|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  5th Meeting: Geneva, 16-23 March, 2011 | Document: JCTVC-E041r2  WG11 Number: m19545 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Weighted Prediction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Philippe Bordes  1, av. De Belle-Fontaine – CS17616  35576 Cesson-Sévigné Cedex - France | Tel: Email: | +33-2-99-27-32-42 [philippe.bordes@technicolor.com](mailto:philippe.bordes@technicolor.com) |
| *Source:* | Technicolor | | |

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

# Abstract

This document presents results of Weighted Prediction (WP) implementation into HM2.0 . The WP has been specially designed to compensate Illumination variation in video sequences. It is part of the AVC standard and this feature is useful in video encoder and video splicing applications in particular. Experimental results are made on Fade sequences and comparison is made with other Illumination Compensation (IC) tools.

The results reports that WP has a large gain (10-38% depending on the config) for the test sequences with linear fade and the comparison states WP outperforms other IC tools. Furthermore, it is reported WP has no impact on coding efficiency for the regular test sequences.

# Introduction

WP has been specially designed to compensate Illumination variation in video sequences. The principle consists in replacing the inter prediction signal *P* by a linear weighted prediction signal *P’*:

Uni-directional prediction: *P’ = w × P + o*

Bi-directional prediction: *P’ = (w0 × P0 + o0 + w1 × P1 + o1) / 2*

Originally, WP was specially designed to handle Fading and Cross-Fading. However, better efficiency is obtained when the reference pictures can be chosen arbitrary at the encoder as with the MMCO[[1]](#footnote-2) system in AVC.

WP can manage local illumination variations using reference picture list with duplicate references [1]. It can also be used to generate cross-fading transitions in video splicing applications.

WP has very limited overhead in PPS (3 bits) and slice header contains only non default WP scaling values (wi,oi).

At last, since WP is an optional PPS parameter, it may be switch on only when necessary.

# The Weighted Prediction principle

The WP is an adaptive weighting of Inter prediction signal to compensate Illumination variation between the reference pictures and the current one. It can be applied in uni-prediction (1) or in bi-prediction (2).

predPartC[ x, y ] = ( ( predPartL0C[ x, y ] \* w0 +) >> shift ) + o0 (1)

predPartC[ x, y ] = ( ( predPartL0C[ x, y ] \* w0 + predPartL1C[ x, y ] \* w1 + shift ) >>   
 ( shift + 1 ) ) + ( ( o0 + o1 + 1 ) >> 1 ) (2)

Depending on the type of slice (P\_SLICE or B\_SLICE), one can choose the weighting algorithm:

* Default: the default HEVC uni-prediction or bi-prediction.
* Explicit: the weighting factors are transmitted explicitly in the slice header.
* Implicit: the weighting factors for bi-prediction are derived from the distance of the current POC with the POC of the reference pictures, relatively to the distance between references POC (B\_SLICE only).

The presence of weighting factors and the weighting method to use are indicated in the PPS with **weighted\_pred\_flag** and **weighted\_bipred\_idc** (Table 1). If present, the adaptive weighting factor parameters are transmitted in the slice header for each allowable reference picture. Syntax for coding of the weighting factors is similar to that done in AVC and is detailed in Annex.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_SLICE** | |  | **B\_SLICE** | |
| **Weighted\_pred\_flag** | **Weighting method** |  | **Weighted\_bipred\_idc** | **Weighting method** |
| 0 | Default |  | 0 | Default |
| 1 | Explicit |  | 1 | Explicit |
|  |  |  | 2 | Implicit |

Table ‑: Interpretation of weighted\_pred\_flag and weighted\_bipred\_idc values.

Explicit: If present, the weighting factor parameters allow deriving the corresponding weights (wi) and offset (oi) values for explicit weighting prediction (equations 1 and 2), both in uni-prediction and bi-prediction. If not present, default (HEVC original) weighting is used.

Implicit: The weights (wi) and offset (oi) prediction are determined with the following equation (3):

w0 = 1 – α w1=α where: , oi=0 (3)

In equation (3), α Є [-1.;2.]. Practically, to avoid floating point multiplication the equation (3) is implemented using only integer multiplications and shift (cf. Annex). Implicit weighting method is used in case of linear fade or cross-fade typically.

# Weighted Prediction implementation in HM

The Weighted Prediction has been implemented in HM2.0 and is provided with this proposal. The corresponding code is encapsulated with #defines WEIGHT\_PRED. Three new input parameters have been added in the cfg file:

weighted\_pred\_flag : 0 (default) or 1

weighted\_bipred\_idc : 0 (default), 1 or 2

The main change in HM software code concern the motion estimation (SAD, SSE, HAD…)[[2]](#footnote-3) and the motion compensation (inter prediction). The syntax adds (PPS and slice header) are minimal.

Additionally, code encapsulated with #defines WEIGHT\_PRED\_TEST allows for loading explicit weighting parameters:

WpScalingFile : input file containing explicit weighting parameters

write\_refPicList\_flag: 0 (default) or 1 (write POC of ref. pictures in refPicList.txt)

To test and validate the weighting prediction, a simple Fading tool is provided. This software tool creates fading and cross-fading sequences, and a file containing (non optimal) explicit weighting parameters may be generated to test explicit WP in HM.

In case both ‘HierarchicalCoding’ and ‘LowDelayCoding’ are zero, a simple GOP structure of size “GOPsize” is built for Cross-Fading experiments, with non reference B frames in between I reference frames (see 4.3).

# Experimental results

## Weighting Prediction design tool

A simple software tool for building Fading-to-black and Cross-fading sequences is provided. It allows setting the corresponding explicit weighting prediction parameters too, in a format compatible with option ‘WpScalingFile’ of the HM2.0 encoder. To use explicit WP, the experimental procedure is described below:

1. Run HM Encoder with option write\_refPicList\_flag=1. A file *refPicList.txt* is created containing the valid reference POC list for each frames.
2. Run the Fading tool software with *refPicList.txt* in input argument. A file containing explicit WP parameters will be created (ex: *outWPexplicit.txt*).
3. Run HM Encoder with options weighted\_pred\_flag=1, weighted\_bipred\_idc=1 and WpScalingFile= *outWPexplicit.txt*.

## Experimental results with linear Fade-to-black

In the past, for experiments involving weighted prediction in H.264/AVC, linear and sigmoid fades varying in length from 1 to 3 seconds were applied to the regular test sequences [2][3][4].

In our experiments, we use same conditions as in [5]: a linear fade is applied to the first 2 seconds of the regular sequence set and the experiments are done with the 2-second sequences. A linear fade-out (sequence to black) is applied in the 1st half and a linear fade-in (black to sequence) in the 2nd half (Figure 1). The fade strength is limited to the range [0.25, 0.75] instead of [0, 1] to avoid extreme PSNR numbers at the frame level that cause abnormal rate-distortion graphs.



Figure : Linear Fade-to-black scenario.

Table 4‑1 indicates the BD rate gain obtained with explicit WP on High Efficiency RA and LD Fading sequences. For all the sequences and both configurations the average BD rate gain are significant: (RA: 15,1%) and (LD: 20,2%).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Random access HE** | | | **Low Delay HE** | | |
|  | **Y BD-rate** | **U BD-rate** | **V BD-rate** | **Y BD-rate** | **U BD-rate** | **V BD-rate** |
| **Class A** | -19,5 | -14,5 | -13,7 |  |  |  |
| **Class B** | -17,3 | -12,2 | -11,9 | -24,1 | -14,2 | -12,6 |
| **Class C** | -12,9 | -7,2 | -8,4 | -11,7 | -5,9 | -5,4 |
| **Class D** | -12,5 | -7,5 | -7,7 | -10,2 | -5,9 | -4,7 |
| **Class E** |  |  |  | -38,6 | -19,3 | -20,2 |
| **All** | -15,1 | -9,9 | -10,1 | -20,2 | -11,0 | -10,3 |
| **Enc. time** | **125%** | | | **118%** | | |
| **Dec. time** | **102%** | | | **101%** | | |

Table ‑: BD rate gain (%) with HM2.0 and linear Fade (WP explicit).

WP implicit has been designed for bi-directional prediction or bi-prediction with limited distance between current poc and reference poc. Then, it is not applicable for LD case. RA-HE uses Hierarchical Coding operating somewhat similar than WP implicit, that is why the BD gains are not so significant (Table 4‑2).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Random access HE** | | |
|  | **Y BD-rate** | **U BD-rate** | **V BD-rate** |
| **Class A** | -0,7 | -1,1 | -1,0 |
| **Class B** | -0,5 | -0,6 | -0,6 |
| **Class C** | -0,4 | -0,3 | -0,5 |
| **Class D** | -1,5 | -1,2 | -0,8 |
| **Class E** |  |  |  |
| **All** | -0,8 | -0,7 | -0,7 |
| **Enc. time** | **139%** | | |
| **Dec. time** | **102%** | | |

Table ‑: BD rate gain (%) with HM2.0 and linear Fade (WP implicit).

In 3rd JCT-VC meeting, TE12 presented performance of several Illumination Compensation tools (PBIC, SIFO and GRefMode), using same Fading sequences and coding conditions [5]. All the results are summarized in Table 4‑3 for High-Efficiency. For all coding conditions (RA and LD) and all classes, explicit WP out-performs the other tools.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **High Efficiency** | | | | | | | |
| **Tool** | **WP Explicit** | | **PBIC** | | **SIFO** | | **GRefMode** | |
| **Condition** | **RA** | **LD** | **RA** | **LD** | **RA** | **LD** | **RA** | **LD** |
| **Ave [%]** | 15,1 | 20,2 | 10.9 | 13.7 | 4.7 | 5.1 | 14.2 | 5.8 |
| **Min [%]** | 7 | 5,1 | 0.9 | 5.2 | -1.5 | 1.4 | 7.5 | 1.1 |
| **Max [%]** | 27,4 | 46,4 | 32.4 | 30.0 | 11.4 | 13.6 | 28.4 | 17.7 |
| **Ave of RA & LD[%]** | 17,6 | | 12.3 | | 4.9 | | 9.9 | |

Table ‑: Comparison of BD rate gain (%) of WP and other Illumination Compensation tools.

# Conclusion

In this contribution, implementation of Weighted Prediction (WP) is proposed. Experiments results done with 2 seconds fading sequences are presented. While WP has no impact for regular sequence set, large gains (10-38%) can be obtained for sequences with linear fade, with almost no impact on decoder run times.

It is stated WP is a simple tool with negligible overhead. In AVC, WP has been adopted and is used in many proprietary applications (ex: video encoders, video splicers…).

Therefore it is recommended to adopt WP in HEVC and to include this WP implementation into HM.

# Patent rights declaration(s)

**Technicolor may have IPR 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. Peng Yin, Tourapis A.M., Boyce J., “Localized Weighted Prediction for Video Coding,” IEEE International Symposium on Circuits and systems,　ISCAS 2005.
2. Jill Boyce, “Changes to Adaptive Reference Picture Weighting,” JVT Document, JVT-E060, Geneva,　October 2002.
3. Jill Boyce, “Weighted Prediction Clean-up,” JVT Document, JVT-F034, Awaji, December, 2002.
4. Yoshihiro Kikuchi and Takeshi Chujoh,” Simplification of the Weighted Prediction and Verification result,” JVT Document, JVT-F077, Awaji, December, 2002.
5. A.Fujibayashi, S.Kanumuri and F.Bossen,” TE12: Performance of Partition Based Illumination Compensation (PBIC),” JVT Document, JCTVC-C041, Guangzhou, October, 2010.

# Annex: Weighted Prediction syntax

**PPS Parameters**

**weighted\_pred\_flag** equal to 0 specifies that weighted prediction shall not be applied to P and SP slices. weighted\_pred\_flag equal to 1 specifies that weighted prediction shall be applied to P and SP slices.

**weighted\_bipred\_idc** equal to 0 specifies that the default weighted prediction shall be applied to B slices. weighted\_bipred\_idc equal to 1 specifies that explicit weighted prediction shall be applied to B slices. weighted\_bipred\_idc equal to 2 specifies that implicit weighted prediction shall be applied to B slices. The value of weighted\_bipred\_idc shall be in the range of 0 to 2, inclusive.

**Slice Header Parameters** present if (weighted\_pred\_flag && sliceP) or (weighted\_bipred\_idc=1 & sliceB)

**luma\_log2\_weight\_denom** is the base 2 logarithm of the denominator for all luma weighting factors. The value of luma\_log2\_weight\_denom shall be in the range of 0 to 7, inclusive.

**chroma\_log2\_weight\_denom** is the base 2 logarithm of the denominator for all chroma weighting factors. The value of chroma\_log2\_weight\_denom shall be in the range of 0 to 7, inclusive.

**luma\_weight\_l0\_flag** equal to 1 specifies that weighting factors for the luma component of list 0 prediction are present. luma\_weight\_l0\_flag equal to 0 specifies that these weighting factors are not present.

**luma\_weight\_l0[** i **]** is the weighting factor applied to the luma prediction value for list 0 prediction using RefPicList0[ i ]. When luma\_weight\_l0\_flag is equal to 1, the value of luma\_weight\_l0[ i ] shall be in the range of –128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, luma\_weight\_l0[ i ] shall be inferred to be equal to 2luma\_log2\_weight\_denom for RefPicList0[ i ].

**luma\_offset\_l0[** i **]** is the additive offset applied to the luma prediction value for list 0 prediction using RefPicList0[ i ]. The value of luma\_offset\_l0[ i ] shall be in the range of –128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, luma\_offset\_l0[ i ] shall be inferred as equal to 0 for RefPicList0[ i ].

**chroma\_weight\_l0\_flag** equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction are present. chroma\_weight\_l0\_flag equal to 0 specifies that these weighting factors are not present.

**chroma\_weight\_l0[** i **][** j **]** is the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr. When chroma\_weight\_l0\_flag is equal to 1, the value of chroma\_weight\_l0[ i ][ j ] shall be in the range of –128 to 127, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** chroma\_weight\_l0[ i ][ j ] shall be inferred to be equal to 2chroma\_log2\_weight\_denom for RefPicList0[ i ].

**chroma\_offset\_l0[** i **][** j **]** is the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr. The value of chroma\_offset\_l0[ i ][ j ] shall be in the range of -128 to 127, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** chroma\_offset\_l0[ i ][ j ] shall be inferred to be equal to 0 for RefPicList0[ i ].

**luma\_weight\_l1\_flag, luma\_weight\_l1**, **luma\_offset\_l1**, **chroma\_weight\_l1\_flag**, **chroma\_weight\_l1**, **chroma\_offset\_l1** have the same semantics as luma\_weight\_l0\_flag, luma\_weight\_l0, luma\_offset\_l0, chroma\_weight\_l0\_flag, chroma\_weight\_l0, chroma\_offset\_l0, respectively, with l0, list 0, and List0 replaced by l1, list 1, and List1, respectively.

**Weighting Prediction algorithm:**

**weighted\_pred\_flag=0 default weighted prediction process ( ½ ; ½ )**

predFlagL0=1, predFlagL1=0 => predPartC[ x, y ] = predPartL0C[ x, y ]

predFlagL0=0, predFlagL1=1 => predPartC[ x, y ] = predPartL1C[ x, y ]

predFlagL0=1, predFlagL1=1 => predPartC[ x, y ] = ( predPartL0C[ x, y ] + predPartL1C[ x, y ] + 1 ) >> 1

**weighted\_pred\_flag=1 weighted sample prediction process ( w0 ; w1 )**

predFlagL0=1, predFlagL1=0 :

if( logWD >= 1 )   
 predPartC[ x, y ] = Clip1C( ( ( predPartL0C[ x, y ] \* w0 + 2logWD - 1 ) >> logWD ) + o0 )   
else  
 predPartC[ x, y ] = Clip1C( predPartL0C[ x, y ] \* w0 + o0 )

predFlagL0=0, predFlagL1=1 :

if( logWD >= 1 )  
 predPartC[ x, y ] = Clip1C( ( ( predPartL1C[ x, y ] \* w1 + 2logWD - 1 ) >> logWD ) + o1 )   
else  
 predPartC[ x, y ] = Clip1C( predPartL1C[ x, y ] \* w1 + o1 )

predFlagL0=1, predFlagL1=1 :

predPartC[ x, y ] = Clip1C( ( ( predPartL0C[ x, y ] \* w0 + predPartL1C[ x, y ] \* w1 + 2logWD ) >>   
 ( logWD + 1 ) ) + ( ( o0 + o1 + 1 ) >> 1 ) )

**weighted\_bipred\_idc=2 in B slice** **implicit weighted prediction (applicable for B slices only):**

logWD = 5, o0 = 0, o1 = 0

If DiffPicOrderCnt( pic1, pic0 ) is equal to 0 or one or both of pic1 and pic0 is marked as "used for long-term reference" or ( DistScaleFactor >> 2 ) < -64 or ( DistScaleFactor >> 2 ) > 128, w0 and w1 are derived as:

w0=32, w1=32

Otherwise:

w0 = 64 – (DistScaleFactor >> 2), w1 = DistScaleFactor >> 2

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) - PicOrderCnt( picB ) (8-2)

tx = ( 16 384 + Abs( td / 2 ) ) / td (8-194)

DistScaleFactor = Clip3( -1024, 1023, ( tb \* tx + 32 ) >> 6 ) (8-195)

tb = Clip3( -128, 127, DiffPicOrderCnt( currPicOrField, pic0 ) ) (8-198)

td = Clip3( -128, 127, DiffPicOrderCnt( pic1, pic0 ) ) (8-199)

**weighted\_pred\_flag=1 in P slice, or weighted\_bipred\_idc=1 in B slice,** **explicit weighted prediction :**

Luma component:

logWD = luma\_log2\_weight\_denom (8-284)

w0 = luma\_weight\_l0[refIdxL0] (8-285)

w1 = luma\_weight\_l1[refIdxL1] (8-286)

o0 = luma\_offset\_l0[refIdxL0] \* ( 1 << ( BitDepthY – 8 ) ) (8-287)

o1 = luma\_offset\_l1[refIdxL1] \* ( 1 << ( BitDepthY – 8 ) ) (8-288)

Chroma component (iCbCr = 0 for Cb, iCbCr = 1 for Cr):

logWD = chroma\_log2\_weight\_denom (8-289)

w0 = chroma\_weight\_l0[refIdxL0][ iCbCr ] (8-290)

w1 = chroma\_weight\_l1[refIdxL1][ iCbCr ] (8-291)

o0 = chroma\_offset\_l0[refIdxL0][ iCbCr ] \* ( 1 << ( BitDepthC – 8 ) ) (8-292)

o1 = chroma\_offset\_l1[refIdxL1][ iCbCr ] \* ( 1 << ( BitDepthC – 8 ) ) (8-293)

If predFlagL0=1 and predFlagL=1, the following constraint shall be obeyed (explicit weighted prediction only, because always the case in implicit):

-128 <= w0 + w1 <= ( ( logWD = = 7 ) ? 127 : 128 ) (8‑294)

NOTE – For explicit mode weighted prediction with logWD equal to 7, when one of the two weights w0 or w1 is inferred to be equal to 128 (as a consequence of luma\_weight\_l0\_flag, luma\_weight\_l1\_flag, chroma\_weight\_l0\_flag, or chroma\_weight\_l1\_flag equal to 0), the other weight (w1 or w0) must have a negative value in order for the constraint expressed in Equation to hold (and therefore the other flag luma\_weight\_l0\_flag, luma\_weight\_l1\_flag, chroma\_weight\_l0\_flag, or chroma\_weight\_l1\_flag must be equal to 1).

**Syntax:**

* se(v): signed integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause .
* u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read\_bits( n ) interpreted as a binary representation of an unsigned integer with most significant bit written first.
* ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause .

|  |  |  |
| --- | --- | --- |
| pred\_weight\_table( ) { | C | Descriptor |
| **luma\_log2\_weight\_denom** | 2 | ue(v) |
| if( chroma\_format\_idc != 0 ) |  |  |
| **chroma\_log2\_weight\_denom** | 2 | ue(v) |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) { |  |  |
| **luma\_weight\_l0\_flag** | 2 | u(1) |
| if( luma\_weight\_l0\_flag ) { |  |  |
| **luma\_weight\_l0[** i **]** | 2 | se(v) |
| **luma\_offset\_l0[** i **]** | 2 | se(v) |
| } |  |  |
| if ( chroma\_format\_idc != 0 ) { |  |  |
| **chroma\_weight\_l0\_flag** | 2 | u(1) |
| if( chroma\_weight\_l0\_flag ) |  |  |
| for( j =0; j < 2; j++ ) { |  |  |
| **chroma\_weight\_l0[** i **][** j **]** | 2 | se(v) |
| **chroma\_offset\_l0[** i **][** j **]** | 2 | se(v) |
| } |  |  |
| } |  |  |
| } |  |  |
| if( slice\_type = = B ) |  |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) { |  |  |
| **luma\_weight\_l1\_flag** | 2 | u(1) |
| if( luma\_weight\_l1\_flag ) { |  |  |
| **luma\_weight\_l1[** i **]** | 2 | se(v) |
| **luma\_offset\_l1[** i **]** | 2 | se(v) |
| } |  |  |
| if( chroma\_format\_idc != 0 ) { |  |  |
| **chroma\_weight\_l1\_flag** | 2 | u(1) |
| if( chroma\_weight\_l1\_flag ) |  |  |
| for( j = 0; j < 2; j++ ) { |  |  |
| **chroma\_weight\_l1[** i **][** j **]** | 2 | se(v) |
| **chroma\_offset\_l1[** i **][** j **]** | 2 | se(v) |
| } |  |  |
| **}** |  |  |
| } |  |  |
| } |  |  |

1. Memory Management Command Operation. [↑](#footnote-ref-2)
2. For simplicity, only the generic (whatever block size) weighted version of the SAD, SSE and HAD functions has been implemented, but the fixed block size version derivation are straightforward. [↑](#footnote-ref-3)