**Issue3: Derivation of the VSP merge candidate**

**H.8.5.2.1.12** **Derivation process for a view synthesis prediction merge candidate**

Inputs to 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,

Outputs of this process are

* the availability flag availableFlagVSP whether the VSP merge candidate is available,
* the reference indices refIdxL0VSP and refIdxL1VSP ,
* the prediction list utilization flags predFlagL0VSP and predFlagL1VSP,
* the motion vectors mvL0VSP and mvL1VSP.

1. The variable availableFlagVSP is set equal to 1, the variables predFlagL0VSP and predFlagL1VSP are set equal to 0, the variables refIdxL0VSP and refIdxL1VSP are set equal to −1 and the variable refViewAvailableFlag is set equal to 0.

* For X in the range of 0 to 1, inclusive, the following applies:
  + For i in the range of 0 to NumRefPicsLX – 1, inclusive, the following applies:
    - When refViewAvailableFlag is equal to 0 and ViewIdx( RefPicListX[ i ] ) is equal to RefViewIdx[ xC ][ yC ], the following applies:
      * 1. refViewAvailableFlag = 1 (H‑)  
           predFlagLXVSP = 1 (H‑)  
           mvLXVSP = MvDisp[ xC ][ yC ] (H‑)  
           refIdxLXVSP = i (H‑)  
           Y = 1 – X (H‑)

When the current slice is a B slice and refViewAvailableFlag is equal to 1, refViewAvailableFlag is set equal to 0 and the following applies:

* + For i in the range of 0 to NumRefPicsLY – 1, inclusive, the following applies.
    - When refViewAvailableFlag is equal to 0 and ViewIdx( RefPicListY[ i ] ) is not equal to RefViewIdx[ xC ][ yC ] and ViewIdx( RefPicListY[ i ] ) is not equal to ViewIdx, the following applies:
      * 1. tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td
        2. distScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 )
        3. dvDispScale = MvDisp[ xC ][ yC ]

dvDispScale[ 0 ] =Clip3( −32768, 32767, Sign2( distScaleFactor \* MvDisp[ xC ][ yC ][ 0 ] ) \* ( (Abs( distScaleFactor \* MvDisp[ xC ][ yC ] ) + 127 ) >> 8 ) )

* + - * 1. where td and tb are derived as
        2. td = Clip3( −128, 127, ViewIdx – RefViewIdx[ xC ][ yC ])
        3. tb = Clip3( −128, 127, ViewIdx – ViewIdx( RefPicListY[ i ] ))
        4. refViewAvailableFlag = 1 (‑137)  
           predFlagLYVSP = 1 (‑138)  
           mvLYVSP =dvDispScale (‑139)  
           refIdxLYVSP = i (‑140)

H.8.5.2.1.10 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 flag vspModeFlag, specifying, whether the current PU is coded using view synthesis prediction,

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, the prediction utilization flags predFlagL0 and predFlagL1, the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1 and the flag vspModeFlag are derived as specified by the following ordered steps:

* 1. The derivation process for spatial merge candidates 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 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 (‑81)   
          predFlagLXCol = availableFlagLXCol (‑82)
  4. Depending on iv\_mv\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If iv\_mv\_pred\_flag[ nuh\_layer\_id ] 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 is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, as the inputs and the output is assigned to 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)..
  5. Depending on view\_synthesis\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 0, the flag availableFlagVSP is set equal to 0.
     + Otherwise (view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause is invoked with the luma locations ( xC, yC ) as input and the outputs are the availability flag availableFlagVSP, the reference indices refIdxL0VSP and refIdxL1VSP, the prediction list utilization flags predFlagL0VSP and predFlagL1VSP, and the motion vectors mvL0VSP and mvL1VSP.
  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 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 lists mergeCandList ~~and~~ mergeCandIsVspFlag, inheritedRefViewIdx and inheritedMvDisp are constructed as specified by the following ordered steps:
  8. The variable numMergeCand is set equal to 0.
  9. When availableFlagT is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to T, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  10. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  11. 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 entry mergeCandList[ numMergeCand ] is set equal to A1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xP − 1 ][ yP + nPbH − 1 ], the entry inheritedRefViewIdx[ numMergeCand ] is set equal to RefViewIdx[ xP – 1 ][ yP + nPbH – 1 ]. the entry inheritedMvDisp[ numMergeCand ] is set equal to MvDisp[ xP – 1 ][ yP + nPbH – 1 ], and the variable numMergeCand is increased by 1.

* 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 entry mergeCandList[ numMergeCand ] is set equal to B1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xP + nPbW − 1 ][ yP − 1 ] the entry inheritedRefViewIdx[ numMergeCand ] is set equal to RefViewIdx[ xP + nPbW − 1 ][ yP − 1 ]. the entry inheritedMvDisp[ numMergeCand ] is set equal to MvDisp[ xP + nPbW − 1 ][ yP − 1 ], and the variable numMergeCand is increased by 1.

* 1. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xP + nPbW ][ yP − 1 ], the entry inheritedRefViewIdx[ numMergeCand ] is set equal to RefViewIdx[ xP + nPbW ][ yP − 1 ]. the entry inheritedMvDisp[ numMergeCand ] is set equal to MvDisp[ xP + nPbW ][ yP − 1 ], 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, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. When availableFlagVSP is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to VSP, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal 1 and the variable numMergeCand is increased by 1.
  2. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xP − 1 ][ yP + nPbH ], the entry inheritedRefViewIdx[ numMergeCand ] is set equal to RefViewIdx[ xP − 1 ][ yP + nPbH ], the entry inheritedMvDisp[ numMergeCand ] is set equal to MvDisp[ xP − 1 ][ yP + nPbH ], and the variable numMergeCand is increased by 1.
  3. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to B2, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xP − 1 ][ yP − 1 ], the entry inheritedRefViewIdx[ numMergeCand ] is set equal to RefViewIdx[ xP − 1 ][ yP – 1 ], the entry inheritedMvDisp[ numMergeCand ] is set equal to MvDisp[ xP − 1 ][ yP – 1 ], 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, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  5. The variable numOrigMergeCand is set equal to numMergeCand.
  6. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause is invoked with mergeCandList, mergeCandIsVspFlag, 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, and mergeCandIsVspFlag[ numOrigMergeCand + k ] is set equal to 0.
  7. 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, and mergeCandIsVspFlag[ numOrigMergeCand + numCombMergeCand + m ] is set equal to 0.
  8. 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 ] (‑83)
       2. mvLX[ 1 ] = mvLXN[ 1 ] (‑84)
       3. refIdxLX = refIdxLXN (‑85)
       4. predFlagLX = predFlagLXN (‑86)
  9. 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 (‑87)
       2. predFlagL1 = 0 (‑88)
  10. The variable vspModeFlag is set equal to mergeCandIsVspFlag[ merge\_idx[ xP][ yP ] ]..
  11. When vspModeFlag is equal to 1, the following applies:
      + If RefPicList0[ 0 ] is a long-term reference picture, refIdxL0 is set equal to 0.
      + Otherwise ( RefPicList0[ 0 ] is a short-term reference picture ), refIdxL0 is set equal to AltRefIdxL0.
      + For i in the range of xP to xP+ nPbW -1, inclusive, the following applies:
        - For j in the range of yP to yP+ nPbH – 1, inclusive, the following applies:

RefViewIdx[ i ][ j ] = inheritedRefViewIdx[ merge\_idx[ xP][ yP ] ] (‑87)

MvDisp[ i ][ j ] = inheritedMvDisp[ merge\_idx[ xP][ yP ] ] (‑87)

* 1. The disparity availability flag ivpMvFlagLX is 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

[Ed. (GT) There is some redundancy in draft and software since disparities equal for both lists.(#7) ]

* + - * 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.

**Issue 4: View synthesis prediction process**

8.5.2.2 Decoding process for inter prediction samples

……….

* Otherwise, ( VspModeFlag[ xC + xB ][ yC + yB ] is equal to 1 ), the following applies:
  + For X in the range of 0 to 1, inclusive, the following applies.
    - When predFlagLX is equal to 1, the arrays predSamplesL ,predSampleCb, and predSampleCr are derived by invoking the view synthesis prediction process as specified in subclause , with the luma locations ( xC, yC ), ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, ~~the motion vector mvLX,~~ the prediction list indicator X, and the reference index refIdxLX as the inputs and the outputs are the sample arrays predSamplesLXL, predSamplesLXCb, and predSamplesLXCr.
  + The array predSampleL of the prediction samples of luma component is derived by invoking the weighted sample prediction process specified in subclause 8.5.2.2.3 with the luma location ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, and the sample arrays predSamplesL0L and predSamplesL1L as well as predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 0 given as input.
  + The array predSampleCb of the prediction samples of component Cb is derived by invoking the weighted sample prediction process specified in subclause 8.5.2.2.3 with the chroma location ( xB/2, yB/2 ), the width and the height of the current chroma prediction block nPbWCb set equal to nPbW/2, nPbHCb set equal to nPbH/2, and the sample arrays predSamplesL0Cb and predSamplesL1Cb as well as predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 1 given as input.
  + The array predSampleCr of the prediction samples of component Cr is derived by invoking the weighted sample prediction process specified in subclause 8.5.2.2.3 with the chroma location ( xB/2, yB/2 ), the width and the height of the current chroma prediction block nPbWCr set equal to nPbW/2, nPbHCr set equal to nPbH/2, and the sample arrays predSamplesL0Cr and predSamplesL1Cr as well as predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 2 given as input.

8.5.2.2.7 View synthesis prediction process

Inputs to this process are:

* a location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top left sample of the current picture,
* a location ( xB, yB ) specifying the top-left sample of the current prediction block relative to the top left sample of the current coding block,
* the width and height of this prediction block, nPbW and nPbH,
* the prediction list indicator X
* ~~a motion vector~~ ~~mvLX~~
* the reference index refIdxLX

Outputs of this process are:

– an array predSamplesL of luma prediction samples,

– an array predSamplesCb of chroma prediction samples for the component Cb

– an array predSamplesCr of chroma prediction samples for the component Cr

The location ( xP, yP ) given in full-sample units of the upper-left luma samples of the current prediction block relative to the upper-left luma sample location of the current picture is derived by:

xP = xC + xB (H‑220)  
yP = yC + yB (H‑221)

The reference picture consisting of an ordered two-dimensional array refPicL of luma samples and two ordered two-dimensional arrays refPicCb and refPicCr of chroma samples is derived by invoking the process specified in subclause 8.5.2.2.1 with refIdxLX as input.

The variable refViewIdx is set equal to the ViewIdx( RefPicListX[ refIdxLX ] ) and the variable depthViewIdx is set equal to RefViewIdx[ xC + xB ][ yC + yB ] and The variable mvDisp is set equal to MvDisp[ xC + xB ][ yC + yB ].

The derivation process for disparity sample array as specified in section H.8.5.4.3 is invoked with the luma location ( xP, yP ), the disparity vector mvDisp ~~being equal to mvLX~~, the variable refViewIdx, the variable depthViewIdx, the variables nPSW and nPSH, the variable nSubBlkW being equal 4 and the variable nSubBlkW being equal to 4 as the inputs, and the output is the array disparitySamples of size (nPSW)x(nPSH).

Let ( xIntL, yIntL ) be a luma location given in full-sample units and ( xFracL, yFracL ) be an offset given in quarter-sample units.

For each luma sample location ( xL = 0..nPbW−1, yL = 0..nPbH−1 ) inside the prediction luma sample array predSamplesL, the corresponding prediction luma sample value predSamplesL[ xL ][ yL ] is derived as follows:

* The variables xIntL, yIntL, xFracL, and yFracL are derived by

xIntL = xP + xL+ disparitySamples[ xL ][ yL ] (H‑222)  
yIntL = yP + yL (H‑223)

xFracL = disparitySamples[ xL ][ yL ] & 3 (H‑224)  
yFracL = 0 (H‑225)

* The prediction luma sample value predSamplesL[ xL][ yL ] is derived by invoking the process specified in subclause 8.5.2.2.2.1 with ( xIntL, yIntL ), ( xFracL, yFracL ) and refPicL given as input.

Let ( xIntC, yIntC ) be a chroma location given in full-sample units and ( xFracC, yFracC ) be an offset given in one-eighth sample units.

For each chroma sample location ( xC = 0..nPbW/2−1, yC = 0..nPbH/2−1 ) inside the prediction chroma sample arrays predSamplesCb and predSamplesCr, the corresponding prediction chroma sample values predSampleLXCb[ xC ][ yC ] and predSamplesCr[ xC ][ yC ] are derived as follows:

* The variables xIntC, yIntC, xFracC, and yFracC are derived by

xIntC = ( xP / 2 ) + xC + disparitySamples[ xC << 1 ][ yC << 1 ] (H‑226)  
yIntC = ( yP / 2 ) + yC (H‑227)

xFracC = disparitySamples[ xC << 1][ yC << 1 ] & 7 (H‑228)  
yFracC = 0 (H‑229)

* The prediction sample value predSamplesCb[ xC ][ yC ] is derived by invoking the process specified in subclause 8.5.2.2.2.2 with ( xIntC, yIntC ), ( xFracC, yFracC ) and refPicCb given as input.
* The prediction sample value predSamplesCr[ xC ][ yC ] is derived by invoking the process specified in subclause 8.5.2.2.2.2 with ( xIntC, yIntC ), ( xFracC, yFracC ) and refPicCr given as input.

**Issue 5: Derivation of the Inter-view DV merge candidate**

**H.8.5.2.1.11 Derivation process for a disparity inter-view motion vector candidate**

This process is not invoked when iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0.

Inputs to 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 nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit,
* a prediction list indication X,
* a reference view index refViewIdx,
* a disparity vector mvDisp,
* a flag mergeFlag specifying whether a merge candidate is derived,
* a reference index refIdxLX specifying a reference picture in the reference picture list RefPicListLX.

Outputs of this process are:

* a flag availableFlagLXInterView specifying whether the disparity inter-view motion vector candidate is available,
* a disparity inter-view motion vector candidate mvLXInterView (if availableFlagLXInterView is equal to 1),
* a reference index refIdxLX specifying a reference picture in the reference picture list RefPicListLX.

The flag availableFlagLXInterView is set equal to 0, both components of mvLXInterView are set equal to 0.

When X is equal to 1 and the current slice is not a B slice the whole decoding process specified in this subclause terminates.

The motion vector mvLXInterView is derived by

mvLXInterView[ 0 ] = mvDisp[ 0 ] (‑129)  
mvLXInterView[ 1 ] = 0 (‑130)

refIdxLX = RefViewIdx[ xP ][ yP ]

~~For each i from ( mergeFlag ? 0 : refIdxLX ) to ( mergeFlag ? num\_ref\_idx\_lX\_active\_minus1 : refIdxLX), inclusive, the following applies:~~

* ~~When PicOrderCntVal of the picture RefPicListX[ i ] is equal to the PicOrderCntVal or the current picture and availableFlagLXInterView is equal to 0, availableFlagLXInterView is set equal to 1 and the following applies:~~
* ~~The motion vector mvLXInterView is derived by~~

~~mvLXInterView[ 0 ] = mvDisp[ 0 ] (‑129)  
mvLXInterView[ 1 ] = 0 (‑130)~~

* ~~When mergeFlag is equal to 1, the reference index refIdxLX is derived by~~

~~refIdxLX = i~~

**Issue 7: DV derivation from the temporal neighboring blocks**

**8.5.4.2 Derivation process for a disparity vector in a block of a candidate picture**

Inputs to this process are:

* a candidate picture candPic,
* a luma location ( xPCand , yPCand ) relative to the top-left luma sample of the candidate picture.

Outputs of this process are:

* a flag availableDV specifying whether the disparity vector is available,
* a disparity vector mvDisp
* a reference view order index refViewIdx.

The motion vector mvDisp and the availability flag availableDV are derived as specified by the following ordered steps and the whole decoding process of this sub-clause terminates once availableDV is set equal to 1.

* 1. Let colPu the prediction unit in candPic covering the position ( ( xPCand  >>  4 ) <<4 ,  ( yPCand  >>  4 ) <<4 ).
  2. The position ( xPCol, yPCol ) is set equal to the position of the top-left sample of colPu relative to the top-left luma sample of the candPic.
  3. If slice\_type is equal to B, the variable dir is set equal to collocated\_from\_l0\_flag, otherwise, dir is set equal to 1 – collocated\_from\_l0\_flag.
  4. The flag availableDV is set equal to 0, and both components of mvDisp are set equal to 0.
  5. For each X from dir to 1 – dir, inclusive, the following applies:
     + Let candPicRefPicList be the reference picture list RefPicListX of candPic.
     + The variable candPredFlag is set equal to the prediction list utilization flag PredFlagLX of candPic.
     + The variable candRefIdx is set equal to the reference indices RefIdxLX of candPic.
     + The variable candMV is set equal to the motion vectors MvLX of candPic.
     + When the ViewIdx of candPicRefPicList[ candRefIdx[ xPCol ][ yPCol ] ] is not equal to the ViewIdx of candPic ~~and~~, candPredFlag[ xPCol ][ yPCol ] is equal to 1 and there is an inter-view reference picture with view index equal to view index of candPicRefPicList[ candRefIdx[ xPCol ][ yPCol ] ] exists in the reference lists of current CU, the following applies:

mvDisp = candMV[ xPCol ][ yPCol ] (H‑250)  
availableDV = 1 (H‑251)  
refViewIdx = ViewIdx of candPicRefPicList[ candRefIdx[ xPCol ][ yPCol ] ] (H‑252)