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 flag vspModeFlag, specifying, whether the current PU is coded using view synthesis prediction,
* the variable refViewIdx specifying a reference view for VSP,
* the flag availableFlagIvMC, specifying whether the IvMc candidate is available.

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 H.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, refIdxLX, and mergeFlag being equal to 1 as the inputs and with the output being the availability flag availableFlagLXCol, the temporal motion vector mvLXCol, and the reference index refIdxLXCol. The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.
     + 1. availableFlagCol = availableFlagL0Col | | availableFlagL1Col (H‑85)   
          predFlagLXCol = availableFlagLXCol (H‑86)
  4. Depending on iv\_mv\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If multi\_view\_mv\_pred\_flag is equal to 0, the flags availableFlagIvMC and availableFlagIvDC are set equal to 0.
     + Otherwise (iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause H.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. Depending on view\_synthesis\_pred\_flag, the following applies.
     1. [ Ed. (GT): The variable view\_synthesis\_pred\_flag is missing in software and HLS. ]
     + If view\_synthesis\_pred\_flag is equal to 0, or the DepthFlag is equal to 1, or PartMode[ xC ][ yC ] is not equal to PART\_2Nx2N, the flag availableFlagVSP is set equal to 0 and the variable refViewIdx is set equal to −1.
     + Otherwise (view\_synthesis\_pred\_flag is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause H.8.5.2.1.12 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 input and the outputs are the availability flag availableFlagVSP, the reference indices refIdxL0VSP and refIdxL1VSP, the prediction list utilization flags predFlagL0VSP and predFlagL1VSP, the motion vectors mvL0VSP and mvL1VSP, and reference view index refViewIdx.
  6. Depending on DepthFlag, the following applies.
     + If DepthFlag is equal to 0, the variable availableFlagT is set equal to 0.
     + Otherwise ( DepthFlag is equal to 1), the derivation process for the texture merging candidate as specified in subclause H.8.5.2.1.13 is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH as the inputs and the outputs are the flag availableFlagT, the prediction utilization flags predFlagL0T and predFlagL1T, the reference indices refIdxL0T and refIdxL1T, and the motion vectors mvL0T and mvL1T.
  7. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps.
  8. The variable numMergeCand is set equal to 0.
  9. The variable vspCandInListFlag is set equal to 0.
  10. When availableFlagT is equal to 1, the insertion process for a candidate from a potentially view synthesis predicted neighbour as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to T, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.
  11. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC and the variable numMergeCand is increased by 1.
  12. When availableFlagA1 is equal to 1, the following applies:
      + - When the following condition is true,
          * availableFlagT = = 0 && availableFlagIvMC  = = 0,
          1. or one or more of the following conditions are true, with N being replaced by T and IvMC:
          * availableFlagN = = 1 && predFlagLXN ! = predFlagLXA1, (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && mvLXN ! = mvLXA1 (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && refIdxLXN ! = refIdxLXA1 (with X being replaced by 0 and 1),

the insertion process for a candidate from a potentially view synthesis predicted neighbour block as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to A1, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.

* 1. When availableFlagB1 is equal to 1, the following applies:
     + - When the following condition is true ,
         * availableFlagT = = 0 && availableFlagIvMC = = 0,

or one or more of the following conditions is true, with N being replaced by T and IvMC:

* + - * + availableFlagN = = 1 && predFlagLXN ! = predFlagLXB1, (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && mvLXN ! = mvLXB1 (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && refIdxLXN ! = refIdxLXB1 (with X being replaced by 0 and 1),

the insertion process for a candidate from a potentially view synthesis predicted neighbour block as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to B1, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.

* 1. When availableFlagB0 is equal to 1, the insertion process for a candidate from a potentially view synthesis predicted neighbour block as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to B0, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.
  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 insertion process for a candidate from a potentially view synthesis predicted neighbour block as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to A0, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.
  2. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the insertion process for a candidate from a potentially view synthesis predicted neighbour block as specified in subclause H.8.5.2.1.1.1 is invoked with the list mergeCandList, the variable numMergeCand, the variable vspCandInListFlag, the candidate position indicator N being equal to B2, the luma location ( xP, yP ), and the variables nPbW and nPbH as the inputs and the outputs are the list mergeCandList, the variable numMergeCand and the flag vspCandInListFlag.
  3. When availableFlagVSP is equal to 1, numMergeCand is less than 4 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag and vspCandInListFlag is 0, vspCandInListFlag is set equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to VSP and the variable numMergeCand is increased by 1.
  4. When availableFlagCol is equal to 1 and numMergeCand is less than 5 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.

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 following assignments are made with N being the candidate at position merge\_idx[ xP][ yP ] in the merging candidate list mergeCandList ( N = mergeCandList[ merge\_idx[ xP][ yP ] ] ) and X being replaced by 0 or 1:
      * 1. mvLX[ 0 ] = mvLXN[ 0 ] (H‑87)
        2. mvLX[ 1 ] = mvLXN[ 1 ] (H‑88)
        3. refIdxLX = refIdxLXN (H‑89)
        4. predFlagLX = predFlagLXN (H‑90)
   5. 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 (H‑91)
        2. predFlagL1 = 0 (H‑92)
   6. The variable vspModeFlag is set equal to ( mergeCandList[ merge\_idx[ xP][ yP ] ] = = VSP ).

[Ed. (GT): In software VspModeFlag[ x ][ y ] is used in several places (deblocking filter, AMVP) this is not yet reflected in the draft.]

* 1. 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.(3D-I0008) ]
       - availableFlagIvMC = = 1
       - merge\_idx[ xP][ yP ] = = 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.