|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extension Development**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  9th Meeting: Sapporo, JP, 3 – 9 July 2014 | Document: JCT3V- I0124 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Title:* | **On DBBP Filtering Simplification** | | | |
| *Status:* | Input Document | | | |
| *Purpose:* | Proposal | | | |
| *Author(s) or Contact(s):* | Zhouye Gu1  Jianhua Zheng2  Nam Ling1  Philipp Zhang2 | Tel: +1-206-816-2367 Email: zgu@scu.edu  zhengjianhua@hisilicon.com | |  |
| *Source:* Santa Clara University1 and HiSilicon Technologies2 | | |  | |

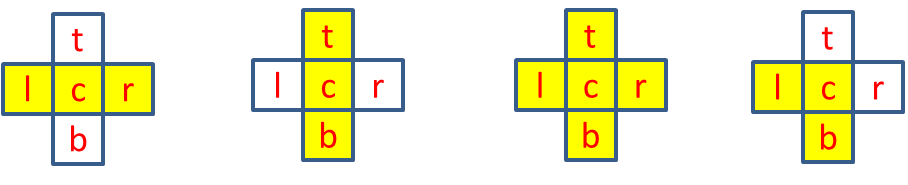
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This contribution proposes a simplified filtering process for Depth Based Block Partitioning (DBBP). It reduces the mask checking position from 4 positions to 5 positions and also reduces filtering cases from 4 cases to 2 cases. Experiment result show that this proposal simplified DBBP boundary filtering process without BD-bitrate loss under CTC.

# Introduction

In Depth Based Block Partition (DBBP) based coding, an arbitrarily shaped block partitioning for the collocated texture block is derived from a binary segmentation mask computed by the corresponding depth block. Two partitions are motion-compensated and then merged by averaging them based on the depth-based segmentation mask. The partition boundary is filtered with a smooth filter based 4 mask cases. In this proposal we reduce the filtering cases from 4 cases to 2 cases without performance loss. Figure 1 illustrated 4 mask-dependent boundary filtering cases of current design [1]. In Figure 1, ‘t’,’b’,’l’,’r’,’c’ present the pixel masks of top, bottom, left, right and center position. The position with the same background means they share the same binary mask values. In case (a) only pixels corresponding to mask ‘t’,’c’ and ‘b’ will be involved into filtering process since masks ‘t’,’c’,’b’ are different while masks ‘l’,’c’,’r’ are the same. Similarly, in case (b) only pixels corresponding to masks ‘l’,’c’ and ‘r’ will be involved into filtering process since masks ‘l’,’c’,’r’ are different while masks ‘t’,’c’,’b’ are the same. In case (c), no filtering will be used since masks ‘t’,’b’,’l’,’r’,’c’ are the same. In case (d), pixels corresponding to mask ‘t’,’b’,’l’,’r’,’c’ are all used for boundary filtering.



1. (b) (c) (d)

Figure 1. Illustration of DBBP mask-dependent boundary filtering cases

In summary the filtering conditions are

If (the masks at position ‘l’, ‘r’, ‘t’, ‘b’, 'c' are the same)

P(c)’ = P(c)

if ( the masks at position ‘l’, ‘r’, 'c' are the same && the masks at position ‘t’, ‘b’, 'c' are not the same)

P(c)’ = (P(t) + P(b) + (P(c)<<1))>>2

if (the masks at position ‘t’, ‘b’, 'c' are the same && the masks at position ‘l’, ‘r’, 'c' are not the same)

P(c)’ = (P(l) + P(r) + (P(c)<<1))>>2

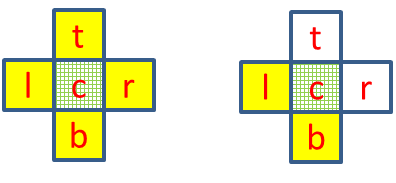
if ( the masks at position ‘l’, ‘r’, 'c' are not the same && the masks at position ‘t’, ‘b’, 'c' are not the same)

P(c)’ = (P(l) + P(r)+ P(t) + P(b) + (P(c)<<2))>>3

Where P(c)’ is the pixel value after filtering at position c. P(l),P(r), P(t), P(b) and P(c) are corresponding pixel values at position ’l’,’r’, t’,’b’,’c’.

# Proposed solution

In the above filtering process, 5 positions need to be checked and 4 filtering cases will be considered. To simplify the filtering process, we reduce the mask checking position to 4 positions and reduce the filtering cases from 4 cases (as shown in Figure 1) to 2 cases as shown in Figure 2. In Figure 2, we only check the masks of position ‘t’,’b’,’l’,’r’. If they are all the same, as shown in figure 2(a), no filtering will be used. Otherwise, if the masks of position ‘t’,’b’,’l’,’r’ are not all the same, pixels corresponding to mask ‘t’,’b’,’l’,’r’,’c’ are all used for boundary filtering as shown in Eqn. (1).



1. (b)

Figure 2. Illustration of proposed mask-dependent boundary filtering cases (mask of position ‘c’ is not check in these cases)

The filtering process is summarized:

If (the binary masks at position ‘l’, ‘r’, ‘t’, ‘b’ are all the same)

P(c)’ = P(c);

else

P(c)’ = (P(l) + P(r)+ P(t) + P(b) + (P(c)<<2))>>3 (1)

Where P(c)’ is the pixel value after filtering at position c. P(l),P(r), P(t), P(b) and P(c) are corresponding pixel values at position ’l’,’r’, t’,’b’,’c’.

# Experimental results

The proposed method is implemented on top of HTM-11.0, and simulations were performed under “common test condition” [3] and configurations.

The results are summarized in Table 1. As it is reported, the proposed method doesn’t incur BD-bitrate loss under CTC.

Table 1: BD rate results for 3-view case under CTC



# References

1. Jin Young Lee, Mikhail Mishurovskiy, Min Woo Park, and Chanyul Kim, “Partition boundary filtering in DBBP,” JCT3V-H0104, Valencia Spain, 29 Mar. – 4 Apr. 2014
2. G. Tech, K. Wegner, Y. Chen, S. Yea “Test Model 8 of 3D-HEVC and MV-HEVC” in JCT3V-H1003, Valencia Spain, 29 Mar. – 4 Apr. 2014
3. K. Müller, A. Vetro, “Common Test Conditions of 3DV Core Experiments,” JCT3V-G1100, San Jose, USA, January 2014.

# Change in WD

I 8.5.3.3.9.1 Derivation process for contour boundary filtered samples

Inputs to this process are:

* a variable nCbSL specifying the size of the current luma coding block,
* a variable nCbSX specifying the size of the current coding block,
* an (nCbSL)x(nCbSL) array segMask
* an (nCbSX)x(nCbSX) array predSamples prediction samples

Outputs to this process are:

* an modified (nCbS)x(nCbS) array predSamples of luma prediction samples

The (nCbSX)x(nCbSX) array p is set equal to predSamples and the variable n is set equal to ( nCbSL/ nCbSX ).

The values of predSamples are derived as specified in the following:

for ( y = 0; y < nCbSX; y++ )  
 for( x = 0; x < nCbSX; x++ ) {  
 tFlag = segMask[ n \* x ][ Max( 0, n \* ( y − 1 ) ) ]  
 lFlag = segMask[ Max( 0, (n \* ( x − 1 ) ) ][ n \* y ]  
 bFlag = segMask[ n \* x ][ Min( n \* ( y + 1 ), nCbSL − 1 ) ]  
 rFlag = segMask[ Min( n \* ( x + 1 ), nCbSL − 1 ) ][ n\*y ]  
 ~~cFlag = segMask[ n \* x ][ n \* y ]~~  
 filt = p[ x ][ y ]

if( ( lFlag | | rFlag | | tFlag | | bFlag ) && ( !lFlag | | !rFlag | | !bFlag | | !tFlag ) )  
 filt = ( p[ Max( 0, x − 1 ) ][ y ] + p[ x ][ Max( 0, y − 1 ) ] + ( filt << 2 ) + p[ x ][ Min( y + 1, nCbSX − 1 ) ] + p[ Min( x + 1, nCbSX − 1 ) ][ y ] ) >> 3  
 ~~if( ( lFlag | | cFlag | | rFlag ) && ( !lFlag | | !cFlag | | !rFlag ) )~~  
 ~~filt = ( p[ Max( 0, x − 1 ) ][ y ] + ( filt << 1 ) + p[ Min( x + 1, nCbS~~~~X~~~~− 1 ) ][ y ] ) >> 2~~  
 ~~if( ( tFlag | | cFlag | | bFlag ) && ( !tFlag | | !cFlag | | !bFlag ) )~~ ~~filt = ( p[ x ][ Max( 0, y − 1 ) ] + ( filt << 1 ) + p[ x ][ Min( y + 1, nCbS~~~~X~~~~− 1 ) ] ) >> 2~~  
 predSamples[ x ][ y ] = filt  
 }

# Patent rights declaration(s)

**HiSilicon Technologies, Santa Clara University and 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).**