### New added parts are highlighted in green and deleted parts are marked as ~~strikethrough~~.

* + - 1. General

…

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xBl..xBl + nPbW − 1 and y = yBl..yBl + nPbH − 1:

* 1. MvL0[ xCb + x ][ yCb + y ] = subPbMotionFlag || VspModeFlag[xCb+x][yCb+y] ? SubPbMvL0[ xCb + x ][ yCb + y ] : mvL0 (‑73)
  2. MvL1[ xCb + x ][ yCb + y ] = subPbMotionFlag || VspModeFlag[xCb+x][yCb+y] ? SubPbMvL1[ xCb + x ][ yCb + y ] : mvL1 (‑74)
  3. RefIdxL0[ xCb + x ][ yCb + y ] = subPbMotionFlag ? SubPbRefIdxL0[ xCb + x ][ yCb + y ] : refIdxL0 (‑75)
  4. RefIdxL1[ xCb + x ][ yCb + y ] = subPbMotionFlag ? SubPbRefIdxL1[ xCb + x ][ yCb + y ] : refIdxL1 (‑76)
  5. PredFlagL0[ xCb + x ][ yCb + y ] = subPbMotionFlag ? SubPbPredFlagL0[ xCb + x ][ yCb + y ] : predFlagL0 (‑77)
  6. PredFlagL1[ xCb + x ][ yCb + y ] = subPbMotionFlag ? SubPbPredFlagL1[ xCb + x ][ yCb + y ] : predFlagL1 (‑78)
     + 1. Derivation process for motion vector components and reference indices

…

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in subclause 8.5.3.2.9 is invoked with mvLX as input, and the output being mvCLX.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xPb.. ( xPb + nPbW − 1 ), y = yPb..( yPb + nPbH− 1 ) (with X being either 0 or 1):

* 1. IvpMvFlagLX[ x ][ y ] = ivpMvFlagLX (‑89)  
     VspModeFlag[ x ][ y ] = vspModeFlag (‑90)
  2. SubPbMotionFlag[x][y] = subPbMotionFlag
     + - 1. View synthesis prediction process

…

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‑235)
  2. yIntL = yP + yL (H‑236)
  3. xFracL = disparitySamples[ xL ][ yL ] & 3 (H‑237)
  4. yFracL = 0 (H‑238)
* The prediction luma sample value predSamplesL[ xL][ yL ] is derived by invoking the process specified in subclause 8.5.3.3.3.2 with ( xIntL, yIntL ), ( xFracL, yFracL ) and refPicL given as input.
* SubPbMvLX[xP + xL][yP + yL] = (disparitySamples[ xL][ yL ], 0)
  + 1. 1. Derivation process for a disparity sample array

The array disparitySamples 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 variable maxDep is set equal to –1 and modified as specified in the following.

xSubB = sBx \* nSubBlkW  
 ySubB = sBy \* nSubBlkH  
 xP0 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB )  
 yP0 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + ySubB )   
 xP1 = Clip3( 0, pic\_width\_in\_luma\_samples – 1, xTL + xSubB + nSubBlkW – 1 )  
 yP1 = Clip3( 0, pic\_height\_in\_luma\_samples – 1, yTL + ySubB + nSubBlkH – 1 )  
 maxDep = Max( maxDep, refDepPels[ xP0 ][ yP0 ] )  
 maxDep = Max( maxDep, refDepPels[ xP0 ][ yP1] )  
 maxDep = Max( maxDep, refDepPels[ xP1][ yP0 ] )  
 maxDep = Max( maxDep, refDepPels[ xP1][ yP1 ] )

* + - The values of the array depthSamples are modified as specified in the following:

for ( yOff = 0; yOff < nSubBlkH; yOff++ )  
 for( xOff = 0; xOff < nSubBlkW; xOff++ ) {  
 x = xSubB + xOff   
 y = ySubB + yOff   
 disparitySamples[ x ][ y ] = DepthToDisparityB[ refViewIdx ][ maxDep ]

VspSubBlkW[xP + x][yP + y] = nSubBlkW

VspSubBlkH[xP + x][yP + y] = nSubBlkH   
 }

* 2. In-loop filter process

The specifications in subclause 8.7 apply with the following modifications

– The step 5 of vertical edge filtering in subclause 8.7.2.1 is replaced by

1. The derivation process of prediction block boundary specified in subclause H. is invoked with the luma location ( xCb, yCb ), the luma coding block size log2CbSize, the prediction partition mode PartMode, the array verEdgeFlags, and the variable edgeType set equal to EDGE\_VER as inputs, and the modified array verEdgeFlags as output.

– The step 5 of horizontal edge filtering in subclause 8.7.2.1 is replaced by

1. The derivation process of prediction block boundary specified in subclause H. is invoked with the luma location ( xCb, yCb ), the luma coding block size log2CbSize, the prediction partition mode PartMode, the array horEdgeFlags, and the variable edgeType set equal to EDGE\_HOR as inputs, and the modified array horEdgeFlags as output.
   * 1. 1. Derivation process of prediction block boundary

Inputs to this process are:

– a luma location ( xCb, yCb ) 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 luma coding block size,

– a prediction partition mode PartMode,

– a two-dimensional (nCbS)x(nCbS) array edgeFlags,

– a variable edgeType specifying whether a vertical (EDGE\_VER) or a horizontal (EDGE\_HOR) edge is filtered.

Output of this process is the modified two-dimensional (nCbS)x(nCbS) array edgeFlags.

Depending on the value of PartMode, the following applies:

– If PartMode is equal to PART\_2Nx2N, the derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), prediction block width 1<<log2CbSize, prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_2NxN, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<log2CbSize, prediction block height 1<<log2CbSize-1, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 1<<log2CbSize-1), prediction block width 1<<log2CbSize, prediction block height 1<<(log2CbSize-1), the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_Nx2N, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<(log2CbSize-1), prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (1<<log2CbSize-1, 0), prediction block width 1<<log2CbSize-1, prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_NxN, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<log2CbSize-1, prediction block height 1<<log2CbSize-1, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (1<<log2CbSize-1, 0), prediction block width 1<<log2CbSize-1, prediction block height 1<<log2CbSize-1, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    3. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 1<<log2CbSize-1), prediction block width 1<<log2CbSize-1, prediction block height 1<<log2CbSize-1, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    4. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (1<<log2CbSize-1, 1<<log2CbSize-1), prediction block width 1<<log2CbSize-1, prediction block height 1<<log2CbSize-1, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_2NxnU, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<log2CbSize, prediction block height 1<<log2CbSize-2, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 1<<log2CbSize-2), prediction block width 1<<log2CbSize, prediction block height 3\*(1<<log2CbSize-2), the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_2NxnD, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<log2CbSize, prediction block height 1<<log2CbSize-2, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 1<<log2CbSize-2), prediction block width 1<<log2CbSize, prediction block height 3\*(1<<log2CbSize-2), the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_2NxnL, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 1<<log2CbSize-2, prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (1<<log2CbSize-2, 0), prediction block width 3\*(1<<log2CbSize-2), prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.

– Otherwise, if the PartMode is equal to PART\_2NxnR, the following ordered steps apply:

* + 1. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (0, 0), prediction block width 3\*(1<<log2CbSize-2), prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
    2. The derivation process defined in H.8.7.2.6 is invoked with luma location ( xCb, yCb ), luma location (xB1, yB1) set to (3\*(1<<log2CbSize-2), 0), prediction block width 1<<log2CbSize-2, prediction block height 1<<log2CbSize, the two-dimensional (nCbS)x(nCbS) array edgeFlags, and the variable edgeType as inputs, and the modified array edgeFlags as output.
       1. Derivation process of sub prediction block boundary within one prediction block

Inputs to this process are:

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

– a luma location ( xBl, yBl ) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nPbW specifying the width of the current luma prediction block,

– a variable nPbH specifying the width of the current luma prediction block,

– a two-dimensional (nCbS)x(nCbS) array edgeFlags,

– a variable edgeType specifying whether a vertical (EDGE\_VER) or a horizontal (EDGE\_HOR) edge is filtered.

Output of this process is the modified two-dimensional (nCbS)x(nCbS) array edgeFlags.

The variable nSbW is set equal to nPbW, and the variable nSbH is set equal to nPbH.

If SubPbMotionFlag[xCb+xB1, yCb+yB1] is equal to 1, the following applies:

* 1. nSbW = nPbW / SubPbSize[ nuh\_layer\_id ] <= 1 ? nPbW : SubPbSize[ nuh\_layer\_id ] (‑173)
  2. nSbH = nPbH / SubPbSize[ nuh\_layer\_id ] <= 1 ? nPbH : SubPbSize[ nuh\_layer\_id ] (‑174)

Otherwise, if the VspModeFlag[xCb+xB1, yCb+yB1] is equal to 1, the following applies:

nSbW = VspSubBlkW[xCb+xB1][yCb+yB1]

nSbH = VspSubBlkH[xCb+xB1][yCb+yB1]

If edgeType is equal to EDGE\_VER, the following applies for j=0..nPbW/nSbW-1 and k=0..nPbH-1:

edgeFlags[ xCb+xB1+j\*nSbW ][ k ] = 1

Otherwise, If edgeType is equal to EDGE\_HOR, the following applies for j = 0..nPbH/nSbH-1 and k=0..nPbW-1 :

edgeFlags[k][yCb+yB1+j\*nSbH] = 1