* + - * 1. 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
* 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:

* 1. xP = xC + xB (H‑208)  
     yP = yC + yB (H‑209)

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 ]. The variable mvDisp is set equal to MvDisp[ xC + xB ][ yC + yB ]

[Ed. (GT) In software refIdxLX is used to derive the depth picture, whereas the reference index of the texture picture is derived similar to the process specified in the derivation process for a view synthesis prediction merge candidate. ]

The derivation process for disparity sample array as specified in section H.8.5.4.2 is invoked with the luma location ( xP, yP ), the disparity vector mvDisp, the variable refViewIdx, the variable depthViewIdx, the variables nPSW and nPSH, the variable nSubBlkW being equal 8 the variable nSubBlkW being equal to 8 and the variable splitFlag equal to 1, 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
  1. xIntL = xP + xL+ disparitySamples[ xL ][ yL ] (H‑210)  
     yIntL = yP + yL (H‑211)
  2. xFracL = disparitySamples[ xL ][ yL ] & 3 (H‑212)  
     yFracL = 0 (H‑213)
* 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.

[Ed. (GT): As for inter prediction the treatment of colour planes for depth needs to be discussed. In software colour planes are set to 128 in VSP process. (#12)]

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
  1. xIntC = ( xP / 2 ) + xC + disparitySamples[ xC << 1 ][ yC << 1 ] (H‑214)  
     yIntC = ( yP / 2 ) + yC (H‑215)
  2. xFracC = disparitySamples[ xC << 1][ yC << 1 ] & 7 (H‑216)  
     yFracC = 0 (H‑217)
* 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.
  + 1. Decoding process for the residual signal of coding units coded in inter prediction mode

Inputs to this process are:

* a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top left luma sample of the current picture,
* a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

* a (nCSL)x(nCSL) array resSamplesL of luma residual samples, where nCSL is derived as specified below,
* a (nCSC)x(nCSC) array resSamplesCb of chroma residual samples for the component Cb, where nCSC is derived as specified below,
* a (nCSC)x(nCSC) array resSamplesCr of chroma residual samples for the component Cr, where nCSC is derived as specified below.

The variable nCSL is set equal to 1 << log2CbSize and the variable nCSC is set equal to ( 1 << log2CbSize ) >> 1.

Let resSamplesL be a (nCSL)x(nCSL) array of luma residual samples and let resSamplesCb and resSamplesCr be two (nCSC)x(nCSC) arrays of chroma residual samples.

* If inter\_sdc\_flag is equal to 0, the following applies, depending on no\_residual\_syntax\_flag, the following applies:
  + If no\_residual\_syntax\_flag is equal to 1, all samples of the (nCSL)x(nCSL) array resSamplesL and all samples of the two (nCSC)x(nCSC) arrays resSamplesCb and resSamplesCr are set equal to 0.
  + Otherwise (no\_residual\_syntax\_flag is equal to 0), the following ordered steps apply:
    1. The decoding process for luma residual blocks as specified in subclause 8.5.3.1 below is invoked with the luma location ( xC, yC ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCS set equal to nCSL, and the (nCSL)x(nCSL) array resSamplesL as the inputs and the output is a modified version of the (nCSL)x(nCSL) array resSamplesL.
    2. The decoding process for chroma residual blocks as specified in subclause 8.5.3.2 below is invoked with the luma location ( xC, yC ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 1, the variable nCS set equal to nCSC, and the (nCSC)x(nCSC) array resSamplesCb as the inputs and the output is a modified version of the (nCSC)x(nCSC) array resSamplesCb.
    3. The decoding process for chroma residual blocks as specified in subclause 8.5.3.2 below is invoked with the luma location ( xC, yC ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCS set equal to nCSC, and the (nCSC)x(nCSC) array resSamplesCr as the inputs and the output is a modified version of the (nCSC)x(nCSC) array resSamplesCr.
* Otherwise (inter\_sdc\_flag is equal to 1), the decoding process for simplified depth coded residual blocks as specified in subclause H.8.5.3.3 is invoked with the luma location ( xC, yC ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCS set equal to nCSL, and the (nCSL)x(nCSL) array resSamplesL as the inputs and the output is a modified version of the (nCSL)x(nCSL) array resSamplesL.

For x in the range of 0 to nCSL − 1 and y in the range of 0 to nCSL − 1, the following applies:

* + ResSamplesL[ xC + x ][ yC + y ] is set equal to resSamplesL[ x ][ y ].

For x in the range of 0 to nCSC − 1 and y in the range of 0 to nCSC − 1, the following applies:

* + ResSamplesCb[ xC /2 + x ][ yC /2 + x] is set equal to resSamplesCb[ x ][ y ].
  + ResSamplesCr[ xC /2 + x ][ yC /2 + x ] is set equal to resSamplesCr[ x ][ y ].
    - 1. Decoding process for luma residual blocks

The specification in subclause 8.5.3.1 applies.

* + - 1. Decoding process for chroma residual blocks

The specification in subclause 8.5.3.2 applies.

* + - 1. Decoding process for simplified depth coded residual blocks

Inputs to this process are:

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

– a variable nCS specifying the size of the current luma coding block,

– a (nCS)x(nCS) array resSamples of luma residual samples.

Output of this process is:

– a modified version of the (nCS)x(nCS) array of luma residual samples.

The values of the variables xOff, yOff, and curInterSdcResi[ i ] for i in the range of 0 to 3, inclusive, depending on the value of PartMode are specified in Table H‑11.

Table H‑11 – Specification of the variables xOff, yOff, and interSdcResiIdx[ i ]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PartMode** | **xOff** | **yOff** | **interSdcResiIdx**[ i ] | | | |
| i = 0 | i = 1 | i = 2 | i = 3 |
| PART\_2Nx2N | nCS | nCs | 0 | 0 | 0 | 0 |
| PART\_2NxN | nCS | ( nCS  >>  1 ) | 0 | 0 | 1 | 1 |
| PART\_2NxnU | nCS | ( nCS  >>  2 ) | 0 | 0 | 1 | 1 |
| PART\_2NxnD | nCS | ( nCS  >>  1 ) + ( nCS  >>  2 ) | 0 | 0 | 1 | 1 |
| PART\_Nx2N | ( nCS  >>  1 ) | nCS | 0 | 1 | 0 | 1 |
| PART\_nLx2N | ( nCS  >>  2 ) | nCS | 0 | 1 | 0 | 1 |
| PART\_nRx2N | ( nCS  >>  1 ) + (  nCS >>  2 ) | nCS | 0 | 1 | 0 | 1 |
| PART\_NxN | ( nCS  >>  1 ) | ( nCS  >>  1) | 0 | 1 | 2 | 3 |

For x in the range of 0 to nCS the following applies:

* For y in the range of 0 to nCS the following applies:
  + The variable i is derived as specified in the following:
    - If x is less than xOff and y is less than yOff, i is set equal to 0.
    - Otherwise, if x greater than or equal to xOff and y is less than to yOff, i is set equal to 1.
    - Otherwise, if x less than xOff and y is greater than or equal to yOff, i is set equal to 2.
    - Otherwise, ( x is greater than or equal to xOff and y is greater than or equal to yOff), i is set equal to 3.
  + The value of resSamples[ x ][ y ] is set equal to InterSdcResi[ xC ][ yC ][ interSdcResiIdx[ i ] ]
    1. Derivation process for disparity vectors

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,
* a variable nCS specifying the size of the current luma coding block,

The flag availableDV is set equal to 0, and both components of the disparity vector mvDisp are set equal to 0.

The variable checkParallelMergeFlag is derived as follows:

* If one or more of the following conditions are true, checkParallelMergeFlag is set equal to 1.
  + PredMode[ xC ][ yC ] is equal to MODE\_SKIP.
  + PredMode[ xC ][ yC ] is equal to MODE\_INTER and merge\_flag[ xC ][ yC ] is equal to 1.
* Otherwise, checkParallelMergeFlag is set equal to 0.

The derivation process for a disparity vector from temporal neighbour block as specified in H.8.5.4.1 is invoked with the luma location ( xC, yC ), and the variable nCS as inputs, and the outputs are the flag availableDV, the disparity vector mvDisp and the reference view order index refViewIdx.

When availableDV is equal to 0, for each N being A1, B1 and ( xN, yN ) being ( xC − 1,  yC + nCS − 1 ), ( xC + nCS − 1,  yC − 1 ), respectively, the following ordered steps apply.

* 1. When yC − 1 is less than ( ( yC  >>  Log2CtbSizeY )  <<  Log2CtbSizeY ), the following applies.
     + - 1. xB1 = ( ( xB1  >>  3 )  <<  3 ) + ( ( xB1  >>  3 ) & 1) \* 7 (H‑218)
  2. The derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to the ( xC, yC ) and the luma location ( xN, yN ) as the input and the output assigned to availableN.
  3. When availableN is equal to 1 and PredMode[ xN ][ yN ] is equal to MODE\_INTRA, availableN is set equal to 0. [Ed. (GT): 2+3 correspond to 6.4.2 for CU when ( xN, yN ) outside CU. Cross-check appreciated. ]
  4. When all of the following conditions are true, availableN is set equal to 0.
     + checkParallelMergeFlag is equal to 1
     + ( xC  >>  ( log2\_parallel\_merge\_level\_minus2 + 2) ) is equal to  
       ( xN  >>  ( log2\_parallel\_merge\_level\_minus2 + 2) )
     + ( yC  >>  ( log2\_parallel\_merge\_level\_minus2 + 2) ) is equal to ( yN  >>  ( log2\_parallel\_merge\_level\_minus2 + 2) ).
  5. The flag availableIvpMvSearchFlagN is set equal to availableN.
  6. When one of the following conditions is true, N is equal to B1 and ( ( yN  >>  Log2CtbSizeY )  <<  Log2CtbSizeY ) is less than ( ( yC >> Log2CtbSizeY )  <<  Log2CtbSizeY), availableIvpMvSearchFlagN is set equal to 0.
  7. The flag availableFlagIvpMvN is set equal to 0.
  8. For each X from 0 to 1, the following applies:
     + When availableDV is equal to 0, availableN is equal to 1, RefIdxLX[ xN ][ yN ] is greater than or equal to 0, and PredFlagLX[ xN ][ yN ] is equal to 1, the following applies:
       - If RefPicListX[ RefIdxLX[ xN ][ yN ] ] is an inter-view reference picture, the following applies:

refViewIdx = ViewIdx( RefPicListX[ RefIdxLX[ xN ][ yN ] ] ) (H‑219)  
mvDisp = MvLXN[ xN ][ yN ] (H‑220)  
availableDV = 1 (H‑221)

* + - * Otherwise (RefPicListX[ RefIdxLX[ xN ][ yN ] ] is not an inter-view reference picture), the following applies:
        + When availableIvpMvSearchFlagN is equal to 1, availableFlagIvpMvN is equal to 0, and PredMode[ xN ][ yN ] is equal to MODE\_SKIP and IvpMvFlagLX[ xN ][ yN ] is equal to 1, the following applies:

ivpMvDispN = MvDispRefined[ xN ][ yN ] (H‑222)  
refViewIdxN = RefViewIdx[ xN ][ yN ] (H‑223)  
availableFlagIvpMvN = 1 (H‑224)  
[Ed. (GT). In HTM-7.0 refViewIdxN (related to #5) is set equal to 0 and ivpMvDispN[ 0 ] are set equal to 0. ]

When availableDV is equal to 0 for each N being A1 and B1, the following applies.

* + When availableDV is equal to 0 and availableFlagIvpMvN is equal to 1, the following applies:
    - 1. mvDisp = ivpMvDispN (H‑225)  
         refViewIdx = refViewIdxN (H‑226)  
         availableDV = 1 (H‑227)

When availableDV is equal to 0, refViewIdx is set equal to 0, and mvDisp is set equal to ( 0, 0 ). The variable mvRefinedDisp is set equal to mvDisp.

[Ed. (GT): It is currently an issue under non-CTC that a view with refViewIdx equal to 0 might not be available in the reference picture list. In software a preliminary fix has been incorporated, to enable simulations in IBP configuration. ]

When depth\_refinement\_flag[ nuh\_layer\_id ]is equal to 1, the following ordered steps apply:

* 1. The derivation process for disparity sample array as specified in subclause H.8.5.4.2 is invoked with the luma locations xC, yC, the disparity vector mvDisp, the view identifier refViewIdx, the variable nPSW equal to nCS, the variable nPSH equal to nCS, the variable nSubBlkW equal to nCS, the variable nSubBlkH equal to nCS, and the variable splitFlag equal to 0, as the inputs, and the output is the array disparitySamples of size (nCS)x(nCS).
  2. The horizontal component of the disparity vector mvRefinedDisp[ 0 ] is set equal to disparitySamples[ 0 ][ 0 ].

[Ed. (GT): The disparity vector derivation has the problem related to refViewIdx as described in the editor's comment in subclause H.8.5.2.1 and needs further discussion. (#5) ]

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xC.. ( xC + nCS − 1 ), y = yC..( yC + nCS− 1 ):

* 1. MvDisp[ x ][ y ] = mvDisp (H‑228)  
     MvRefinedDisp[ x ][ y ] = mvRefinedDisp (H‑229)  
     RefViewIdx[ x ][ y ] = refViewIdx (H‑230)  
     DefaultDispFlag[ x ][ y ] = !availableDV (H‑231)
     + 1. Derivation process for a disparity vector from temporal neighbour blocks

Inputs to this process are

* a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block.

Outputs of this process are

* the disparity vector mvDisp,
* the reference view order index refViewIdx,
* the availability flag availableFlag.

The luma location ( xCCtr , yCCtr) specifying the centre position of the current luma coding block is derived as follows:

* 1. xCCtr = xC + ( nCS >> 1 ) (H‑232)
  2. yCCtr = yC + ( nCS >> 1 ) (H‑233)

The flag availableFlag is set equal to 0, and mvDisp is set equal to ( 0, 0 ).

For i from 0 to NumDdvCandPics − 1, inclusive, the following ordered steps apply and the whole decoding process of this sub-clause terminates once availableFlag is set to 1.

* 1. Let colPu the prediction unit in DdvCandPicsList[ i ] covering the position ( ( xCCtr  >>  4 ) <<4 ,  ( yCCtr  >>  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 DdvCandPicsList[ i ].
  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. For each X from dir to 1 – dir, inclusive, the following applies:
     + Let candPicRefPicList be the reference picture list RefPicListX of DdvCandPicsList[ i ].
     + The variable candPredFlag is set equal to the prediction list utilization flag PredFlagLX of DdvCandPicsList[ i ].
     + The variable candRefIdx is set equal to the reference indices RefIdxLX of DdvCandPicsList[ i ].
     + The variable candMV is set equal to the motion vectors MvLX of DdvCandPicsList[ i ].
     + When candPredFlag[ xPCol ][ yPCol ] is equal to 1, the following applies:
       - The variable candRefViewIdx is set equal to the ViewIdx of candPicRefPicList[ candRefIdx[ xPCol ][ yPCol ] ].
       - When candRefViewIdx is not equal to the ViewIdx of DdvCandPicsList[ i ] and there is an inter-view reference picture with ViewIdx equal to candViewIdx in RefPicList0 or RefPicList1, the following applies:

refViewIdx = candRefViewIdx (H‑234)  
mvDisp = candMV[ xPCol ][ yPCol ] (H‑235)  
availableFlag = 1 (H‑236)

* + - 1. Derivation process for a disparity sample array

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,
* a disparity vector mvDisp,
* a view order index refViewIdx specifying a reference view,
* a view order index depthViewIdx specifying the view the depth should be derived from
* variables nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit.
* variable nSubBlkW and nSubBlkW specifying the conversion precision of the corresponding depth samples,
* variable splitFlag specifying whether further to split to 8x4 or 4x8 blocks,

Outputs of this process are:

* a (nPSW)x(nPSH) array disparitySamples of disparities values.

Let refDepPels be an array of reconstructed depth samples of the depth view component with ViewIdx equal to depthViewIdx. The luma location (xTL, yTL) of top-left luma sample of a block in refDepPels is derived by

* 1. xTL = xP + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) (H‑237)  
     yTL = yP + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) (H‑238)

The array disparitySamples of size (nPSW)x(nPSH) is derived as specified in the following:

* For sBy in the range of 0 to ( ( nPSH / nSubBlkH) –1 ), inclusive, the following applies:
  + For sBx in the range of 0 to ( ( nPSW / nSubBlkW) –1 ), inclusive, the following applies:
    - The variables xB, y,B, xP0, yP0, xP1, yP1, are derived as specified in the following:

xB = sBx \* nSubBlkW  
 yB = sBy \* nSubBlkH  
 xP0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB )  
 yP0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB )   
 xP1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xB + nSubBlkW – 1 )  
 yP1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + yB + nSubBlkH – 1 )

* + - The variable nSubSubBlkW is set equal to nSubBlkW and the variable nSubSubBlkH is set equal to nSubBlkH.
    - When splitFlag is equal to 1, nSubSubBlkW and nSubSubBlkW are modified as follows:
      * The variable horSplitFlag is derived as specified in the following.
        1. horSplitFlag = ( refDepPels[ xP0 ][ yP0 ] > refDepPels[ xP1 ][ yP1 ] )  
            = = ( refDepPels[ xP1 ][ yP0 ] > refDepPels[ xP0 ][ yP1] ) ) (H‑239)
        2. nSubSubBlkW = horSplitFlag ? nSubSubBlkW : ( nSubSubBlkW >> 1 ) (H‑240)
        3. nSubSubBlkH = horSplitFlag ? ( nSubSubBlkH >> 1 ) : nSubSubBlkW (H‑241)
    - The derivation process for a disparity sample block as specified in subclause H.8.5.4.2.1 is invoked with the luma location ( xB, yB ), variables nSubBlkW and nSubBlkH, the array of reconstructed depth samples refDepPels, the luma location ( xTL, yTL ), the variables nSubSubBlkW and nSubSubBlkW, the view order index refViewIdx, and the array disparitySamples as the inputs, and the output is the modified array disparitySamples.