumption that regions, which can be synthesized completely, do not need to be coded again. View synthesis prediction (VSP) is an efficient way of reducing the required bitrate for the dependent texture views, but at the same time it introduces additional complexity to the decoder as it requires a very fine-grained disparity compensation to be carried out. In an extreme case, VSP can be seen as a pixel-wise disparity compensation including random access to the decoded picture buffer.

In the proposed method, the dependent views' depth infor-

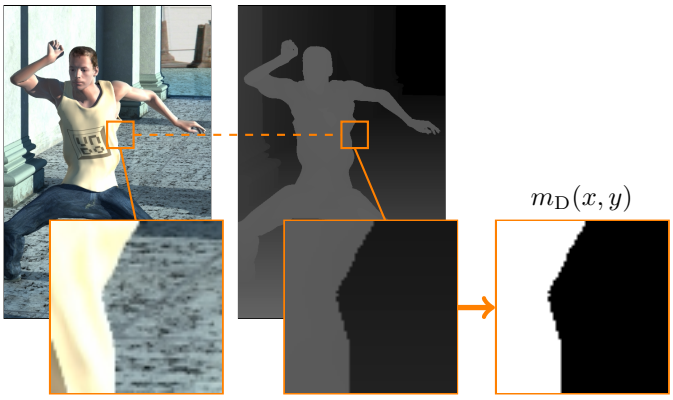


Fig. 1. Cropped frame from *Undo_Dancer* test sequence with magnified component blocks of an example coding unit. The collocated depth block segments the block into foreground and background, which is defined by a binary mask $m_{\mathrm{D}}(x,y)$.

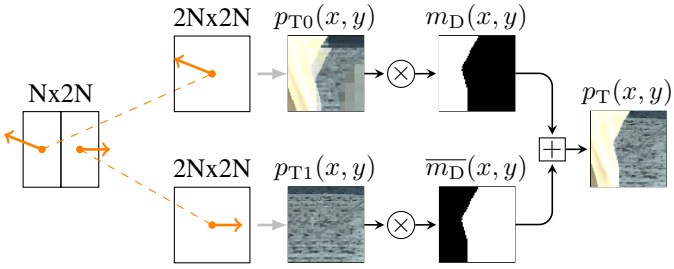


Fig. 2. DBBP merging process: For each of the two decoded motion parameters a $2N \times 2N$ motion compensation is performed. The resulting prediction signals $p_{\mathrm{T0}}(x,y)$ and $p_{\mathrm{T1}}(x,y)$ are combined using the DBBP mask $m_{\mathrm{D}}(x,y)$.

mation is coded before the texture component to be able to utilize this coded depth data. Based on a binary segmentation mask from the reconstructed depth map an arbitrarily shaped block partitioning for the collocated texture block is derived. Each of the two partitions are motion compensated and afterwards re-combined based on the segmentation mask.

The remainder of this paper is structured as follows: Section II describes the overall concept of *Depth-based Block Partitioning* (DBBP) before Section III explains the actual coding scheme. Experimental results based on the proposed algorithm are presented in Section IV. Finally, Section V summarizes the results of the proposed coding method and gives an outlook on potential follow-up activities.

## II. Block Partitioning Algorithm

The proposed algorithm for using depth information for block partitioning in texture views consists of three sequential steps, which will be described in more detail in the following section.

### A. Depth Segmentation

In an initial step the collocated depth block of the current coded tree block (CTB) of the texture component is segmented into two arbitrarily shaped segments. The segmentation is performed based on a very simple thresholding mechanism where the threshold $\bar{d}$ is computed from the mean depth value.

$$\bar{d} = \frac{1}{(2N)^2} \sum_{x=0}^{2N-1} \sum_{y=0}^{2N-1} d(x,y) \tag{1}$$

Here, $2N$ defines the width/height of the current texture block and $d(x,y)$ resembles the already coded, corresponding depth map of the current texture frame.

Afterwards, a binary segmentation mask $m_{\mathrm{D}}(x,y)$ is generated based on $\bar{d}$ as follows.

$$m_{\mathrm{D}}(x,y) = \begin{cases} 1, & \text{if } d(x,y) \geq \bar{d}, \\ 0, & \text{otherwise.} \end{cases}, x,y \in [0, 2N-1] \tag{2}$$

The resulting binary segmentation mask defines the shape of the partitioning of the texture CTB. While motion or disparity compensation in a modern video coder (e.g. in HEVC) is performed on a block-basis, an arbitrarily shaped block partitioning typically requires pixel-based compensation. This concept is applied in view-synthesis prediction (VSP), which warps each pixel position based on the corresponding depth value to the position of the particular reference frame. By this fine-grained compensation approach higher order deformations can be approximated. To achieve this amount of precision for the prediction stage of a video coder, VSP introduces very high complexity compared to conventional block-based motion/disparity compensation. This is mainly due to the irregular access to the reference buffer and the pixel-wise conversion between depth and disparity. To overcome this issue of VSP, the proposed depth-based block partitioning (DBBP) scheme still uses block-based compensation in the prediction stage, as it is described in the following subsection.

### B. Block-based Compensation

In the proposed DBBP scheme, the actual motion or disparity compensation is performed on a $2N \times 2N$ partitioning, which means that the full CTB is shifted by the coded vector information. This full-size motion/disparity compensation is performed twice, once for each segment, and results in two prediction signals $p_{\mathrm{T0}}(x,y)$ and $p_{\mathrm{T1}}(x,y)$. Consequently, two sets of vector information need to be coded for a DBBP block. The assumption behind this approach is that a texture block is typically segmented into foreground and background based on the collocated depth block. These two depth layers can then be compensated independently by their own sets of motion or disparity vectors.

### C. Merging of Prediction Signals

After having generated two full-size prediction signals $p_{\mathrm{T0}}(x,y)$ and $p_{\mathrm{T1}}(x,y)$ for a DBBP-coded block, the segmentation mask $m_{\mathrm{D}}(x,y)$ is used to merge these into the final prediction signal $p_{\mathrm{T}}(x,y)$ for the current texture CTB.

$$p_{\mathrm{T}}(x,y) = \begin{cases} p_{\mathrm{T0}}(x,y), & \text{if } m_{\mathrm{D}}(x,y) = 1, \\ p_{\mathrm{T1}}(x,y), & \text{otherwise.} \end{cases}, x,y \in [0, 2N-1] \tag{3}$$

By merging the two prediction signals, shape information from the depth map allows to independently compensate foreground and background objects in the same texture CTB. At the same time, DBBP does not require pixel-wise motion/disparity compensation. Memory access to the reference buffers is always regular (block-based) for DBBP-coded blocks in contrast to approaches like VSP. Moreover, DBBP always uses full-size blocks for compensation. This is preferable with respect to complexity, because of the higher probability of finding the data in the memory cache.

## III. Coding of Partitioning Information

As described in the previous section, DBBP requires to code two sets of motion information, one for each segment. A modern video coder, such as HEVC, allows to use rectangular, non-square partitioning modes within a coding tree unit (CTU) for finer-grained motion compensation. For each of the partitions in a CTU a separate set of motion information is coded. This coding scheme is reused in the proposed depth-based block partitioning.

After the encoder has derived the optimal motion/disparity information for each DBBP segment, this information is mapped into one of the available rectangular, non-square partitioning modes of HEVC. This includes asymmetric motion partitioning modes [6], which were introduced for HEVC. The mapping of the binary segmentation mask to one of the 6 available two-segment partitioning modes is performed by a correlation analysis. For each of the available partitioning modes ($i \in [0,5]$) 2 binary masks $m_{2i}(x,y)$ and $m_{2i+1}(x,y)$ are generated, where $m_{2i+1}(x,y)$ is the negation of $m_{2i}(x,y)$. To find the best matching partitioning mode $i_{\text{opt}}$ for the current depth-based segmentation mask $m_{\text{D}}(x,y)$, the following algorithm is performed:

$$
\begin{aligned}
k_{\text{opt}} &= \arg\max_{k} \sum_{x=0}^{2N-1} \sum_{y=0}^{2N-1} m_{\text{D}}(x,y) \cdot m_k(x,y) \quad , k \in [0,11] \\
i_{\text{opt}} &= \left\lfloor \frac{k_{\text{opt}}}{2} \right\rfloor \\
b_{\text{inv}} &= \begin{cases} 1, & \text{if } k_{\text{opt}} \text{ is odd,} \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
\tag{4}
$$

The boolean variable $b_{\text{inv}}$ defines whether the derived segmentation mask $m_{\text{D}}(x,y)$ needs to be inverted or not. This might be necessary in some cases where the indexing of the conventional partitioning schemes is complementary to the indexing in the segmentation mask. In the conventional partitioning modes, index 0 always defines the partition in the top-left corner of the current block, while the same index in the segmentation mask defines the segment with the lower depth values (background objects). To align the positioning of the corresponding sets of motion information between $m_{\text{D}}(x,y)$ and $i_{\text{opt}}$, the indexing in $m_{\text{D}}(x,y)$ is inverted, if $b_{\text{inv}}$ is set.

After having found the best matching conventional partitioning mode, motion information is stored and coded according to this optimal mode $i_{\text{opt}}$. Succeeding coding units (CUs) can access the already coded motion information conventionally when deriving motion vector candidates for advanced motion vector prediction (AMVP) or motion vector merging.

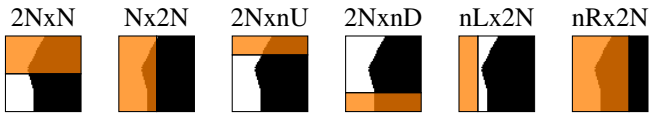


Fig. 3. Superposition of conventional partitioning modes (orange) and depth-based block partitioning (black/white). The best matching partitioning ($i_{\text{opt}}$) mode is selected for storing motion information.

A single flag is added to the coding syntax to signal to the decoder that a block uses DBBP for prediction. An obvious choice would be to send this flag for all of the conventional partitioning modes. But this approach would result in unwanted coding overhead for blocks that do not use DBBP. Therefore, the partitioning mode for DBBP-coded blocks is always set to $N \times 2N$ before coding the partitioning mode. Afterwards, only for $N \times 2N$ partitioned blocks, the DBBP flag is coded in the bitstream. For all other partitioning modes, the DBBP flag is not coded. At the decoder side, the true partitioning is derived after having decoded the DBBP flag. All further processing steps at the decoder remain the same as in the HEVC base specification. This modified part of the decoding process is described in Algorithm 1.
By this modified coding scheme the number of DBBP flags for CUs not using DBBP can be reduced significantly.

**Algorithm 1** Decoding Process for Partitioning Mode in Dependent Texture Views

**Precondition:** Prediction mode of CU is not INTRA.

```
function DECODEPARTMODE(CU)
    part_mode ← read(bitstream)
    dbbp_flag ← false
    if part_mode == SIZE_2NxN then
        dbbp_flag ← read(bitstream)
    if dbbp_flag == true then
        part_mode ← derivePartModeFromDepth(CU)
    return part_mode
```

## IV. Experimental Results

The proposed depth-based block partitioning algorithm was implemented into the JCT-3V HEVC Test Model (HTM 6.1) [7]. An encoder configuration parameter enables or disables the usage of DBBP. The JCT-3V common test conditions [8] only define a texture-first coding order, whereas the proposed method requires depth-first coding for the dependent views. Therefore, the presented simulation results are performed according to the CTC with a depth-first coding order for all dependent views. In this configuration, some coding tools, which are implemented in HTM 6.1, are automatically disabled for the dependent views as they require the availability

of coded texture information when coding depth maps. The disabled tools are texture-dependent depth modeling modes (DMM3 and DMM4) [1], motion vector inheritance (MPI) [2] and depth quad tree limitation (QTLPC) [3].

TABLE I
BD-RATE SAVINGS WHEN USING DEPTH-BASED BLOCK PARTITIONING.

| Sequence | Texture View 1 | Texture View 2 | Total Texture |
|---|---|---|---|
| Balloons | -2.40 % | -2.42 % | -1.07 % |
| Kendo | -4.99 % | -5.71 % | -2.28 % |
| Newspaper_CC | -1.64 % | -2.15 % | -0.79 % |
| GT_Fly | -0.79 % | -1.13 % | -0.24 % |
| Poznan_Hall2 | -0.56 % | -2.01 % | -0.67 % |
| Poznan_Street | -1.23 % | -1.47 % | -0.45 % |
| Undo_Dancer | -3.07 % | -3.98 % | -0.93 % |
| **1024x768** | **-3.01 %** | **-3.43 %** | **-1.38 %** |
| **1920x1088** | **-1.41 %** | **-2.15 %** | **-0.57 %** |
| **AVERAGE** | **-2.10 %** | **-2.70 %** | **-0.92 %** |

The simulation results in Table I clearly show the benefits of depth-based block partitioning with respect to coding efficiency. While the bitrate reduction for the dependent views gets up to $5.71\%$, the resulting bitrate reduction w.r.t. the total bitrate finds its maximum at about $2.28\%$. This effect can be explained by the fact that the bitrate of the base view remains unchanged when using DBBP as it is only applied to the dependent views. By this restriction the base view is still HEVC-compatible. Moreover, it can be seen that the bitrate reduction varies between the different test sequences, which is mainly due to the varying quality of reconstructed depth maps. Whenever the segmentation mask from the reconstructed depth map is aligned with object boundaries in the texture component, DBBP yields a high prediction quality for a CU. Consequently, for sequences with less structural information or weaker depth discontinuities, DBBP cannot provide better prediction quality than conventional partitioning modes.

When it comes to complexity, the proposed depth-based block partitioning does not require additional computational resources at the decoder compared to the current reference implementation. This reference decoder uses relatively complex tools like view-synthesis prediction (VSP). DBBP is mostly used in regions where the reference utilizes VSP and consequently the overall complexity can even be reduced by enabling DBBP. Moreover, the number of very small partitions in $8 \times 8$ CUs is reduced, because DBBP is often applied to blocks of $32 \times 32$ or $16 \times 16$ instead of further splitting them for motion or disparity compensation. These bigger block sizes reduce random access to the picture buffer memory and increase data locality, which is beneficial for software and hardware implementations.

To further reduce the resulting complexity at the decoder, DBBP can be restricted to bigger block sizes by defining a minimum block size. Simulation results show that restricting DBBP to a minimum block size of $16 \times 16$ does not impact the coding efficiency.

## V. SUMMARY

This paper proposes a novel partitioning scheme for texture videos in a 3D video coding scenario. With depth-based block partitioning (DBBP) already coded depth information is used to derive a segmentation mask, which thereafter allows for for fine-grained motion compensation of foreground and background independently. The segmentation mask is eventually utilized to merge the two compensated blocks to the final prediction signal. With DBBP, information about the location of depth discontinuities is utilized to improve motion and disparity compensation quality.

The proposed algorithm can reduce the required bitrate of the two dependent views by approximately $2.1\%$ and $2.7\%$ on average. For some sequences the bitrate reduction for the dependent views is even at up to $5.0\%$ and $5.7\%$.

Further research is needed to investigate how joint encoder optimization of texture and depth components may improve the bitrate reduction. In the current encoder implementation, the depth component is optimized independently and therefore some edge information might get lost during optimization. This depth discontinuity information might be used by tools like DBBP to increase coding efficiency for the texture component.

## REFERENCES


[1] P. Merkle, C. Bartnik, K. Müller, D. Marpe, and T. Wiegand, "3D video: Depth coding based on inter-component prediction of block partitions," in *Proceedings of IEEE Picture Coding Symposium*, Kraków, Poland, May 2012, pp. 149–152.

[2] M. Winken, H. Schwarz, and T. Wiegand, "Motion vector inheritance for high efficiency 3D video plus depth coding," in *Proceedings of IEEE Picture Coding Symposium*, Kraków, Poland, 2012, pp. 53–56.

[3] Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T VCEG and ISO/IEC MPEG, "3D-CE3.h: Depth quadtree prediction for 3DHTM 4.1," JCT3V-B0068, Tech. Rep., October 2012.

[4] C. Lee and Y.-S. Ho, "A framework of 3D video coding using view synthesis prediction," in *Proceedings of IEEE Picture Coding Symposium*, Kraków, Poland, 2012, pp. 9–12.

[5] F. Jäger and C. Feldmann, "Warped-skip mode for 3D video coding," in *Proceedings of IEEE Picture Coding Symposium*, Kraków, Poland, 2012, pp. 145–148.

[6] I.-K. Kim, S. Lee, M.-S. Cheon, T. Lee, and J. Park, "Coding efficiency improvement of HEVC using asymmetric motion partitioning," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2012 IEEE International Symposium on*, Seoul, June 2012, pp. 1–4.

[7] Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T VCEG and ISO/IEC MPEG, "3D-HEVC test model 4," Doc. JCT3V-D1005, Incheon, Republic of Korea, Tech. Rep., April 2013.

[8] ——, "Common test conditions of 3DV core experiments," JCT3V-D1100, Incheon, Republic of Korea, Tech. Rep., April 2013.