|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  8th Meeting: Valencia, ES, 29 March – 4 April 2014 | Document: JCT3V-H0057 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Sub-PU Restriction for DBBP** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Fabian Jäger Institut für Nachrichtentechnik RWTH Aachen University | Tel: Email: | +49 (0) 241 80 27678 [jaeger@ient.rwth-aachen.de](mailto:jaeger@ient.rwth-aachen.de) |
| *Source:* | RWTH Aachen University | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

At the 6th JCT-3V meeting, a motion/disparity prediction method was introduced in JCT3V-G0106 [1] that uses a depth-derived binary segmentation mask for the derivation of PU partitioning and for merging of two prediction signals. The method is called Depth-based Block Partitioning (DBBP). The segmentation process in JCT3V-G0106 results in two segments, which motion information is coded as a conventional bi-partitioned coding unit with two sets of motion information. In the original proposal, Sub-PU Inter-View Motion Prediction (SPIVMP), as proposed in JCT3V-F0101 [2], is disabled for coding units using DBBP, as sub-PU motion information interferes with the sample-precise segmentation mask design. At the 7th JCT-3V meeting it was proposed in JCT3V-G0138 [3] to use sub-PU storage of disparity information for block using View Synthesis Prediction (VSP). Consequently, there is currently an incompatibility between VSP and DBBP, which is resolved in this proposal by introducing a restriction on DBBP-coded blocks to not use VSP for disparity compensation. The impact on the coding efficiency is negligible.

# Depth-based 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.

## 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. As the depth component is coded after the corresponding texture view in the current CTC, a virtual depth map is derived from the base view’s reconstructed depth and shifted by a disparity vector, which is itself derived from the neighboring blocks (by means of DoNBDV).

The segmentation of the (virtual) depth map is performed based on a very simple thresholding mechanism where the threshold is computed from the mean depth value.

Here, defines the width/height of the current texture block and resembles the already coded, corresponding depth map of the reference view’s texture frame.

Afterwards, a binary segmentation mask is generated based on as follows.

The resulting binary segmentation mask defines the shape of the partitioning of the texture block. 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 pixel-based 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, pixel-based 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.

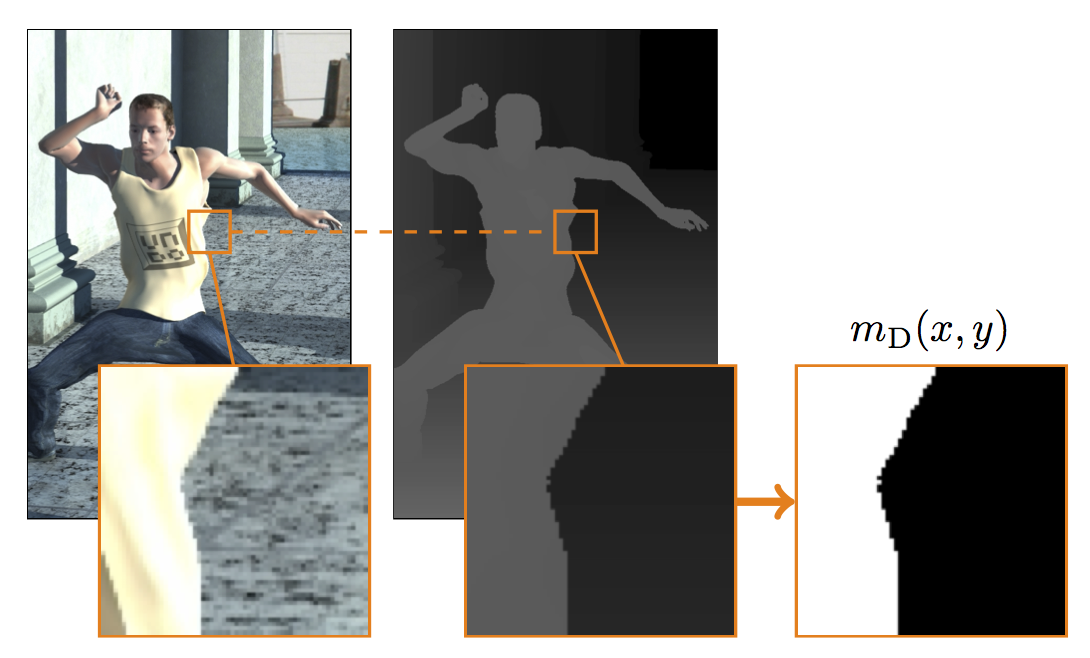


Figure 1: Cropped frame from *Undo\_Dancer* test sequence with magnified component blocks of an example coding unit. The collocated depth block segments the block into foreground and background, which is defined by a binary mask .

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.

## Block-based Compensation

In the proposed DBBP scheme, the actual motion or disparity compensation is performed on a 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 and .

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.

## Merging of Prediction Signals

After having generated two full-size prediction signals and for a DBBP-coded block, the segmentation mask is used to merge these into the final prediction signal for the current texture CTB.

By merging the two prediction signals, shape information from the depth map allows to independently compensate foreground and background objects in the same texture block. 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.

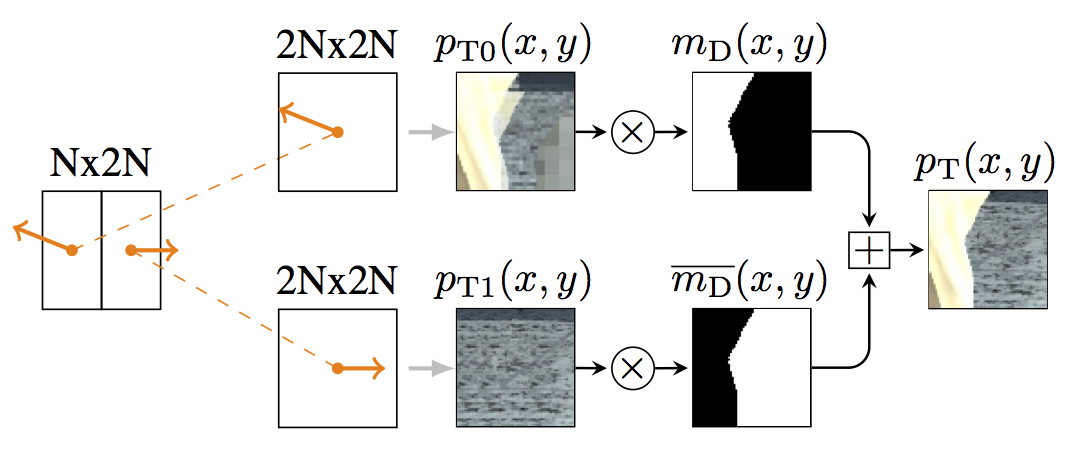
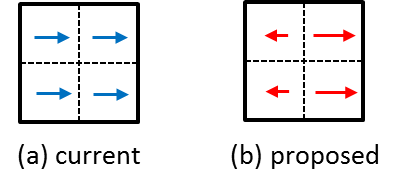


Figure 2: DBBP merging process: For each of the two decoded motion parameters a motion compensation is performed. The resulting prediction signals and are combined using the DBBP mask .

# Sub-PU Storage of Disparity Vectors in VSP

In JCT3V-G0148, it was proposed to store the disparity parameters (the red vectors in **Figure 3**), which are actually utilized for prediction, in the motion storage instead of those disparity parameters (the light blue vector in **Figure 3**), which are utilized to fetch the virtual depth map as shown in Figure 3. The stored disparity parameters are subjected to the other processes including de-blocking filter, AMVP, TMVP, NBDV derivation, and MPI as specified in the current WD.



**Figure 3: Disparity parameters stored in motion memory**

Since the view synthesis prediction process is performed on sub-PU level as the sub-PU inter-view motion prediction, it was also proposed to merge the processes of view synthesis prediction and of sub-PU inter-view motion prediction.

# Simulation Results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video PSNR / video bitrate | video PSNR / total bitrate | synth PSNR / total bitrate | enc time | dec time |
| Balloons | 0,0% | -0,1% | 0,0% | 0,0% | 0,0% | -0,1% | 99,2% | 99,3% |
| Kendo | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 102,7% | 101,2% |
| Newspaper\_CC | 0,0% | -0,1% | 0,0% | 0,0% | 0,0% | 0,0% | 100,5% | 102,8% |
| GT\_Fly | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 103,0% | 98,1% |
| Poznan\_Hall2 | 0,0% | 0,1% | -0,2% | 0,0% | 0,0% | 0,0% | 99,3% | 99,9% |
| Poznan\_Street | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 98,6% | 100,1% |
| Undo\_Dancer | 0,0% | 0,0% | 0,1% | 0,0% | 0,0% | 0,0% | 100,2% | 102,0% |
| Shark | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 104,2% | 104,3% |
| 1024x768 | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 100,8% | 101,1% |
| 1920x1088 | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 101,1% | 100,9% |
| **average** | **0,0%** | **0,0%** | **0,0%** | **0,0%** | **0,0%** | **0,0%** | **101,0%** | **101,0%** |

# Cross Check

The cross check of the proposed restriction of DBBP to disallow usage of VSP was performed by NTT Docomo. They investigated the required source code modifications and ran the simulations for verification of the presented results.

In their investigation they did not find any problems with the source code. Their simulation results perfectly match with those presented in this document.

# Specification Text Changes

I.8.5.3.2.1 Derivation process for luma motion vectors for merge mode

This process is only invoked when MergeFlag[ xPb ][ yPb ] is equal to 1, where ( xPb, yPb ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

* the luma motion vectors mvL0 and mvL1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0 and predFlagL1,
* the flag ivpMvFlag, specifying, whether the current PU is coded using inter-view motion prediction,
* the flag vspModeFlag, specifying, whether the current PU is coded using view synthesis prediction,
* the flag subPbMotionFlag, specifying, whether the motion data of the current PU has sub prediction block size motion accuracy,
* the flag dispDerivedDepthFlag, specifying, whether the current PU uses disparity derived depth,
* the variable dispDerivedDepthVal (when dispDerivedDepthFlag is equal to 1).

The function differentMotion( N, M ) is specified as follows:

* If one of the following conditions is true, differentMotion( N, M ) is equal to 1:
  + predFlagLXN != predFlagLXM (with X being replaced by 0 and 1),
  + mvLXN != mvLXM (with X being replaced by 0 and 1),
  + refIdxLXN != refIdxLXM (with X being replaced by 0 and 1),
* Otherwise, differentMotion( N, M ) is equal to 0.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

* 1. The derivation process for the base merge candidate list as specified in subclause 0 is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being a modified luma location ( xPb, yPb ), the modified variables nPbW and nPbH, the modified variable partIdx, the luma location ( xOrigP, yOrigP ), the variables nOrigPbW and nOrigPbH, the merge candidate list baseMergeCandList, the luma motion vectors mvL0N and mvL1N, the reference indices refIdxL0N and refIdxL1N, and the prediction list utilization flags predFlagL0N and predFlagL1N, with N being replaced by all elements of baseMergeCandList.
  2. For N being replaced by A1, B1, B0, A0 and B2, the following applies:
     + If N is an element in baseMergeCandList, availableFlagN is set equal to 1.
     + Otherwise (N is not an element in baseMergeCandList), availableFlagN is set equal to 0.
  3. Depending on iv\_mv\_pred\_flag[ nuh\_layer\_id ] and DispAvailabilityIdc[ xPb ][ yPb ], the following applies:
     + If iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0 or DispAvailabilityIdc[ xPb ][ yPb ] is not equal to DISP\_NONE, the flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC are set equal to 0.
     + Otherwise (iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause I.8.5.3.2.10 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH as inputs, and the output is assigned to the availability flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC, the reference indices refIdxLXIvMC, refIdxLXIvMCShift and refIdxLXIvDC, the prediction list utilization flags predFlagLXIvMC, predFlagLXIvMCShift and predFlagLXIvDC, and the motion vectors mvLXIvMC, mvLXIvMCShift and mvLXIvDC (with X being 0 or 1, respectively).
  4. Depending on view\_synthesis\_pred\_flag[ nuh\_layer\_id ], dbbp\_flag[ xPb ][ yPb ] and DispAvailabilityIdc[ xPb ][ yPb ], the following applies:
     + If view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 0 or DispAvailabilityIdc[ xPb ][ yPb ] is equal to DISP\_NONE or dbbp\_flag[ xPb ][ yPb ] is equal to 1, the flag availableFlagVSP is set equal to 0.
     + Otherwise (view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause I.8.5.3.2.13 is invoked with the luma locations ( xPb, yPb ) and the variables nPbW and nPbH as inputs, and the outputs are the availability flag availableFlagVSP, the reference indices refIdxL0VSP and refIdxL1VSP, the prediction list utilization flags predFlagL0VSP and predFlagL1VSP, and the motion vectors mvL0VSP and mvL1VSP.
  5. Depending on mpi\_flag[ nuh\_layer\_id ], the following applies:
     + If mpi\_flag[ nuh\_layer\_id ] is equal to 0, the variables availableFlagT and availableFlagD are set equal to 0.
     + Otherwise (mpi\_flag[ nuh\_layer\_id ] is equal to 1), the following applies:
       - The derivation process for inter layer predicted sub prediction block motion vector candidates as specified in subclause I.8.5.3.2.16 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the variable refViewIdx being equal to −1, and the variable mvDisp being equal to ( 0, 0 ) as inputs, and the outputs are the prediction utilization flag predFlagLXT, the motion vector mvLXT and the reference indices refIdxLXT (with X being 0 or 1, respectively).
       - The flag availableFlagT is set equal to ( predFlagL0T | | predFlagL1T ).
       - The derivation process for the disparity derived merging candidates as specified in subclause I.8.5.3.2.19 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH as inputs, and the outputs are the flag availableFlagD, the prediction utilization flag predFlagLXD, the reference index refIdxLXD, the motion vector mvLXD (with X being 0 or 1, respectively), and the variable dispDerivedDepthVal.
  6. The merging candidate list, extMergeCandList, is constructed as follows:

i = 0  
 if( availableFlagT )  
 extMergeCandList[ i++ ] = T  
 if( availableFlagD )  
 extMergeCandList[ i++ ] = D  
 if( availableFlagIvMC && ( !availableFlagT  | |  differentMotion( T, IvMC ) ) )  
 extMergeCandList[ i++ ] = IvMC  
 N = DepthFlag ? T : IvMC  
 if( availableFlagA1 && ( !availableFlagN  | |  differentMotion( N, A1 ) ) )  
 extMergeCandList[ i++ ] = A1 if( availableFlagB1 && ( !availableFlagN  | |  differentMotion( N, B1 ) ) )  
 extMergeCandList[ i++ ] = B1 if( availableFlagB0 )  
 extMergeCandList[ i++ ] = B0 (I‑109) if( availableFlagIvDC && ( !availableFlagA1  | |  differentMotion( A1, IvDC ) ) &&   
 ( !availableFlagB1  | |  differentMotion( B1, IvDC ) ) && ( i < ( 5 + NumExtraMergeCand ) ) )  
 extMergeCandList[ i++ ] = IvDC  
 if( availableFlagVSP && !ic\_flag && iv\_res\_pred\_weight\_idx = = 0 &&   
 i < ( 5 + NumExtraMergeCand ) )  
 extMergeCandList[ i++ ] = VSP  
 if( availableFlagA0 && i < ( 5 + NumExtraMergeCand ) )  
 extMergeCandList[ i++ ] = A0 if( availableFlagB2 && i < ( 5 + NumExtraMergeCand ) )  
 extMergeCandList[ i++ ] = B2 if( availableFlagIvMCShift && i < ( 5 + NumExtraMergeCand ) &&  
 ( !availableFlagIvMC  | |  differentMotion( IvMC, IvMCShift ) ) )  
 extMergeCandList[ i++ ] = IvMCShift

* 1. The variable availableFlagIvDCShift is set equal to 0, and when availableFlagIvMCShift is equal to 0, DepthFlag is equal to 0, and i is less than ( 5 + NumExtraMergeCand ), the derivation process for the shifted disparity merging candidate as specified in subclause I.8.5.3.2.15 is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, and the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N, of every candidate N being in extMergeCandList, extMergeCandList, and i as inputs, and the outputs are the flag availableFlagIvDCShift, the prediction utilization flags predFlagL0IvDCShift and predFlagL1IvDCShift, the reference indices refIdxL0IvDCShift and refIdxL1IvDCShift, and the motion vectors mvL0IvDCShift and mvL1IvDCShift.
  2. The merging candidate list, extMergeCandList, is constructed as follows:

if( availableFlagIvDCShift )  
 extMergeCandList[ i++ ] = IvDCShift  
 j = 0  
 while( i < MaxNumMergeCand ) {(I‑110)  
 N = baseMergeCandList[ j++ ]  
 if( N != A1 && N != B1 && N != B0 && N != A0 && N != B2 )  
 extMergeCandList[ i++ ] = N  
 }

* 1. The variable N is set equal to extMergeCandList[ MergeIdx[ xOrigP ][ yOrigP ] ].
  2. The derivation process for a view synthesis prediction flag as specified in subclause I.8.5.3.2.17 is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nPbW and nPbH, the merge candidate indicator N as the inputs, and the output is the mergeCandIsVspFlag.
  3. The variable vspModeFlag is derived as specified in the following:
     + 1. vspModeFlag = mergeCandIsVspFlag && !ic\_flag &&   
           ( iv\_res\_pred\_weight\_idx = = 0 ) && availableFlagVSP (I‑111)
  4. The variable subPbMotionFlag is derived as specified in the following:
     + 1. subPbMotionFlag = ( ( ( N = = IvMC ) && ( PartMode = = PART\_2Nx2N ) )  
           | | vspModeFlag ) && !dbbp\_flag (I‑112)
  5. The following assignments are made with X being replaced by 0 or 1:
     + 1. mvLX = subPbMotionFlag ? 0 : mvLXN (I‑113)
       2. refIdxLX = subPbMotionFlag ? −1 : refIdxLXN (I‑114)
       3. predFlagLX = subPbMotionFlag ? 0 : predFlagLXN (I‑115)
  6. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nOrigPbW + nOrigPbH ) is equal to 12, the following applies:
     + 1. refIdxL1 = −1 (I‑116)
       2. predFlagL1 = 0 (I‑117)
  7. The disparity availability flag ivpMvFlag is derived as follows:
     + 1. ivpMvFlag = !DepthFlag && ( ( N  =  =  IvMC ) | | ( N  = =  IvMCShift ) ) (I‑118)
  8. The variable dispDerivedDepthFlag is derived as follows:
     + 1. dispDerivedDepthFlag = ( N  = =  D ) (I‑119)

# Patent rights declaration(s)

**RWTH Aachen University may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

# References

[1] F. Jäger, J. Konieczny, and G. Cordara, “CE3: Results on Depth-based Block Partitioning (DBBP),” 7th Meeting, San Jose, USA, 2014.

[2] A. J, K. Zhang, J. L. Lin, and S. Lei, “3D-CE3: Sub-PU level inter-view motion prediction,” 6th Meeting, Geneva, Switzerland, JCT3V-F0110, Oct. 2013.

[3] S. Shimizu and S. Sugimoto, “CE1: Motion parameters stored for VSP-coded blocks,” 7th JCT-3V Meeting, San José, USA, JCT3V-G0148, Jan. 2014.