|  |  |
| --- | --- |
| **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-H0058 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Low Complex Partitioning Derivation for DBBP** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Fabian Jäger Institut für Nachrichtentechnik RWTH Aachen University  Jacek Konieczny, Giovanni Cordara Huawei Technologies Munich Office, European Research Center | Tel: Email:  Tel: Email: | +49 (0) 241 80 27678 [jaeger@ient.rwth-aachen.de](mailto:jaeger@ient.rwth-aachen.de)  +49 (0) 89 158834 4334 [jacek.konieczny@huawei.com](mailto:jacek.konieczny@huawei.com) |
| *Source:* | RWTH Aachen University, Huawei Technologies | | |

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

# Abstract

At the 7th 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 partitioning derivation process in JCT3V-G0106 is relatively high complex, as it needs to analyze the full block of either depth or mask samples to find the resulting partitioning mode. In this contribution, a low complex approach for the same task is proposed, which requires only up to 3 binary comparisons per coding unit for achieving almost the same results as the original method. The impact on the coding efficiency of DBBP is negligible.

# Original Partitioning Derivation Algorithm

As the Depth-based Block Partitioning coding scheme reuses the HEVC syntax to code its information, its depth-derived segmentation mask needs to be mapped into one of the available rectangular, non-square partitioning modes. This includes asymmetric motion partitioning modes [2], 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 ) 2 binary masks and are generated, where is the negation of . To find the best matching partitioning mode for the current depth-based segmentation mask , the following algorithm is performed:

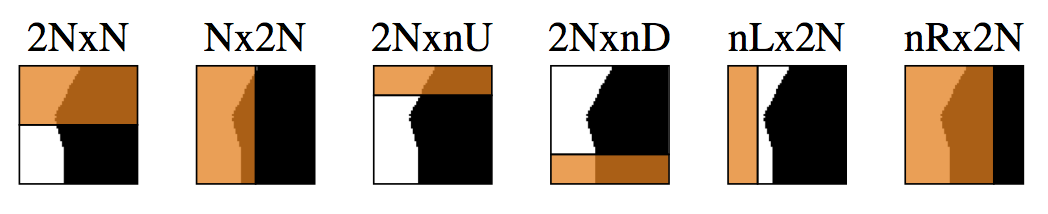


Figure 1: Superposition of conventional partitioning modes (gray) and depth-based block partitioning (black/white). The best matching partitioning () mode is selected for storing motion information.

After having found the best matching conventional partitioning mode, motion information is stored and coded according to this optimal mode . 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.

# Proposed Partitioning Derivation Algorithm

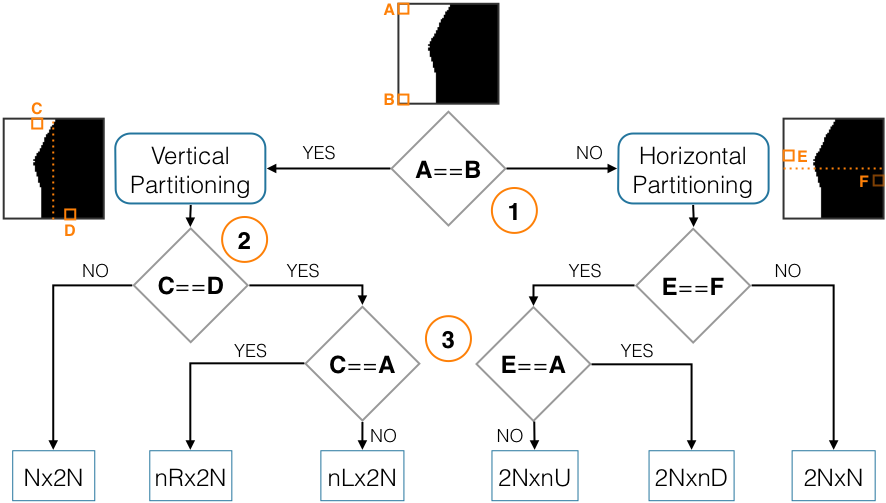


Figure 2: Sample-based derivation of best-matching conventional partitioning mode from depth-based segmentation mask. Only up to 3 comparisons are required for the whole derivation process. In this method, sample positions C, D, E and F are located at quarter block positions.

In the proposed scheme, the partitioning derivation process for DBBP requires only up to 4 sample values of the segmentation mask and up to 3 binary comparison operations.

## Computation of Segmentation Threshold

In the original scheme of DBBP, the mean value of all depth values of the current virtual depth block is used as the threshold for the derivation of the binary segmentation mask. In the proposed method, only the four corner samples are used to compute the mean value used for thresholding in the partitioning derivation process. By this, the complexity is not only reduced in the final partitioning derivation process, but also in the preceding thresholding of the corresponding depth values.

# Complexity Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block Size** | **Original Method** | | **Proposed Method** | |
| **Thresholding** | **Part. Derivation** | **Thresholding** | **Part. Derivation** |
| **8x8** | 3 ADDs 1 SHIFT | 24+18 COMPs 24 ADDs | 3 ADDs 1 SHIFT | 2/3 COMPs |
| **16x16** | 15 ADDs 1 SHIFT | 96+18 COMPs 96 ADDs | 3 ADDs 1 SHIFT | 2/3 COMPs |
| **32x32** | 63 ADDS 1 SHIFT | 384+18 COMPs 384 ADDs | 3 ADDs 1 SHIFT | 2/3 COMPs |
| **64x64** | 255 ADDs 1 SHIFT | 1536+18 COMPs 1536 ADDs | 3 ADDs 1 SHIFT | 2/3 COMPs |

As the table indicates, the number of necessary additions (ADD), binary shifting operations (SHIFT) and binary comparisons (COMP) increases with the block size in the original DBBP scheme. With the proposed partitioning derivation process, the number of computations is independent of the block size and very low compared to the original method. When considering that the DBBP coding scheme is mostly applied on bigger blocks due to its capability of better approximating object boundaries, the effective complexity reduction of the proposed scheme is significant.

# 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% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 98,2% | 96,2% |
| Kendo | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 100,1% | 99,7% |
| Newspaper\_CC | 0,0% | 0,0% | -0,1% | 0,0% | 0,0% | 0,0% | 98,5% | 97,3% |
| GT\_Fly | 0,0% | -0,1% | -0,2% | 0,0% | 0,0% | 0,0% | 99,3% | 97,8% |
| Poznan\_Hall2 | 0,0% | 0,1% | -0,3% | 0,0% | 0,0% | -0,1% | 98,0% | 100,2% |
| Poznan\_Street | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 100,7% | 97,4% |
| Undo\_Dancer | 0,0% | -0,1% | 0,0% | 0,0% | 0,0% | 0,0% | 97,2% | 97,8% |
| Shark | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 99,0% | 100,9% |
| 1024x768 | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 0,0% | 98,9% | 97,7% |
| 1920x1088 | 0,0% | 0,0% | -0,1% | 0,0% | 0,0% | 0,0% | 98,8% | 98,8% |
| **average** | **0,0%** | **0,0%** | **-0,1%** | **0,0%** | **0,0%** | **0,0%** | **98,9%** | **98,4%** |

# Cross Check

The cross check of the proposed complexity reduction method for DBBP was performed by Samsung. 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.8.5.7 Derivation process for a modified partitioning mode**

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 variable nCbS specifying the size of the current luma coding block.

~~The derivation process for a depth predicted contour pattern as specified in subclause I.8.5.8 is invoked with the sampling interval sampInt equal to 4, the sample location ( xTb, yTb ) equal to ( xCb, yCb ), and the block size nTbS equal to nCbS as inputs, and the output is a binary partition pattern wedgePattern[ x ][ y ].~~

~~For p in the range of 0 to 6, inclusive and i in the range of 0 to 1, inclusive, partSum[ p ][ i ] is set equal to 0.~~

~~for( y = 0; y < nCbS ; y += 4 )  
 for( x = 0; x < nCbS ; x += 4 ) {   
 segFlag = wedgePattern[ x ][ y ]  
 partSum[ 0 ][ ( x < ( nCbS  >>  1 ) ) ? segFlag : !segFlag ]++  
 partSum[ 1 ][ ( y < ( nCbS  >>  1 ) ) ? segFlag : !segFlag  ]++  
 if( nCbS > 8 ) { (I‑300)  
 partSum[ 2 ][ ( y < ( nCbS  >>  2 ) ) ? segFlag : !segFlag ]++  
 partSum[ 3 ][ ( y < ( nCbS  >>  2 + nCbS >> 1 ) ) ? segFlag : !segFlag ]++  
 partSum[ 4 ][ ( x < ( nCbS  >>  2 ) ) ? segFlag : !segFlag ]++  
 partSum[ 5 ][ ( x < ( nCbS  >>  2 + nCbS >> 1 ) ) ? segFlag : !segFlag ]++  
 }  
 }~~

~~The variable partIdc is derived as specified in the following:~~

~~partIdc = 0  
 maxPartSum = 0  
 for( p = 0; p < 6; p++ )  
 for( i = 0; i < 2; i++ ) {  
 if( partSum[ p ][ i ] > maxPartSum ) { (I‑301)  
 maxPartSum = partSum[ p ][ i ]  
 partIdc = p  
 }  
 }~~

~~The variables x0 and y0 are derived and the variable PartMode is modified depending on partIdc as specified in Table I‑11~~

The derivation process for a disparity sample array as specified in subclause I.8.5.5.2 is invoked with the luma locations ( xP, yP ) equal to ( xCb, yCb ), the disparity vector mvDisp equal to MvRefinedDisp[ xCb ][ yCb ], the view identifier refViewIdx equal to RefViewIdx[ xCb ][ yCb ], the variables nPbW and nPbH, equal to nCbS, and the variable partIdc equal to 1 as inputs, and the output is the array refSamples of size (nCbS)x(nCbS) and the flag horSplitFlag.

The partitioning threshold partThreshold is derived from the four corner samples if refSamples as follows:

partT = ( refSamples[ 0 ][ 0 ] + refSamples[ nCbS – 1 ][ 0 ] + refSamples[ 0 ][ nCbS – 1 ] + refSamples[ nCbS – 1 ][ nCbS – 1 ] ) >> 2

The modified partitioning mode indicator partIdc is derived as follows:

if( ( refSamples[ 0 ][ 0 ] > partT ) = = ( refSamples[ 0 ][ nCbS – 1 ] > partT ) ) {  
 partIdc = 0  
 if( nCbS > 8 ) {  
 if( ( refSamples[ nCbS >> 2 ][ 0 ] > partT ) = = ( refSamples[ ( nCbS >> 2 ) \* 3 ][ nCbS – 1 ] > partT ) )  
 partIdc = 5  
 else  
 partIdc = 4  
 }  
} else {  
 partIdc = 1  
 if( nCbS > 8 ) {  
 if( ( refSamples[ 0 ][ nCbS >> 2 ] > partT )  = = ( refSamples[ nCbS – 1 ][ ( nCbS >> 2 ) \* 3 ] > partT ) )  
 partIdc = 3  
 else  
 partIdc = 2  
 }  
}

The following applies:

MergeIdx[ x0 ][ y0 ] = merge\_idx[ xCb ][ yCb + nCbS / 2 ] (I‑302)

MergeFlag[ x0 ][ y0°] = merge\_flag[ xCb ][ yCb + nCbS / 2 ] (I‑303)

InterPredIdc[ x0 ][ y0°] = inter\_pred\_idc[ xCb ][ yCb + nCbS / 2 ] (I‑304)

For X in the range of 0 to 1, inclusive, the following applies:

PuRefIdxLX[ x0 ][ y0°] = ref\_idx\_lX[ xCb ][ yCb + nCbS / 2 ] (I‑305)

MvpLXFlag[ x0 ][ y0°] = mvp\_lX\_flag[ xCb ][ yCb + nCbS / 2 ] (I‑306)

MvdLX[ x0 ][ y0°] = MvdLX[ xCb ][ yCb + nCbS/2 ] (I‑307)

**Table I‑11 – Specification of x0, y0 and PartMode depending on partIdc**

|  |  |  |  |
| --- | --- | --- | --- |
| **partIdc** | **x0** | **y0** | **PartMode** |
| 0 | xCb + nCbS / 2 | yCb | SIZE\_Nx2N |
| 1 | xCb | yCb + nCbS / 2 | SIZE\_2NxN |
| 2 | xCb | yCb + nCbS / 4 | SIZE\_2NxnU |
| 3 | xCb | yCb + ( nCbS \* 3 / 4 ) | SIZE\_2NxnD |
| 4 | xCb + nCbS / 4 | yCb | SIZE\_nLx2N |
| 5 | xCb + ( nCbS \* 3 / 4 ) | yCb | SIZE\_nRx2N |

# 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).**

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