|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, SE, 11 – 20 July 2012 | Document: JCTVC-J0121r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG10: Motion related hooks for HEVC multiview/3DV extension based on long-term reference pictures** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Ying Chen Ye-Kui Wang Li Zhang Vadim Seregin  5775 Morehouse Dr San Diego, CA 92121 USA | Tel: Email: | 1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com)  1-858-651-6660 [lizhang@qualcomm.com](mailto:lizhang@qualcomm.com)  1-858-651-3481 vseregin@qualcomm.com |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

In the current HEVC design, similarly as in AVC, scaling of motion vectors involving long-term reference pictures, in both the AMVP and merge modes, is disabled. A reference picture from a different view in the potential multiview/3DV extension of HEVC may be indicated as a long-term reference picture. In this document, it is proposed that, in AMVP, prediction of an MV belonging to a long-term reference picture from an MV belonging to a short-term reference picture or vice versa is disabled, so that in the potential multiview/3DV extension, inter-view motion vectors and short-term motion vectors do not predict from each other.

# Introduction

## Multiview video coding

In MPEG HTM (HEVC based 3DV Test Model), one view can be predicted from the other by the so-called disparity motion compensation. It is realized in a way that a decoded view component from a different view but in the same time instance can be added in a reference picture list of the current view component. For example, as shown in Figure 1, wherein the vertical indices (V1, V0 and V2) of the pictures correspond to view identifiers and the horizontal indices (T0 through T11) correspond to POC values. Each square is a view component and when a vertical arrow links view component A (e.g., T5/V0) to another view component B (e.g., T5/V1, in the same time instance), view component A can be added into the reference picture list 0 of view component B.

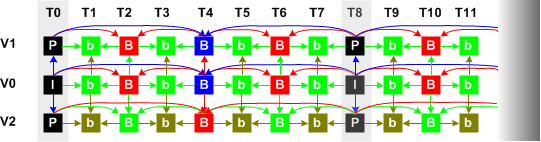


Figure 1: Typical motion prediction structure in 3DV (texture only).

Note that in the above example, both view component A and view component B have the same POC value (5).

A motion vector pointing to a view component of a different view is called disparity motion vector in this document and this view component is referred to as an inter-view reference picture.

## Motion vector prediction in HTM

In HTM, the MPEG 3DV reference software for 3D-HEVC, when a temporal motion vector is to be predicted using AMVP, only the spatially/temporally neighboring vectors that are also temporal motion vectors are considered as available. This means a motion vector corresponding to inter-view reference is not used to predict a temporal motion vector. Similarly, a temporal motion vector is not used to predict a motion vector corresponding to inter-view reference.

# Proposal

To follow the HTM design (expected to have a higher coding efficiency) which disables prediction between inter-view motion vectors and temporal motion vectors and to reuse the long-term property to represent an inter-view reference picture in a potential “HLS-only” HEVC profile for MVC, 3DV or even SVC, it is proposed to consider motion vectors belonging to a different category as unavailable during AMVP, including TMVP. Note that in the current specification, the only two categories are short-term and long-term.

## Changes in AMVP

The changes in this sub-section are only related to the first aspect.

8.5.2.1.6 Derivation process for motion vector predictor candidates

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are (with N being replaced by A, or B)

* the motion vectors mvLXN of the neighbouring prediction units,
* the availability flags availableFlagLXN of the neighbouring prediction units.



Figure 8‑3 – Spatial motion vector neighbours

The variable isScaledFlagLX with X being 0 or 1 is set equal to 0.

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. Let a set of two sample locations be (xAk, yAk), with k = 0, 1, specifies sample locations with xAk = xP − 1, yA0 = yP + nPSH and yA1 = yA0 - MinPuSize. The set of sample locations ( xAk, yAk ) represent the sample locations immediately to the left side of the left partition boundary and it’s extended line.
2. Let the availability flag availableFlagLXA be initially set equal to 0 and the both components of mvLXA are set equal to 0.
3. When one or more of the following conditions are true, the variable isScaledFlagLX is set equal to 1.

* the prediction unit covering luma location ( xA0, yA0 ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] and PredMode is not MODE\_INTRA.
* the prediction unit covering luma location ( xA1, yA1 ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] and PredMode is not MODE\_INTRA.

1. For ( xAk, yAk ) from ( xA0, yA0 ) to ( xA1, yA1 ) where yA1 = yA0 − MinPuSize, the following applies repeatedly until availableFlagLXA is equal to 1:

* If the prediction unit covering luma location ( xAk,yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xAk ][ yAk ] is equal to 1 and the reference index refIdxLX[ xAk ][ yAk ] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXA is set equal to 1 and the motion vector mvLXA is set equal to the motion vector mvLX[ xAk ][ yAk ], refIdxA is set equal to refIdxLX[ xAk ][ yAk ] and ListA is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xAk ][ yAk ] (with Y = !X) is equal to 1 and PicOrderCnt( RefPicListY[ refIdxLY[ xAk ][ yAk ] ] ) is equal to PicOrderCnt( RefPicListX[ refIdxLX ] ), availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[ xAk ][ yAk ], refIdxA is set equal to refIdxLY[ xAk ][ yAk ] ,  ListA is set equal to ListY and mvLXA is set equal to mvLXA.

1. When availableFlagLXA is equal to 0, for ( xAk, yAk ) from ( xA0, yA0 ) to ( xA1, yA1 ) where yA1 = yA0 - MinPuSize, the following applies repeatedly until availableFlagLXA is equal to 1:

* If the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xAk ][ yAk ] is equal to 1, and RefPicListX[ refIdxLX ] and RefPicListX[ refIdxLX[ xAk ][ yAk ] ] are both long-term reference pictures or are both short-term reference pictures, availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLX[ xAk ][ yAk ], refIdxA is set equal to refIdxLX[ xAk ][ yAk ], ListA is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xAk ][ yAk ] (with Y = !X) is equal to 1, and RefPicListX[ refIdxLX ] and RefPicListY[ refIdxLY[ xAk ][ yAk ] ] are both long-term reference pictures or are both short-term reference pictures, availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[ xAk ][ yAk ], refIdxA is set equal to refIdxLY[ xAk ][ yAk ], ListA is set equal to ListY.
* When availableFlagLXA is equal to 1, and both RefPicListA[ refIdxA ] and RefPicListX[ refIdxLX ] are short-term reference pictures, mvLXA is derived as specified below.

tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td (8‑126)

DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑127)

mvLXA = Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvLXA ) \*    
 ( (Abs( DistScaleFactor \* mvLXA ) + 127 ) >> 8 ) ) (8‑128)

where td and tb are derived as

td = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListA[ refIdxA ] ) ) (8‑129)

tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ) ) (8‑130)

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. Let a set of three sample location (xBk, yBk), with k = 0,1,2, specifies sample locations with xB0 = xP + nPSW, xB1 = xB0− MinPuSize , xB2 = xP − MinPuSize and yBk = yP − 1. The set of sample locations ( xBk, yBk ) represent the sample locations immediately to the upper side of the above partition boundary and its extended line. [Ed. (BB): Define MinPuSize in the SPS but the derivation should depend on the use of an AMP flag ]
2. When yP−1 is less than (( yC >> Log2CtbSize ) << Log2CtbSize), the following applies.

xB0 = (xB0>>3)<<3) + ((xB0>>3)&1)\*7 (8‑131)  
xB1 = (xB1>>3)<<3) + ((xB1>>3)&1)\*7 (8‑132)  
xB2 = (xB2>>3)<<3) + ((xB2>>3)&1)\*7 (8‑133)

1. Let the availability flag availableFlagLXB be initially set equal to 0 and the both components of mvLXB are set equal to 0.
2. For ( xBk, yBk ) from ( xB0, yB0 ) to ( xB2, yB2 ) where xB0 = xP + nPSW, xB1 = xB0 − MinPuSize , and xB2 =  xP − MinPuSize, the following applies repeatedly until availableFlagLXB is equal to 1:

* If the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xBk ][ yBk ] is equal to 1, and the reference index refIdxLX[ xBk ][ yBk ] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXB is set equal to 1 and the motion vector mvLXB is set equal to the motion vector mvLX[ xBk ][ yBk ], refIdxB is set equal to refIdxLX[ xBk ][ yBk ] and ListB is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xBk ][ yBk ] (with Y = !X) is equal to 1, and PicOrderCnt( RefPicListY[ refIdxLY[ xBk ][ yBk ] ] ) is equal to PicOrderCnt( RefPicListX[ refIdxLX ] ), availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[ xBk ][ yBk ], refIdxB is set equal to refIdxLY[ xBk ][ yBk ],  and ListB is set equal to ListY.

1. When isScaledFlagLX is equal to 0 and availableFlagLXB is equal to 1, mvLXA is set equal to mvLXB and refIdxA is set equal to refIdxB and availableFlagLXA is set equal to 1.
2. When isScaledFlagLX is equal to 0, availableFlagLXB is set equal to 0 and for ( xBk, yBk ) from ( xB0, yB0 ) to ( xB2, yB2 ) where xB0 = xP +nPSW, xB1 = xB0 - MinPuSize , and xB2 =  xP - MinPuSize, the following applies repeatedly until availableFlagLXB is equal to 1:

* If the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xBk ][ yBk ] is equal to 1, and RefPicListX[ refIdxLX ] and RefPicListX[ refIdxLX[ xBk ][ yBk ] ] are both long-term reference pictures or are both short-term reference pictures, availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLX[ xBk ][ yBk ], refIdxB is set equal to refIdxLX[ xBk ][ yBk ], ListB is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xBk ][ yBk ] (with Y = !X) is equal to 1, and RefPicListX[ refIdxLX ] and RefPicListY[ refIdxLY[ xBk ][ yBk ] ] are both long-term reference pictures or are both short-term reference pictures, availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[ xBk ][ yBk ], refIdxB is set equal to refIdxLY[ xBk ][ yBk ], ListB is set equal to ListY.
* When availableFlagLXB is equal to 1 and PicOrderCnt( RefPicListB[ refIdxB ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ) and both RefPicListB[ refIdxB ] and RefPicListX[ refIdxLX ] are short-term reference pictures, mvLXB is derived as specified below.

tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td (8‑134)

DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑135)

mvLXB =Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvLXA ) \*   
 ( (Abs( DistScaleFactor \* mvLXA ) + 127 ) >> 8 ) ) (8‑136)

[Ed. (GJS): I believe the thing that is being clipped is an integer, so does that make sense?]

where td and tb are derived as

td = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListB[ refIdxB ] ) ) (8‑137)

tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ) ) (8‑138)

## Changes in TMVP

This sub-section includes changes for both aspects of this proposal, highlighted in different colors.

8.5.2.1.7 Derivation process for temporal luma motion vector prediction

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are

* the motion vector prediction mvLXCol,
* the availability flag availableFlagLXCol.

The function RefPicOrderCnt( picX, refIdx, LX ) returns the picture order count PicOrderCntVal of the reference picture with index refIdx from reference picture list LX of the picture picX and is specified as follows.

RefPicOrderCnt( picX, refIdx, LX ) = PicOrderCnt(RefPicListX[ refIdx ] of the picture picX) (8 141)

Depending on the values of slice\_type, collocated\_from\_l0\_flag, and collocated\_ref\_idx, the variable colPic, specifying the picture that contains the collocated partition, is derived as follows.

* If slice\_type is equal to B and collocated\_from\_l0\_flag is equal to 0, the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList1[ collocated\_ref\_idx ].
* Otherwise (slice\_type is equal to B and collocated\_from\_l0\_flag is equal to 1 or slice\_type is equal to P) , the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList0[ collocated\_ref\_idx ].

Variable colPu and its position ( xPCol, yPCol ) are derived in the following ordered steps:

1. The variable colPu is derived as follows

yPRb = yP + nPSH (8‑139)

* + If ( yP >> Log2CtbSize ) is equal to ( yPRb >> Log2CtbSize ), the horizontal component of the right-bottom luma position of the current prediction unit is defined by

xPRb = xP + nPSW (8‑140)

and the variable colPu is set as the prediction unit covering the modified position given by ( ( xPRb >> 4 ) << 4, ( yPRb >> 4 ) << 4 ) inside the colPic.

* + Otherwise ( ( yP >> Log2CtbSize ) is not equal to ( yPRb >> Log2CtbSize ) ), colPu is marked as "unavailable".

1. When colPu is coded in an intra prediction mode or colPu is marked as "unavailable", the following applies.
   * Central luma position of the current prediction unit is defined by

xPCtr = ( xP + ( nPSW >> 1 ) (8‑141)

yPCtr = ( yP + ( nPSH >> 1 ) (8‑142)

* + The variable colPu is set as the prediction unit covering the modified position given by ( ( xPCtr >> 4 ) << 4, ( yPCtr >> 4 ) << 4 ) inside the colPic.

1. ( xPCol, yPCol ) is set equal to the top-left luma sample of the colPu relative to the top-left luma sample of the colPic.

The function LongTermRefPic( picX, refIdx, LX ) is defined as follows. If the reference picture with index refIdx from reference picture list LX of the picture picX was marked as "used for long term reference" at the time when picX was the current picture, LongTermRefPic( picX, refIdx, LX ) returns 1; otherwise LongTermRefPic( picX, refIdx, LX ) returns 0.

The variables mvLXCol and availableFlagLXCol are derived as follows.

* If one or more of the following conditions are true, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
  + colPu is coded in an intra prediction mode.
  + colPu is marked as "unavailable".
  + pic\_temporal\_mvp\_enable\_flag is equal to 0.
* Otherwise, the motion vector mvCol, the reference index refIdxCol, and the reference list identifier listCol are derived as follows.
  + If PredFlagL0[ xPCol ][ yPCol ] is equal to 0, mvCol, refIdxCol, and listCol are set equal to MvL1[ xPCol ][ yPCol ], RefIdxL1[ xPCol ][ yPCol ], and L1, respectively.
  + Otherwise (PredFlagL0[ xPCol ][ yPCol ] is equal to 1), the following applies.
  + If PredFlagL1[ xPCol ][ yPCol ] is equal to 0, mvCol, refIdxCol, and listCol are set equal to MvL0[ xPCol ][ yPCol ], RefIdxL0[ xPCol ][ yPCol ], and L0, respectively.
  + Otherwise (PredFlagL1[ xPCol ][ yPCol ] is equal to 1), the following assignments are made.
    - * If PicOrderCnt( pic ) of every picture pic in every reference picture lists is less than or equal to PicOrderCntVal, mvCol, refIdxCol, and listCol are set equal to MvLX[ xPCol ][ yPCol ], RefIdxLX[ xPCol ][ yPCol ] and LX, respectively with X being the value of X this process is invoked for.
      * Otherwise (PicOrderCnt( pic ) of at least one picture pic in at least one reference picture list is greater than PicOrderCntVal, mvCol, refIdxCol and listCol are set equal to MvLN[ xPCol ][ yPCol ], RefIdxLN[ xPCol ][ yPCol ] and LN, respectively with N being the value of collocated\_from\_l0\_flag.
  + If one of the following conditions is true, the variable availableFlagLXCol is set equal to 0:
  + RefPicListX[ refIdxLX ] is a long-term reference picture and LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 0;
  + RefPicListX[ refIdxLX ] is a short-term reference picture and LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 1;
  + Otherwise, the variable availableFlagLXCol is set equal to 1, and the following applies.
  + If RefPicListX[ refIdxLX ] is a long-term reference picture, or LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 1, or PicOrderCnt( colPic ) – RefPicOrderCnt( colPic, refIdxCol, listCol ) is equal to PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ),  
     mvLXCol = mvCol (8‑143)
  + Otherwise, mvLXCol is derived as scaled version of the motion vector mvCol as specified below  
     tx = ( 16384 + ( Abs( td ) >>1 ) ) / td (8‑144)
  + DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑145)

mvLXCol =  Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvCol ) \*   
 ( (Abs( DistScaleFactor \* mvCol ) + 127 ) >> 8 ) ) (8‑146)

where td and tb are derived as  
td = Clip3( −128, 127, PicOrderCnt( colPic ) – RefPicOrderCnt( colPic, refIdxCol, listCol ) ) (8‑147)  
tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX [ refIdxLX ] ) ) (8‑148)

# Simulation Results

This section provides simulation results of the proposed method in comparison with the 3DV-HTM with modifications to make it consistent with current HEVC design and the inter-view reference pictures are treated as long-term pictures. During the simulation tests, the latest software HTM3.1 (based on HM6) and the common test conditions for coding texture views in 3DV-HTM are utilized except that low-level coding tools are disabled, i.e., inter-view motion prediction and inter-view residual prediction.

The coding gain of the proposed method compared to the anchor for texture views coding in 3-veiw case is reported in this subsection. Table 1 shows the average coding gain of the proposed method with respect to the anchor. The column denoted by ‘Video i’( i = 0, 1 or 2, and is the view index) lists the coding gain of the i-th texture views in the coding order. Here, the bitrate and PSNR values represent the bitrate and PSNR vaules of i-th texture views, respectively. The column denoted by ‘Video only’ lists the coding gain of all three texture views where the bitrates represent the total bitrates of bitstreams containing three texture views and the PSNR values are the average PSNR values of the three decoded texture views.

As shown in Table 1, the overall average bitrate saving is around 0.8% for all decoded texture views and 2.2% for dependent views.

Table 1: Coding gain with respect to anchor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | video 0 | video 1 | video 2 | video only |
| Balloons | 0.0% | -2.4% | -2.0% | -1.0% |
| Kendo | 0.0% | -2.1% | -1.6% | -0.9% |
| Newspapercc | 0.0% | -1.4% | -1.4% | -0.6% |
| GhostTownFly | 0.0% | -5.2% | -4.9% | -1.5% |
| PoznanHall2 | 0.0% | -0.6% | -1.6% | -0.5% |
| PoznanStreet | 0.0% | -1.6% | -1.6% | -0.6% |
| UndoDancer | 0.0% | -2.2% | -2.1% | -0.8% |
| 1024x768 | 0.0% | -2.0% | -1.7% | -0.8% |
| 1920x1088 | 0.0% | -2.4% | -2.6% | -0.9% |
| average | 0.0% | -2.2% | -2.2% | -0.8% |

# Patent rights declaration(s)

**Qualcomm Incorporated 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).**