|  |  |
| --- | --- |
| **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**  3rd Meeting: Geneva, CH, 17–23 Jan. 2013 | Document:JCT3V-C0223r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Results on Weighted Prediction for 3D-HEVC** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Jiwook Jung, Hongbin Liu  Sehoon Yea  221, Yangjae-Dong, Seocho-Gu, Seoul 137-130, Korea | Tel: Email: | +82-2-6912-6477 [jiwook.jung@lge.com](mailto:jiwook.jung@lge.com) |
| *Source:* | LG Electronics | | |

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

# Abstract

In this contribution, firstly, a SW bug-fix is provided that makes WP (Weighted Prediction) work for each view independently. Secondly, an algorithm for inheritance of WP parameters for a dependent view from the base view is proposed. A flag indicating whether the WP parameters on the dependent view will be inherited from the base view is introduced in the slice header. The proposed inheritance algorithm reportedly shows BD-rate savings, over independent WP, of 7.4%, 7.5%, 2.6% for view 1, view 2, and video only cases, respectively for the 3D fading sequences generated in the same manner as was done in the HEVC standardization process.

# Introduction

Weighted Prediction (WP) is known for its benefits in coding fading in/out sequences [1][2] and part of the HEVC specification. In the current HEVC design, WP parameters for every reference picture is coded for each frame, thereby resulting in non-negligible amount of side-information. On the other hand, in 3D video, the fading in/out effects happen in all views simultaneously with the almost same magnitude. Therefore it is expected that the same WP parameters could be re-used for dependent views for coding benefits. Based upon these observations, after providing a SW bug-fix to make WP in the HTM work properly for each view, this contribution proposes an algorithm for inheriting the WP parameters for the dependent views from the base view.

# SW bug-fix

The SW bug in the current 3D-HTM is related to the variable pcDtParam->pUsed in TComRdCostWeightPrediction::xGetSADw, which represents whether the tool InterViewSkip is on or off. When it is set to zero, the SADw value will not be properly calculated in the original code. Therefore an additional code dealing with such a case is provided.

Figures 1 and 2 show the original buggy and the bug-fixed codes, respectively.

Figure 1. The original (buggy) code

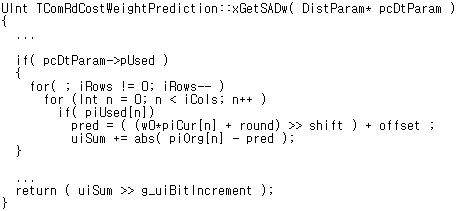
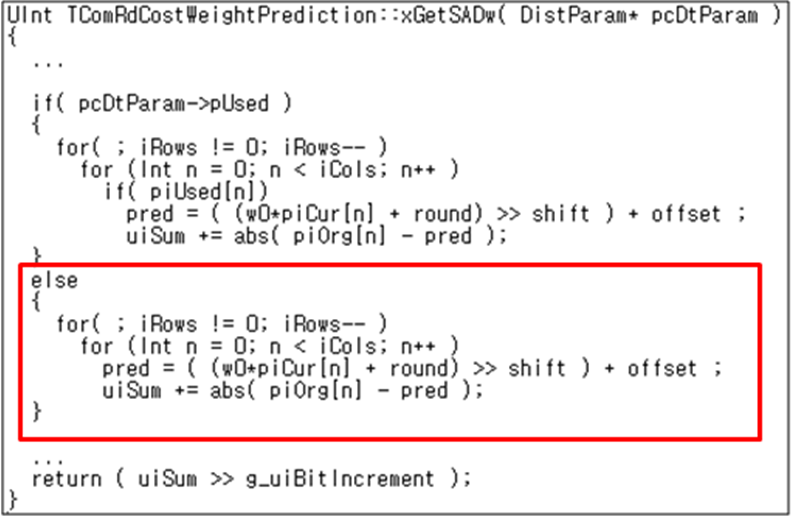


Figure 2. The bug-fixed code



# WP Parameter Inheritance

The basic idea of WP parameter inheritance is that fading is predominantly a temporal phenomenon and it is expected that the same WP parameters would work for different views as well, hence justifying the inheritance. If the flag, **base\_pred\_weight\_table\_flag,** in slice header of a dependent view is equals to 1, the WP parameters for the dependent view are inherited from the base view. Figures 3 and 4 present an illustration and a pseudo code of the inheritance process, respectively.

The proposed algorithm consists of the following two parts:

1. Sets the WP parameters for the inter-view reference picture for the dependent view to the default, i.e., 1 and 0 for weight and offset, respectively. Since the base-view does not have the inter-view reference, there is no WP parameters associated with such a reference picture to copy from the base-view.
2. Searches the reference lists of the base-view to find the reference picture with the same POC as that of the current candidate reference picture for the dependent-view. If found, the WP parameters will be copied. Otherwise, set to the default.

Figure 3. An example of WP parameter inheritance

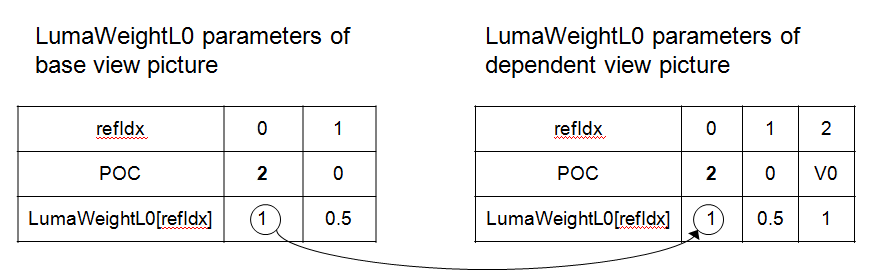


Figure 4. A pseudo code for WP parameter inheritance

|  |
| --- |
| if (base\_pred\_weight\_table\_flag)  {  reset WP parameter table of current dependent view  for( X=0 ; X< (B\_SLICE?2:1) ; X++ )  {  for (i = 0; i<=num\_ref\_idx\_lX\_active\_minus1; i++)  {  for (j = 0; j<base\_num\_ref\_idx\_lX\_active\_minus1; j++)  {  if (PicOrderCnt ( RefIdxLX[i] ) == PicOrderCnt ( BaseRefIdxLX[j]) )  {  copy WP parameters of base view to the current view;  }  }  }  }  } |

# Test sequences

Fading in/out sequences were generated in the same manner as done for HEVC. In JCTVC-E041 [1], a software was provided that could create linear fading sequences. For this contribution, we used the same fading conditions from [1][2]: a linear fade is applied to the first 2 seconds of the 3DV. A linear fade-out is applied during the first half (0~1 second) and a linear fade-in is applied during the second half (1~2 second) (see Figure 5). The fade strength is the range of [0.25, 1]. Only texture videos were encoded.

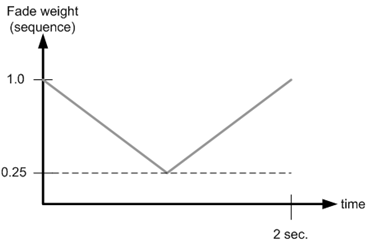


Figure 5. Fading in/out strength

# Results

Table 1 shows the performance of the WP with the suggested bug-fix for the fading test sequences with respect to HTM5.0.1. Each view performs WP independently and the WP parameters were coded for each view without inheritance. Also the weight and offset values were set to the default (1 and 0) for inter-view reference pictures when applying the WP for the dependent views.

Table 1. WP without parameter inheritance vs. HTM5.0.1 on fading 3DV sequences

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video only | enc time | dec time |
| Balloons | -36.0% | -19.5% | -18.3% | -29.7% | 147.0% | 102.6% |
| Kendo | -18.6% | -4.6% | -2.5% | -13.5% | 186.2% | 100.4% |
| Newspapercc | -33.0% | -21.9% | -23.5% | -29.2% | 127.7% | 97.8% |
| GhostTownFly | -29.6% | -2.0% | -2.8% | -24.6% | 159.6% | 106.2% |
| PoznanHall2 | -32.1% | -9.7% | -15.9% | -24.8% | 160.2% | 100.4% |
| PoznanStreet | -27.5% | -16.3% | -14.5% | -24.6% | 128.5% | 101.3% |
| UndoDancer | -37.2% | -19.0% | -23.7% | -33.3% | 150.7% | 98.6% |
| 1024x768 | -29.2% | -15.3% | -14.8% | -24.1% | 151.8% | 100.3% |
| 1920x1088 | -31.6% | -11.8% | -14.2% | -26.8% | 149.2% | 101.6% |
| average | **-30.6%** | **-13.3%** | **-14.5%** | **-25.7%** | **150.3%** | **101.0%** |

Table 2 compares the results with vs. without the proposed WP parameter inheritance.

Table 2. WP with vs. without parameter inheritance on fading 3DV sequences

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video only | enc time | dec time |
| Balloons | 0.0% | -8.7% | -8.8% | -3.4% | 103.1% | 101.0% |
| Kendo | 0.0% | -8.6% | -8.9% | -3.3% | 100.6% | 96.2% |
| Newspapercc | 0.0% | -7.5% | -7.3% | -2.8% | 104.0% | 94.3% |
| GhostTownFly | 0.0% | -9.7% | -8.2% | -1.9% | 101.2% | 100.6% |
| PoznanHall2 | 0.0% | -8.1% | -6.8% | -2.9% | 101.4% | 100.8% |
| PoznanStreet | 0.0% | -4.2% | -5.5% | -1.3% | 104.1% | 100.0% |
| UndoDancer | 0.0% | -4.6% | -3.2% | -0.8% | 101.5% | 100.9% |
| 1024x768 | 0.0% | -8.3% | -8.4% | -3.2% | 102.6% | 97.1% |
| 1920x1088 | 0.0% | -6.7% | -5.9% | -1.7% | 102.0% | 100.6% |
| **Average** | **0.0%** | **-7.4%** | **-7.0%** | **-2.3%** | **102.3%** | **99.1%** |

Table 3 shows the result with the proposed WP parameter inheritance against the HTM5.0.1 (i.e., without WP)

Table 3. WP with inheritance vs. HTM5.0.1 on fading 3DV sequences

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video only | enc time | dec time |
| Balloons | -36.0% | -26.6% | -25.5% | -32.1% | 151.6% | 103.7% |
| Kendo | -18.6% | -12.9% | -11.3% | -16.4% | 187.3% | 96.6% |
| Newspapercc | -33.0% | -27.9% | -29.2% | -31.3% | 132.8% | 92.2% |
| GhostTownFly | -29.6% | -11.7% | -10.9% | -26.1% | 161.5% | 106.8% |
| PoznanHall2 | -32.1% | -17.2% | -21.7% | -27.1% | 162.4% | 101.2% |
| PoznanStreet | -27.5% | -20.0% | -19.4% | -25.6% | 133.7% | 101.2% |
| UndoDancer | -37.2% | -23.0% | -26.4% | -33.9% | 153.0% | 99.5% |
| 1024x768 | -29.2% | -22.5% | -22.0% | -26.6% | 155.6% | 97.4% |
| 1920x1088 | -31.6% | -18.0% | -19.6% | -28.1% | 152.2% | 102.1% |
| **Average** | **-30.6%** | **-19.9%** | **-20.6%** | **-27.5%** | **153.7%** | **100.1%** |

# Conclusion

In this contribution, firstly, a SW bug-fix is provided that makes WP (Weighted Prediction) work for each view independently. Secondly, an algorithm for inheritance of WP parameters for a dependent view from the base view is proposed. Experiments for fading sequences show that inheritance of WP parameters further improves coding efficiency of dependent views by -7.4%, -7.5%, and -2.6% for V1, V2, and coded video, respectively. It is recommended that the proposed weighted prediction parameter inheritance method be adopted into the HTM.

# References

1. Hongbin Liu, Jiwook Jung, Jaewon Sung, Jie Jia, Sehoon Yea, “3D-CE2.h : Results of Illumination Compensation for Inter-View Prediction”, JCT3V-B0045, Shanghai, Oct. 2012.
2. Philippe Bordes, “Weighted Prediction ", JCTVC-E041, Geneva, March 2011.
3. Philippe Bordes, “Weighted Prediction ", JCTVC-F265, Torino, July 2011.
4. Peng Yin, Jill Boyce, Purvin Pandit, “Weighted Prediction in SVC ", JVT-P064, Poznan, July 2005.
5. Heiko Schwarz, Dmytro Rusanovskyy, "Common Test Conditions for 3DV experimentation”, ISO/IEC JTC1/SC29/WG11, JCT3V-A1100, Stockholm, July 2012.

# Patent rights declaration(s)

**LG Electronics 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).**

# Suggested WD changes

Table 4 shows the changed slice header syntax for the proposed algorithm.

Table 4. Syntax of slice header

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **first\_slice\_in\_pic\_flag** | u(1) |
| **…** |  |
| if( !entropy\_slice\_flag ) { |  |
| **…** | **…** |
| if( ( weighted\_pred\_flag && slice\_type = = P) | |  ( weighted\_bipred\_flag = = 1 && slice\_type = = B ) ) { |  |
| if( ViewIdx>0 && !DepthFlag ) { |  |
| **base\_pred\_weight\_table\_flag** | u(1) |
| if( base\_pred\_weight\_table\_flag == 0 ) |  |
| pred\_weight\_table( ) |  |
| } |  |
| **…** |  |
| } |  |

**base\_pred\_weight\_table\_flag** equals to 1 specifies that the weighted prediction parameter for the texture of dependent view are inherited from the texture picture of base view. When the base\_pred\_weight\_table\_flag is not present, it is inferred to be 0.

G.8.3.4.2 Initialization process for reference picture lists

The specifications in subclause 8.3.4.2 apply. If base\_pred\_weight\_table\_flag is equal to 1 the specification in subclause G.8.3.7 apply at the end of the specification in subclause 8.3.4.2.

G.8.3.7 Weighted prediction parameters inheritance

The base view picture is an inter-view reference picture of current picture whose ViewIdx is equal to 0 and let viewIdxBase be the variable ViewIdx of the base view picture.

The LumaWeightLX[ ViewIdx ], luma\_offset\_lX[ ViewIdx ], ChromaWeightLX[ ViewIdx ], ChromaOffsetLX[ ViewIdx ] are derived from the LumaWeightLX[ viewIdxBase ], luma\_offset\_lX[ viewIdxBase ], ChromaWeightLX[ viewIdxBase ], ChromaOffsetLX[ viewIdxBase ] as follows.

For X being replaced by 0 and 1, the following applies. A variable num\_ref\_idx\_lX\_active\_minus1\_base is set to the num\_ref\_idx\_lX\_minus1 of the base view picture. A reference picture list RefPicListLXBase is set to the reference picture list RefPicListLX of the base view picture.

for( rIdx=0 ; rIdx<num\_ref\_idx\_lX\_active\_minus1+1; rIdx++ )

Let refPicViewIdx be the variable ViewIdx of the RefPicListLX[ rIdx ]

for( rIdxBase = 0; rIdxBase < num\_ref\_idx\_lX\_active\_minus1\_base+1; rIdxBase ++ )

if( refPicViewIdx != ViewIdx )

LumaWeightLX[ ViewIdx ][ rIdx ] is set 2luma\_log2\_weight\_denom

ChromaWeightLX[ ViewIdx ][ rIdx ][ 0 ] is set to 2ChromaLog2WeightDenom

ChromaWeightLX[ ViewIdx ][ rIdx ][ 1 ] is set to 2ChromaLog2WeightDenom

luma\_offset\_lX[ ViewIdx ][ rIdx ], ChromaOffsetLX[ ViewIdx ][ rIdx ][ 0 ], and ChromaOffsetLX[ ViewIdx ][ rIdx ][ 1 ] are set to 0

Otherwise, if PicOrderCnt( RefPicListLX[ rIdx ] is equal to PicOrderCnt( RefPicListXBase[ rIdxBase ]

LumaWeightLX[ ViewIdx ][ rIdx ] is set to LumaWeightLX[viewIdxBase][ rIdxBase ]

luma\_offset\_lX[ ViewIdx ][ rIdx ] is set to luma\_offset\_lX[viewIdxBase ][ rIdxBase ] ChromaWeightLX [ ViewIdx ][ rIdx ][ 0 ] is set to ChromaWeightLX[viewIdxBase ][ rIdxBase ][ 0 ] ChromaWeightLX [ ViewIdx ][ rIdx ][ 1 ] is set to ChromaWeightLX[viewIdxBase ][ rIdxBase ][ 1 ]

G.8.5.2.2 Decoding process for inter prediction samples

The specifications in subclause 8.5.2.2 apply with the following modification:

– All invocations of the process specified in subclause 8.5.2.2.3 are replaced with invocations of the process specified in subclause .2.3

.

G.8.5.2.2.3  Weighted sample prediction process

The specifications in subclause 8.5.2.2.3 apply with the following modification:

– All invocations of the process specified in subclause 8.5.2.2.3.2 are replaced with invocations of the process specified in subclause .2.3.2

G.8.5.2.2.3.2 Weighted sample prediction process

Inputs to this process are:

– a location ( xB, yB ) specifying the top-left sample of the current prediction block relative to the top left sample of the current coding block,

– the width and height of this prediction block, nPbW and nPbH,

– two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,

– prediction list utilization flags, predFlagL0 and predFlagL1,

– reference indices, refIdxL0 and refIdxL1,

– a variable cIdx specifying colour component index,

– a bit depth of samples, bitDepth.

Outputs of this process are:

– the (nPbW)x(nPbH) array predSamples of prediction sample values.

The variables shift1 is set equal to 14 – bitDepth.

The variables log2WD, o0, o1, and w0, w1 are derived as follows.

– If cIdx is equal to 0 for luma samples,

log2WD = luma\_log2\_weight\_denom + shift1 (8‑216)

w0 = LumaWeightL0[ ViewIdx ][ refIdxL0 ] (8‑217)

w1 = LumaWeightL1[ ViewIdx ][ refIdxL1 ] (8‑218)

o0 = luma\_offset\_l0[ ViewIdx ][ refIdxL0 ] \* ( 1 << ( bitDepth − 8 ) ) (8‑219)

o1 = luma\_offset\_l1[ ViewIdx ][ refIdxL1 ] \* ( 1 << ( bitDepth − 8 ) ) (8‑220)

– Otherwise (cIdx is not equal to 0 for chroma samples),

log2WD = ChromaLog2WeightDenom + shift1 (8‑221)

w0 = ChromaWeightL0[ ViewIdx ][ refIdxL0 ][ cIdx − 1 ] (8‑222)

w1 = ChromaWeightL1[ ViewIdx ][ refIdxL1 ][ cIdx − 1 ] (8‑223)

o0 = ChromaOffsetL0[ ViewIdx ][ refIdxL0 ][ cIdx − 1 ] \* ( 1 << ( bitDepth − 8 ) ) (8‑224)

o1 = ChromaOffsetL1[ ViewIdx ][ refIdxL1 ][ cIdx − 1 ] \* ( 1 << ( bitDepth − 8 ) ) (8‑225)

The prediction sample predSamples[ x ][ y ] with x = 0..(nPbW)−1 and y = 0..(nPbH)−1 are derived as follows:

– If the predFlagL0 is equal to 1 and predFlagL1 is equal to 0, the prediction samples are derived by:

if( log2WD >= 1 )   
 predSamples[ x ][ y ] = Clip3( 0, ( 1 << bitDepth ) − 1,   
 ( (predSamplesL0[ x ][ y ] \* w0 + 2log2WD − 1) >> log2WD ) + o0 ) (8‑226)  
else  
 predSamples[ x ][ y ] = Clip3( 0, ( 1 << bitDepth ) − 1, predSamplesL0[ x ][ y ] \* w0 + o0 ) (8‑227)

– Otherwise, if the predFlagL0 is equal to 0 and predFlagL1 is equal to 1, the final predicted sample values predSamples [ x ][ y ] are derived by

if( log2WD >= 1 )  
 predSamples[ x ][ y ] = Clip3( 0, ( 1 << bitDepth ) − 1,   
 ( (predSamplesL1[ x ][ y ] \* w1 + 2log2WD − 1) >> log2WD ) + o1 ) (8‑228)  
else  
 predSamples[ x ][ y ] = Clip3( 0, ( 1 << bitDepth ) − 1, predSamplesL1[ x ][ y ] \* w1 + o1 ) (8‑229)

– Otherwise, the final predicted sample values predSamples[ x ][ y ] are derived by

predSamples[ x ][ y ] = Clip3( 0, ( 1 << bitDepth ) − 1,   
 ( predSamplesL0 [ x ][ y ] \* w0 + predSamplesL1[ x ][ y ] \* w1 +   
 ((o0 + o1 + 1) << log2WD) ) >> (log2WD + 1) ) (8‑230)