|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, 14-22July, 2011 | Document: JCTVC-F356 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Motion Compensation Complexity Reduction for Bi-Prediction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Hui Yong Kim,  \*Kyung Yong Kim, \*Gwang Hoon Park, Sung-Chang Lim, Jinho Lee,  and JinSoo Choi  ETRI, 138 Gajeong-Ro, Yuseong-Gu, Daejeon, 305-700, Korea  \*Media Lab., Kyung Hee University 1, Seochun, Kiheung, Youngin, Gyonggi, 446-701, Korea | Tel: Email: | +82-42-860-5069 {hykim5, sclim, jinosoul,jschoi} @etri.re.kr  +82-31-201-3680 {\*kimky, \*ghpark} @khu.ac.kr |
| *Source:* | ETRI and Kyung Hee University | | |

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

# Abstract

This contribution reports that roughly 30% and 5% area of forward B-Slices are observed to have identical motion information within each PU, under HM3 LD and RA configuration, respectively. It is indicated that when the L0 and L1 motion information is the same, the L1 interpolation process and the weighted averaging process in HM3 could be bypassed for complexity reduction.

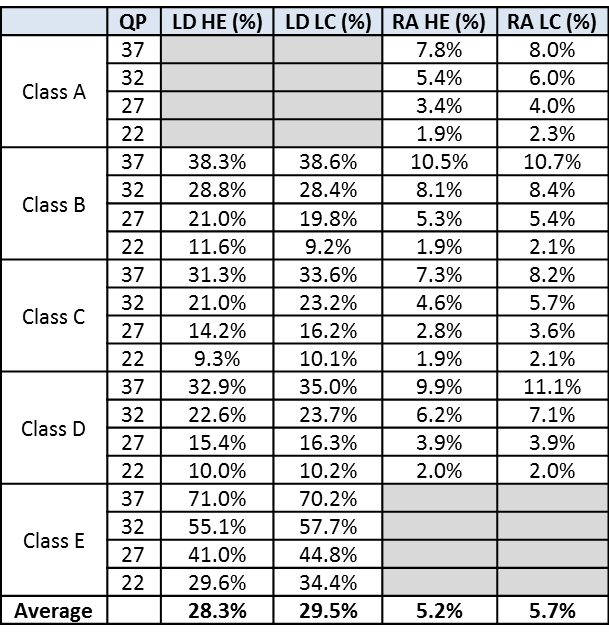
Several slightly different methods are presented for handling the same-motion cases. Method-1 checks the motion-sameness for a Bi-predicted PUs and use the L0 MC process instead of the weighted MC process if the two motion information are identical. Method-2 checks the motion-sameness only for Bi-predicted Merge and SKIP PUs and forces the prediction mode to be as was Pred\_LC if the two motion information are identical. Method-3 uses Method-2 as a basis and adds an encoder-only modification: when the L0 and L1 motion information for a Bi-predicted non-Merge Inter PU are identical, then the encoder is forced to use Pred\_LC mode with the identical motion information.

It is reported that decoding time reduction for Method-1 is roughly 4% and 1% for LD and HE configurations, respectively, without noticeable change in encoding time and coding performance. Reportedly for Method-2, decoding time reduction is roughly 4% and 0% for LD and HE configurations, respectively. Method-3 reportedly gives slightly less encoding time than Method-2.

# Introduction

Table 1 shows the % area covered by the PUs having two motion vectors with identical values and identical reference frame. As could be seen in the table, the % area is very large in LD configuration and even reaches up to over 70% for Class E at QP=37.

**Table 1 - % area of the Bi-predicted PUs satisfying “mvL0==mvL1 && - RefPicOrderCnt(currPic, refIdxL0, L0)==RefPicOrderCnt(currPic, refIdxL1, L1)”**

****

* LD: % Area is computed for all B-Slices
* RA: % Area is computed for the B-Slices of temporal-level 0

An explanation for this is as follows. Assume that current PU is coded in Bi-predictive temporal Merge or SKIP mode and that the collocated PU has L0 motion vector but does not have L1 motion vector. Then the two motion vectors for current PU are identical since the two TMVPs become identical according to HM3 TMVP derivation process. The same motion vector could be propagated spatially by spatial Merge or SKIP mode as illustrated in Figure 1. Thus, in forward B-Slices, where the L1 list is always identical to L0 list, there might be large chance that the two motion information of a Bi-predicted PU are exactly the same.



* Red circle: temporal SKIP mode, Gray circle: spatial SKIP mode, Yellow circle: spatial Merge mode

Figure 1 – Illustration of spatial propagation of the same motion information

When the L0 and L1 motion information of a PU are the same, the L1 interpolation process and the weighted averaging process in HM3 becomes redundant and thus could be bypassed.

This contribution presents several methods of removing such redundancy for complexity reduction.

# Method-1:Modification to the decoding process for inter prediction samples

Method-1 checks the motion-sameness of Bi-predicted PUs and use the L0 MC process instead of the weighted MC process if the two motion information are identical. Figure 2 illustrates the concept of the proposed method.

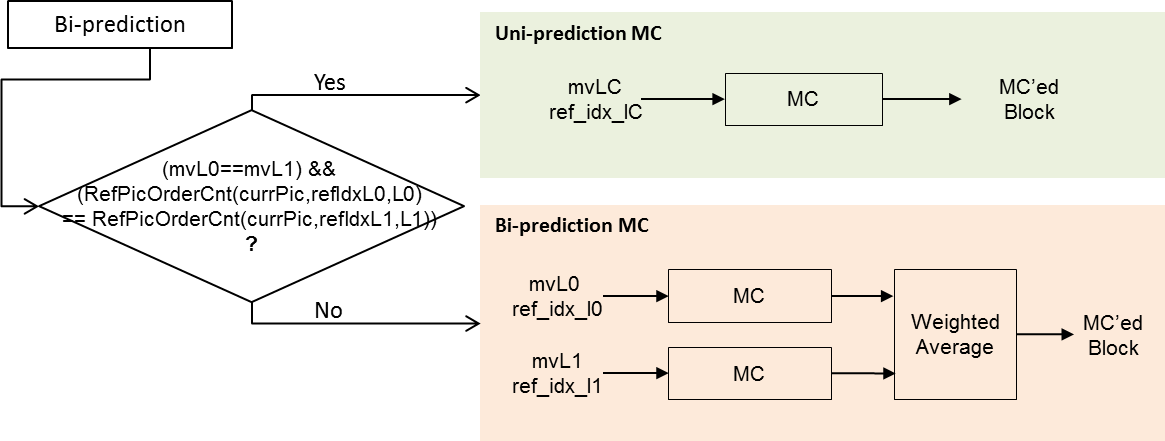


Figure 2 – Illustration of Method-1

Appendix A describes the required modification to WD3 text for Method-1.

Table 2 shows the experimental results for Method-1.

**Table 2 - Method-1 results (anchor: HM3.0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Access HE | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.1 |
| Class B | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 |
| Class D | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 |
| Class E |  |  |  |  |  |  |
| **Overall** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| Enc Time[%] | 101% | | | 100% | | |
| Dec Time[%] | 98% | | | 99% | | |
|  |  |  |  |  |  |  |
|  | Low delay B HE | | | Low delay B LC | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 | -0.1 |
| Class C | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | -0.2 |
| Class D | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | -0.2 |
| Class E | 0.1 | -0.2 | -0.7 | -0.1 | 0.2 | -0.5 |
| **Overall** | **0.0** | **0.1** | **-0.1** | **0.0** | **0.0** | **-0.2** |
| Enc Time[%] | 99% | | | 99% | | |
| Dec Time[%] | 97% | | | 96% | | |

# Method-2: Modification to the TMV derivation process for merge mode

Method-2 checks the motion-sameness only for Bi-predicted Merge and SKIP PUs and forces the prediction mode to be as was Pred\_LC if the two motion information are identical.

Appendix B describes the required modification to WD3 text for Method-2.

Table 3 shows the experimental results for Method-2.

**Table 3 - Method-2 results (anchor: HM3.0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Access HE | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.0 | -0.1 | 0.1 | 0.0 | 0.0 | 0.1 |
| Class B | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.1 | 0.0 | 0.0 | -0.1 | -0.1 |
| Class D | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.1 |
| Class E |  |  |  |  |  |  |
| **Overall** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 100% | | | 99% | | |
|  |  |  |  |  |  |  |
|  | Low delay B HE | | | Low delay B LC | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | -0.1 | 0.2 | 0.0 | 0.0 | -0.1 | -0.3 |
| Class C | 0.0 | 0.1 | 0.0 | 0.0 | -0.1 | -0.1 |
| Class D | 0.0 | 0.1 | 0.3 | 0.0 | -0.3 | -0.3 |
| Class E | 0.0 | 0.3 | -0.2 | 0.0 | 0.0 | -0.6 |
| **Overall** | **0.0** | **0.2** | **0.0** | **0.0** | **-0.1** | **-0.3** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 96% | | | 96% | | |

# Method-3: Method-2 with encoder-only modification

Method-3 uses Method-2 as a basis and adds an encoder-only modification: when the L0 and L1 motion information for a Bi-predicted non-Merge Inter PU are identical, then the encoder is forced to use Pred\_LC mode with the identical motion information.

Appendix C describes the required modification to the HM encoder for Method-3.

Table 4 shows the experimental results for Method-3.

**Table4 - Method-3 results (anchor: HM3.0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Access HE | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 | 0.2 |
| Class B | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | -0.1 |
| Class D | 0.0 | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class E |  |  |  |  |  |  |
| **Overall** | **0.0** | **-0.1** | **0.0** | **0.0** | **0.0** | **0.0** |
| Enc Time[%] | 100% | | | 101% | | |
| Dec Time[%] | 99% | | | 99% | | |
|  |  |  |  |  |  |  |
|  | Low delay B HE | | | Low delay B LC | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | -0.1 | -0.1 | -0.1 | 0.0 | -0.1 | -0.3 |
| Class C | -0.1 | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class D | -0.1 | 0.1 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class E | 0.0 | -0.4 | -0.5 | 0.1 | -0.1 | -0.3 |
| **Overall** | **0.0** | **-0.1** | **-0.2** | **0.0** | **0.0** | **-0.2** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 96% | | | 96% | | |

# Conclusions

This contribution presents several methods for reducing motion compensation complexity in Bi-prediction. Experimental results show that roughly 4% decoding time reduction can be achieved by only slight modification to the current HM, with no harm to both encoding time and coding performance. It is recommended that the propose method be integrated into the next version of HM.

# Reference

1. F.Bossen, "Common test conditions and software reference configurations", JCTVC-E700, March 2011, Geneva, CH

# Patent rights declaration(s)

**ETRI and Kyung Hee University 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).**

# Appendix A: WD3(draft8)modification for Method-1

***Insert the yellow marked text into section 8.4.2.2.***

##### 8.4.2.2 Decoding process for inter prediction samples

Inputs to this process are:

– a luma location ( xC, yC ) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

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

– a variable nCS specifying the size of the current coding unit,

– variables specifying the width and the height of the prediction unit, nPSW and nPSH,

– luma motion vectors mvL0 and mvL1, and chroma motion vectors mvCL0 and mvCL1,

– reference indices refIdxL0 and refIdxL1,

– prediction list utilization flags, predFlagL0 and predFlagL1.

Outputs of this process are:

– a (nCSL)x(nCSL) array predSamplesL of luma prediction samples, where nCSL is derived as specified below,

– a (nCSC)x(nCSC) array preSamplesCb of chroma prediction samples for the component Cb, where nCSC is derived as specified below,

– a (nCSC)x(nCSC) array predSamplesCr of chroma residual samples for the component Cr, where nCSC is derived as specified below.

The variable nCSL is set equal to nCS and the variable nCSC is set equal to nCS >> 1. [Ed: (WJ) revisit for supporting other chroma formats]

Let predSamplesL0L and predSamplesL1L be (nPSW)x(nPSH) arrays of predicted luma sample values and predSampleL0Cb, predSampleL1Cb, predSampleL0Cr, and predSampleL1Cr be (nPSW/2)x(nPSH/2) arrays of predicted chroma sample values.

For LX being replaced by either L0 or L1 in the variables predFlagLX, RefPicListX, refIdxLX, refPicLX, and predPartLX, the following is specified.

If mvL0 is equal to mvL1 and RefPicOrderCnt( currPic, refIdxL0, L0 ) is equal to RefPicOrderCnt( currPic, refIdxL1, L1 ), the following applies.

* predFlagL1 = 0

When predFlagLX is equal to 1, the following applies.

– The reference picture consisting of an ordered two-dimensional array refPicLXL of luma samples and two ordered two-dimensional arrays refPicLXCb and refPicLXCr of chroma samples is derived by invoking the process specified in subclause8.4.2.2.1 with refIdxLX and RefPicListX given as input.

– The arrays predSamplesLXL, predSamplesLXCb, and predSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause8.4.2.2.2 with the luma locations ( xC, yC ), ( xB, yB ), the width an the height of the current prediction unit nPSW, nPSH, the motion vectors mvLX, mvCLX, and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr given as input.

The array predSampleL of the prediction samples of luma component is derived by invoking the weighted sample prediction process specified in subclause8.4.2.2.3 with the luma location ( xB, yB ), the width an the height of the current prediction unit nPSW, nPSH, and the sample arrays predSamplesL0L and predSamplesL1L as well as predFlagL0, predFlagL1 and BitDepthY given as input.

For C being replaced by Cb, or Cr, the array predSampleC of the prediction samples of component C is derived by invoking the weighted sample prediction process specified in subclause8.4.2.2.3 with the chroma location ( xB/2, yB/2 ), the width an the height of the current prediction unit nPSWC set equal to nPSW/2, nPSHC set equal to nPSH/2, and the sample arrays predSamplesL0C and predSamplesL1C as well as predFlagL0, predFlagL1 and BitDepthC given as input.

# Appendix B: WD3(draft8) modification for Method-2

***Insert the yellow marked text into section 8.4.2.1.1.***

##### 8.4.2.1.1Derivation process for luma motion vectors for merge mode

This process is only invoked when PredMode is equal to MODE\_ INTER and merge\_flag[ xP ][ yP ] is equal to 1, where ( xP, yP ) specify the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture.

Inputs of this process are

* a luma location ( xP, yP ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

– theluma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

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

1. The derivation process for merging candidates from neighboring prediction unit partitions in subclause 8.4.2.1.4 is invoked with luma location ( xP, yP ), the width and the height of the prediction unit nPSW and nPSH and the partition index PartIdx as inputs and the output is assigned to the availability flags availableFlagN, the 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 A, B, C or D.
2. The derivation process of reference indices for temporal merging candidate in subclause 8.4.2.1.3 is invoked with luma location ( xP, yP ), nPSW, nPSH as the inputs and the output is directly assigned to refIdxLX.
3. The derivation process for temporal luma motion vector prediction in subclause 8.4.2.1.7 is invoked with luma location ( xP, yP ), refIdxLX as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector mvLXCol.  
     
   If mvL0Col is equal to mvL1Col and RefPicOrderCnt( currPic, refIdxL0, L0 ) is equal to RefPicOrderCnt( currPic, refIdxL1, L1 ), the following applies.  
     
   availableFlagL1Col = 0  
     
   The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.

availableFlagCol = availableFlagL0Col || availableFlagL1Col

predFlagLXCol = availableFlagLXCol

1. The merging candidate list, mergeCandList, is constructed of which elements are given as specified order:
   1. A, if availableFlagA is equal to 1
   2. B, if availableFlagB is equal to 1
   3. Col, if availableFlagCol is equal to 1
   4. C, if availableFlagC is equal to 1
   5. D, if availableFlagD is equal to 1
2. If several merging candidates have the motion vectors and the same reference indices, the merging candidates are removed from the list except the merging candidate which has the smallest order in the mergeCandList.
3. If the number of elements NumMergeCand within the mergeCandList is equal to 1, mergeIdx is set equal to 0, otherwise, mergeIdx is set equal to merge\_idx[ xP][ yP ].
4. The following assignments are made with N being the candidate at position mergeIdx in the merging candidate list mergeCandList ( N = mergeCandList[ mergeIdx ] ) and X being replaced by 0 or 1:

mvLX[ 0 ] = mvLXN[ 0 ]

mvLX[ 1 ] = mvLXN[ 1 ]

refIdxLX = refIdxLXN

predFlagLX = predFlagLXN

1. If all availability flags availableFlagN (with N being replaced by A, B, Col, C, or D) are equal to 0, mergeIdx is set equal to 0 and the variables mvLX, refIdxLX and predFlagLX (with X being replaced by 0 or 1) are inferred as follows.

If slice\_type is equal to P, the following applies.

mvLX[ 0 ] = 0

mvLX[ 1 ] = 0

refIdxL0 = 0

refIdxL1 = -1

predFlagL0 = 1

prefFlagL1 = 0

Otherwise ( slice\_type is equal to B ), the following applies.

If RefPicOrderCnt(currPic, 0, L0 ) is equal to RefPicOrderCnt( currPic, 0, L1 ), the following applies. mvLX[ 0 ] = 0

mvLX[ 1 ] = 0

refIdxL0 = 0

refIdxL1 = -1

predFlagL0 = 1

prefFlagL1 = 0

Otherwise, the following applies.

mvLX[ 0 ] = 0

mvLX[ 1 ] = 0

refIdxL0 = 0

refIdxL1 = 0

predFlagL0 = 1

prefFlagL1 = 1

# Appendix C: HM Encoder modification for Method-3

***In the inter mode decision process, if a mode satisfies all the following conditions,***

* PredMode == MODE\_INTER
* merge\_flag == 0
* inter\_pred\_flag == Pred\_BI
* mvL0==mvL1
* RefPicOrderCnt(currPic, refIdxL0, L0)==RefPicOrderCnt(currPic, refIdxL1, L1)

***thenperformthe following assignments,***

* inter\_pred\_flag = Pred\_LC
* mvLC = mvL0
* refIdxLC = refIdxL0.

***.***