|  |  |
| --- | --- |
| **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-C0148-Proposed\_Text |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Proposed text for JCT3V-C0148 based on 3D-HEVC Test Model 2** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Christine Guillemot  Laurent Guillo | Tel: Email: | +33299847100 Christine.guillemot@inria.fr  Laurent.guillo@inria.fr |
| *Source:* | INRIA, France. | | |

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

# Abstract

This document provides text for Draft 2 of 3D-HEVC Test Model Description. The proposal only changes the motion vector candidate list derivation process as specified in sub-clause G.8.5.2.1.

All the changes are highlighted in green.

G.8.5.2.1.1 Derivation process for luma motion vectors for merge mode

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

Inputs of this process are

* a luma location ( xC, yC ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xP, yP ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* variables specifying the width and the height of the luma prediction block, nPbW and nPbH,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

* the luma motion vectors mvL0 and mvL1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0 and predFlagL1,
* the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1,
* the disparity vectors ivpMvDispL0 and ivpMvDispL1.

The variables singleMCLFlag is derived as follows.

* + If log2\_parallel\_merge\_level\_minus2 is greater than 0 and nCS is equal to 8, singleMCLFlag is set to 1.
  + Otherwise, singleMCLFlag is set to 0.

When singleMCLFlag is equal to 1, xP is set equal to xC, yP is set equal to yC, and both nPbW and nPbH are set equal to nCS.

NOTE – When singleMCLFlag is equal to 1, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, ~~and~~ the prediction utilization flags predFlagL0 and predFlagL1, the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1, and the disparity vectors ivpMvDispL0 and ivpMvDispL1 are derived as specified by the following ordered steps:

* 1. The derivation process for spatial merge ~~ing~~ candidates ~~from neighboring prediction unit partitions~~ as specified in subclause G.8.5.2.1.2 is invoked with the luma coding block location ( xC, yC ), the coding block size nCS, the luma prediction block location ( xP, yP ), the variable singleMCLFlag, the width and the height of the luma prediction block nPbW and nPbH and the partition index partIdx as inputs and the output is assigned to the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N and the motion vectors mvL0N and mvL1N with N being replaced by A0, A1, B0, B1 or B2.
  2. The reference index for temporal merging candidate refIdxLX (with X being 0 or 1) is set equal to 0.
  3. The derivation process for temporal luma motion vector prediction in subclause G.8.5.2.1.7 is invoked with luma location ( xP, yP ), the width and the height of the luma prediction block nPbW and nPbH, and refIdxLX as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector mvLXCol. The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.
     + 1. availableFlagCol = availableFlagL0Col | | availableFlagL1Col (G‑87)   
          predFlagLXCol = availableFlagLXCol (G‑88)
  4. Depending on multi\_view\_mv\_pred\_flag, the following applies.
     + If multi\_view\_mv\_pred\_flag is equal to 0, the flags availableFlagIvMC and availableFlagIvDC are set equal to 0.
     + Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause G.8.5.2.1.9 is invoked with the luma locations ( xC, yC ) and ( xP, yP ), the coding block size nCS, the variables nPSW and nPSH, and the partition index partIdx, as the inputs and the output is assigned to the view order index of the reference view refViewIdx, the availability flags availableFlagIvMC and availableFlagIvDC, the reference indices refIdxLXIvMC and refIdxLXIvDC, the prediction list utilization flags predFlagLXIvMC and predFlagLXIvDC, and the motion vectors mvLXIvMC and mvLXIvDC (with X being 0 or 1, respectively), and the disparity vector mvDisp.
  5. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps.
  6. The variable numMergeCand is set equal to 0.
  7. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC and the variable numMergeCand is increased by 1.
  8. When availableFlagA1 is equal to 1 and one or more of the following conditions is true,
     + - * availableFlagIvMC  = = 0,
         * predFlagLXIvMC ! = predFlagLXA1, (with X being replaced by 0 and 1),
         * mvLXIvMC ! = mvLXA1 (with X being replaced by 0 and 1),
         * refIdxLXIvMC ! = refIdxLXA1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.

* 1. When availableFlagB1 is equal to 1, and one or more of the following conditions is true ,
     + - * availableFlagIvMC = = 0,
         * predFlagLXIvMC ! = predFlagLXB1, (with X being replaced by 0 and 1),
         * mvLXIvMC ! = mvLXB1 (with X being replaced by 0 and 1),
         * refIdxLXIvMC ! = refIdxLXB1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.

* 1. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
  2. When availableFlagIvDC is equal to 1, and one or more of the following conditions is true,
     + - * availableFlagA1 = = 0,
         * predFlagLXA1 ! = predFlagLXIvDC, (with X being replaced by 0 and 1),
         * mvLXA1 ! = mvLXIvDC(with X being replaced by 0 and 1),
         * refIdxLXA1 ! = refIdxLXIvDC(with X being replaced by 0 and 1),

and one or more of the following conditions is true,

* + - * + availableFlagB1 = = 0,
        + predFlagLXB1 ! = predFlagLXIvDC, (with X being replaced by 0 and 1),
        + mvLXB1 ! = mvLXIvDC(with X being replaced by 0 and 1),
        + refIdxLXB1 ! = refIdxLXIvDC(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvDC and the variable numMergeCand is increased by 1.

* 1. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
  2. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + multi\_view\_mv\_pred\_flag, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
  3. When availableFlagCol is equal to 1 and numMergeCand is less than 5 + multi\_view\_mv\_pred\_flag, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.
  4. Select the first candidate among the first three candidates in the mergeCandList which is a disparity compensated predictor . A candidate, with refCandIdx as index in the mergeCandList, is a disparity compensated predictor is the following conditions are true (with X being replaced by 0 and 1):
     + - * (curRefIdxLX = pcMVFieldNeighbours[refCandIdx<<1 + (X==0?0:1)].getRefIDx()) != -1
         * (refViewIdLX != getslice()->getRefPic(REF\_PIC\_LIST\_X, curRefIdxLX)-> getViewId()) != -1
         * refViewIdLX != curViewId (with curViewId > 0)

if such a candidate does not exist the initial mergeCandList is kept unchanged.

Otherwise, two new candidates will be added at the index lnum and lnum+1:

* + 1. First, tempNumMergeCand is set equal to min(MRG\_MAX\_NUM\_CANDS\_SIGNALED +extraMergeCand, numMergeCand+2)
    2. when refCandIdx is lower than tempNumMergeCand-2, the entry mergeCandList[tempNumMergeCand-3] is first moved to mergeCandList[tempNumMergeCand-1] and lnum is set equal to tempNumMergeCand-3 otherwise it is set equal to tempNumMergeCand-2.
    3. The entry mergeCandList[lnum] is set equal to mergeCandList[refCandIdx] with the horizontal motion vector (belonging to L0 or L1) subtracted by 4 or added by 4 depending on whether the current view is 1 or 2 respectively.
    4. The entry mergeCandList[lnum+1] is set equal to mergeCandList[refCandIdx] with the horizontal motion vector (belonging to L0 or L1) added by 4 or subtracted by 4 depending on whether the current view is 1 or 2 respectively.
    5. numMergeCand is set equal to tempNumMergeCand.

1. ~~A~~~~1~~~~, if availableFlagA~~~~1~~ ~~is equal to 1~~
2. ~~B~~~~1~~~~, if availableFlagB~~~~1~~ ~~is equal to 1~~
3. ~~B~~~~0~~~~, if availableFlagB~~~~0~~ ~~is equal to 1~~
4. ~~A~~~~0~~~~, if availableFlagA~~~~0~~ ~~is equal to 1~~
5. ~~B~~~~2~~~~, if availableFlagB~~~~2~~ ~~is equal to 1~~
6. ~~Col, if availableFlagCol is equal to 1~~
   1. The variable ~~numMergeCand and~~ numOrigMergeCand ~~are~~ is set equal to numMergeCand ~~to the number of merging candidates in the mergeCandList~~.
   2. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause 8.5.2.1.3 is invoked with mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList, numMergeCand and numOrigMergeCand given as input and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0combCandk and refIdxL1combCandk, the prediction list utilization flags predFlagL0combCandk and predFlagL1combCandk and the motion vectors mvL0combCandk and mvL1combCandk of every new candidate combCandk being added in mergeCandList. The number of candidates being added numCombMergeCand is set equal to ( numMergeCand – numOrigMergeCand ). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive.
   3. The derivation process for zero motion vector merging candidates specified in subclause 8.5.2.1.4 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList and the NumMergeCand as the inputs and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0zeroCandm and refIdxL1zeroCandm, the prediction list utilization flags predFlagL0zeroCandm and predFlagL1zeroCandm, the motion vectors mvL0zeroCandm and mvL1zeroCandm of every new candidate zeroCandm being added in mergeCandList. The number of candidates being added numZeroMergeCand is set equal to ( numMergeCand – numOrigMergeCand – numCombMergeCand ). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand − 1, inclusive.
   4. The variable MergeIdx is derived as follows.
      * If use\_mvi\_flag is equal to 0, MergeIdx is set equal to merge\_idx[ xP][ yP ].
      * Otherwise (use\_mvi\_flag is equal to 1), MergeIdx is set equal to merge\_idx[ xP][ yP ] −  1.
   5. The following assignments are made with N being the candidate at position MergeIdx ~~merge\_idx[ xP][ yP ]~~ in the merging candidate list mergeCandList ( N = mergeCandList[ MergeIdx ~~merge\_idx[ xP][ yP ]~~ ] ) and X being replaced by 0 or 1:
      * 1. mvLX[ 0 ] = mvLXN[ 0 ] (G‑89)
        2. mvLX[ 1 ] = mvLXN[ 1 ] (G‑90)
        3. refIdxLX = refIdxLXN (G‑91)
        4. predFlagLX = predFlagLXN (G‑92)
   6. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nPbW + nPbH ) is equal to 12, the following applies.
      * 1. refIdxL1 = −1 (G‑93)
        2. predFlagL1 = 0 (G‑94)
   7. The disparity availability flag ivpMvFlagLX and the disparity vector ivpMvDispLX are derived as follows (with X being replace by 0 or 1).
      * If all of the following conditions are true, ivpMvFlagLX is set equal to 1 and ivpMvDispLX is set equal to mvDisp [Ed. (GT) There is some redundancy in draft and software since ivpMvDispLX is derived for each list, although it is always equal for both lists.]
        + availableFlagIvMC = = 1
        + MergeIdx = = 0
        + predFlagLXIvMC = = 1

[Ed. (GT): PredMode[ xC ][ yC ] = = MODE\_SKIP might be added here instead of testing it in the disparity vector derivation process]

* + - Otherwise, ivpMvFlagLX is set equal to 0 and both components of ivpMvDispLX are set equal to 0.