* + - * 1. Inter-view residual prediction process

The process is only invoked if res\_pred\_flag is equal to 1.

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 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 variable nCS specifying the size of the current luma coding block,
* variables nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit,prediction list utilization flags, predFlagL0 and predFlagL1,
* reference indices refIdxL0 and refIdxL1,
* a (nPSW)x(nPSH) array predSamplesL of luma prediction samples,
* two (nPSW / 2)x(nPSH / 2) arrays predSamplesCb and predSamplesCr of chroma prediction samples.

Output of this process are:

* a variable refViewIdx specifying the view order index of the reference view disparity derivation,
* a modified version of the (nPSW)x(nPSH) array predSamplesL,
* a modified versions of the (nPSW / 2)x(nPSH / 2) arrays predSamplesCb and predSamplesCr.

The derivation process for a disparity vector as specified in subclause G.8.5.2.1.13 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 outputs are the view order index refViewIdx, the flag availableDV and the disparity vector mvDisp.

Let refResSamplesL be the (PicWidthInSamplesL)x(PicHeightInSamplesL) array of constructed luma residual samples for inter-coded coding units for the view component with ViewIdx equal to refViewIdx. Let refResSamplesCb and refResSamplesCr be the (PicWidthInSamplesL / 2)x(PicHeightInSamplesL / 2) arrays of constructed Cb and Cr residual samples, respectively, for inter-coded coding units for the view component with ViewIdx equal to refViewIdx.

When the flag availableDV is equal to 0 the whole decoding process of this sub-clause terminates.

The variable log2resPredDenom is set equal to 0 and the following ordered steps apply.

* 1. When predFlagL0 is equal to 1 and ViewIdx is not equal to the view order index of RefPicListL0[ refIdxL0 ], log2resPredDenom is set equal to log2resPredDenom + 1.
  2. When predFlagL1 is equal to 1 and ViewIdx is not equal to the view order index of RefPicListL1[ refIdxL1 ], log2resPredDenom is set equal to log2resPredDenom + 1.
  3. The variable log2MaxResPredDenom is derived by
     1. log2MaxResPredDenom = ( ( predFlagL0 = = 1 ) && ( predFlagL1 == 1 ) ? 1 : 0 ) (G‑)

When log2resPredDenom is greater than log2MaxResPredDenom the whole decoding process of this sub-clause terminates.

For y proceeding over the values 0..(nPSH – 1) and x proceeding over the values 0..(nPSW – 1), the following ordered steps apply.

* 1. The variables xR0, xR1, w0, and w1 are derived by
     + 1. xR0 = Clip3( 0, PicWidthInSamplesL – 1, xP + x + (mvDisp[ 0 ] >> 2 ) ) (G‑189)  
          xR1 = Clip3( 0, PicWidthInSamplesL – 1, xP + x + (mvDisp[ 0 ] >> 2 ) + 1 ) (G‑190)
       2. yR0 = Clip3( 0, PicHeightInSamplesL – 1, yP + y + (mvDisp[ 1 ] >> 2 ) ) (G‑189)  
          yR1 = Clip3( 0, PicHeightInSamplesL – 1, yP + y + (mvDisp[ 1 ] >> 2 ) + 1 ) (G‑190)  
          w0 = 4 – mvDisp[0] + ( ( mvDisp[0] >> 2 ) << 2 ) (G‑191)  
          w1 = mvDisp[0] − ( ( mvDisp[0] >> 2 ) << 2 ) (G‑192)
       3. w2 = 4 – mvDisp[1] + ( ( mvDisp[1] >> 2 ) << 2 ) (G‑191)  
          w3 = mvDisp[1] − ( ( mvDisp[1] >> 2 ) << 2 ) (G‑192)
       4. ~~xR = Clip3( 0, PicWidthInSamples~~~~L~~~~– 1, xP + x + (mvDisp[ 0 ] >> 2 ) ) (G‑189)~~~~yR = Clip3( 0, PicHeightInSamples~~~~L~~~~– 1, yP + y + (mvDisp[ 1 ] >> 2 ) ) (G‑189)~~
  2. The sample predSamplesL[ x, y ] is modified by
     + 1. deltaL1 = ( w0 \* refResSamplesL[ xR0, yR0 ] + w1 \* refResSamplesL[ xR1, yR0 ] + 4 ) >> 3 (G‑193)
       2. deltaL2 = ( w0 \* refResSamplesL[ xR0, yR1 ] + w1 \* refResSamplesL[ xR1, yR1 ] + 4 ) >> 3 (G‑193)
       3. deltaL3 = ( w2 \* deltaL1 + w3 \* deltaL2 + 4 ) >> 3 (G‑193)  
          predSamplesL[ x, y ] = predSamplesL[ x, y ] + ( deltaL3 >> log2resPredDenom ) (G‑194)

For y proceeding over the values 0..(nPSH / 2 – 1) and x proceeding over the values 0..(nPSW / 2 – 1), the following ordered steps are specified:

* 1. The variables xR0, xR1, w0, and w1 are derived by
     + 1. xR0 = Clip3( 0, PicWidthInSamplesL / 2 – 1, xP / 2 + x + (mvDisp[0] >> 3 ) ) (G‑195)  
          xR1 = Clip3( 0, PicWidthInSamplesL / 2 – 1, xP / 2 + x + (mvDisp[0] >> 3 ) + 1 ) (G‑196)
       2. yR0 = Clip3( 0, PicHeightInSamplesL / 2 – 1, yP / 2 + y + (mvDisp[1] >> 3 ) ) (G‑195)  
          yR1 = Clip3( 0, PicHeightInSamplesL / 2 – 1, yP / 2 + y + (mvDisp[1] >> 3 ) + 1 ) (G‑196)  
          w0 = 8 – mvDisp[0] + ( (mvDisp[0] >> 3 ) << 3 ) (G‑197)  
          w1 = mvDisp[0] − ( (mvDisp[0] >> 3 ) << 3 ) (G‑198)
       3. w2 = 8 – mvDisp[1] + ( (mvDisp[1] >> 3 ) << 3 ) (G‑197)  
          w3 = mvDisp[1] − ( (mvDisp[1] >> 3 ) << 3 ) (G‑198)
  2. The sample predSamplesCb[ x, y ] is modified by
     + 1. deltaCb1 = ( w0 \* refResSamplesCb[ xR0, yR0] + w1 \* refResSamplesCb[ xR1, yR0 ] + 8 ) >> 4 (G‑199)

deltaCb2 = ( w0 \* refResSamplesCb[ xR0, yR1 ] + w1 \* refResSamplesCb[ xR1, yR1 ] + 8 ) >> 4 (G‑199)  
deltaCb3 = ( w2 \* deltaCb1 + w3 \* deltaCb2 + 8 ) >> 4 (G‑199)  
predSamplesCb[ x, y ] = predSamplesCb[ x, y ] + ( deltaCb3  >> log2resPredDenom ) (G‑200)

* 1. The sample predSamplesCr[ x, y ] is modified by
     + 1. deltaCr1 = ( w0 \* refResSamplesCr[ xR0, yR0 ] + w1 \* refResSamplesCr[ xR1, yR0 ] + 8 ) >> 4 (G‑201)
       2. deltaCr2 = ( w0 \* refResSamplesCr[ xR0, yR1 ] + w1 \* refResSamplesCr[ xR1, yR1] + 8 ) >> 4 (G‑201)  
          deltaCr3 = ( w0 \* deltaCr2+ w1 \* deltaCr3] + 8 ) >> 4 (G‑201)  
          predSamplesCr[ x, y ] = predSamplesCr[ x, y ] + ( deltaCr3  >> log2resPredDenom ) (G‑202)