|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_one\_bit** | u(1) |
| **num\_additional\_layer\_operation\_points** | u(8) |
| **num\_additional\_profile\_level\_sets\_minus1** | u(8) |
| for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| // mapping of layer ID to scalability dimension IDs |  |
| **reserved\_zero\_4bits\_num\_types**[ i ] | u(4) |
| **reserved\_zero\_4bits\_type**[ i ] | u(4) |
| **view\_id[** i **]** | u(8) |
| if ( i>0 ) |  |
| **num\_direct\_ref\_layers**[ i ] | u(6) |
| for( j = 0; j < num\_direct\_ref\_layers[ i ] && i; j++ ) |  |
| **ref\_layer\_id**[ i ][ j ] | u(6) |
| } |  |
| for( i = 0; i <= num\_additional\_profile\_level\_sets\_minus1; i++ ) |  |
| profile\_tier\_level( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| for( i = 1; i <= num\_additional\_layer\_operation\_points; i++ ) { |  |
| op\_point( i ) |  |
| if (num\_additional\_profile\_level\_sets) |  |
| **profile\_level\_idx**[ i ] |  |
| } |  |
| for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| if ( (i ! = 0)  && !( i % 2 ) ) { |  |
| **multi\_view\_mv\_pred\_flag**[ i ] | u(1) |
| **multi\_view\_residual\_pred\_flag**[ i ] | u(1) |
| } |  |
| if ( i % 2 ) { |  |
| **enable\_dmm\_flag**[ i ] | u(1) |
| **use\_mvi\_flag**[ i ] | u(1) |
| **lim\_qt\_pred\_flag**[ i ] | u(1) |
| **dlt\_flag**[ i ] | u(1) |
| if( dlt\_flag[ i ] ) { |  |
| **num\_depth\_values\_in\_dlt**[ i ] | ue(v) |
| for ( j = 0; j < num\_depth\_values\_in\_dlt ; j++) { |  |
| **dlt\_depth\_value**[ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| **enable\_dv\_constraint** | u(1) |
| } |  |

* + - * 1. Derivation process for a temporal inter-view motion vector predictor candidate

This process is not invoked when multi\_view\_mv\_pred\_flag 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 reference index refIdxLX (with X being equal to 0 or 1) specifying a reference picture in the reference picture list RefPicListLX,
* a disparity vector mvDisp specifying the disparity to the reference block,
* a view index refViewIdx specifying a reference view.

Outputs of this process are:

* a flag availableFlagLXInterView specifying whether the inter-view motion vector candidate is available,
* a motion vector candidate mvLXInterView (if availableFlagLXInterView is equal to 1).

The reference layer luma location ( xRef, yRef ) is derived by

xRef = Clip3( 0, PicWidthInSamplesL – 1, xP + ( ( nPSW – 1 ) >> 1 ) + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) ) (G‑)

if(**enable\_dv\_constraint == 1)**

**{**  
yRef = Clip3( 0, PicHeightInSamplesL – 1, yP + ( ( nPSH – 1 ) >> 1 ) + ( ( mvDisp[ 1 ] + 2 ) >> 2 ) ) (G‑)

}

Else

{

yRef = yP + ( ( nPSH – 1 ) >> 1 ) (G‑110)

}

Let refPU be the coding unit that covers the luma location ( xRef, yRef ) in the view component with ViewIdx equal to refViewIdx.

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

When the variable PredMode for the coding unit refCU is equal to MODE\_SKIP or MODE\_INTER, the following ordered steps apply.

* 1. For Y being replaced by 0 and 1, the following applies.
     + The variable refPredFlagLY is set equal to the variable predFlagLY of the prediction unit refPU.
     + The variable refMvLY is set equal to mvLY of the prediction unit refPU.
     + The variable refRefPicListLY is set equal to the reference picture list RefPicListLY of refPU.
  2. When refPredFlagL0 is equal to 1 and the picture order count of the picture refRefPicListL0[ refRefIdxL0 ] is equal to the picture order count of the picture RefPicListLX[ refIdxLX ], the flag availableFlagLXInterView is set equal to 1 and the motion vector mvLXInterView is derived by
     + 1. mvLXInterView[ 0 ] = refMvL0[ 0 ] (G‑111)  
          mvLXInterView[ 1 ] = refMvL0[ 1 ] (G‑112)
  3. When availableFlagLXInterView is equal to 0 and refPredFlagL1 is equal to 1 and the picture order count of the picture refRefPicListL1[ refRefIdxL1 ] is equal to the picture order count of the picture RefPicListLX[ refIdxLX ], the flag availableFlagLXInterView is set equal to 1 and the motion vector mvLXInterView is derived by
     + 1. mvLXInterView[ 0 ] = refMvL1[ 0 ] (G‑113)  
          mvLXInterView[ 1 ] = refMvL1[ 1 ] (G‑114)
          1. Derivation process for a temporal inter-view motion vector merging candidate

This process is not invoked when multi\_view\_mv\_pred\_flag 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 disparity vector mvDisp,
* a prediction list indication X,
* a reference view index refViewIdx.

Outputs of this process are:

* a flag availableFlagLXInterView specifying whether the inter-view motion vector candidate is available,
* a 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, and refIdxLX is set equal to −1.

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 reference layer luma location ( xRef, yRef ) is derived by

xRef = Clip3( 0, PicWidthInSamplesL – 1, xP + ( ( nPSW – 1 ) >> 1 ) + ( ( mvDisp[ 0 ] + 2 ) >> 2 ) ) (G‑127)

if(**enable\_dv\_constraint == 1)**

**{**  
yRef = Clip3( 0, PicHeightInSamplesL – 1, yP + ( ( nPSH – 1 ) >> 1 ) + ( ( mvDisp[ 1 ] + 2 ) >> 2 )) (G‑128)

}

Else

{

yRef = yP + ( ( nPSH – 1 ) >> 1 ) (G‑128)

}

Let refCU be the coding unit that covers the luma location ( xRef, yRef ) in the view component with ViewIdx equal to refViewIdx.

When the variable PredMode for the coding unit refCU is equal to MODE\_SKIP or MODE\_INTER, the following ordered steps apply:

* 1. The variable refPredFlagLX is set equal to the prediction utilization flag predFlagLX of the prediction unit refPU.
  2. The variable refRefIdxLX, is set equal to the reference index refIdxLX of the prediction unit refPU.
  3. The variable refMvLX is set equal to the motion vector mvLX of the prediction unit refPU.
  4. The variable refRefPicListLX, is set equal to the reference picture list RefPicListLX of the view component with ViewIdx equal to refViewIdx.
  5. When refPredFlagLX is equal to 1, the following applies for each i from 0 to num\_ref\_idx\_lX\_active\_minus1, inclusive
     + When availableFlagLXInterView is equal to 0, and the picture order count of the picture refRefPicListLX[ refRefIdxLX ] is equal to the picture order count of the picture RefPicListLX[ i ], the flag availableFlagLXInterView is set equal to 1 and the following applies.
       - 1. mvLXInterView[ 0 ] = refMvLX[ 0 ] (G‑)  
            mvLXInterView[ 1 ] = refMvLX[ 1 ] (G‑)  
            refIdxLX = i (G‑)
         2. 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.

if(**enable\_dv\_constraint == 0)**

* + - 1. **{**

The variables xR, yR are derived by

xR = Clip3( 0, PicWidthInSamplesL – 1, xP + x + ((mvDisp[0]+2) >> 2 ) ) (G‑123)

yR = Clip3( 0, PicHeightInSamplesL – 1, yP + x + ((mvDisp[1]+2) >> 2 ) ) (G‑123)

1. The sample predSamplesL[ x, y ] is modified by

deltaL = refResSamplesL[ xR, yR ] (G‑127)  
predSamplesL[ x, y ] = predSamplesL[ x, y ] + deltaL (G‑128)

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 xR, yR are derived by

xR = Clip3( 0, PicWidthInSamplesL / 2 – 1, xP / 2 + x + ((mvDisp[0] +4)>> 3 ) ) (G‑129)

yR = Clip3( 0, PicHeightInSamplesL / 2 – 1, yP / 2 + y + ((mvDisp[1] +4)>> 3 ) ) (G‑129)

1. The sample predSamplesCb[ x, y ] is modified by

deltaCb = refResSamplesCb[ xR, yR ] (G‑133)  
predSamplesCb[ x, y ] = predSamplesCb[ x, y ] + deltaCb (G‑134)

1. The sample predSamplesCr[ x, y ] is modified by
   * + 1. deltaCr = refResSamplesCr[ xR, yR ] (G‑135)  
          predSamplesCr[ x, y ] = predSamplesCr[ x, y ] + deltaCr (G‑136)

}

Else

{

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‑)  
          xR1 = Clip3( 0, PicWidthInSamplesL – 1, xP + x + (mvDisp[ 0 ] >> 2 ) + 1 ) (G‑)  
          w0 = 4 – mvDisp[0] + ( ( mvDisp[0] >> 2 ) << 2 ) (G‑)  
          w1 = mvDisp[0] − ( ( mvDisp[0] >> 2 ) << 2 ) (G‑)
       2. xR = Clip3( 0, PicWidthInSamplesL – 1, xP + x + (mvDisp[ 0 ] >> 2 ) ) (G‑)  
          yR = Clip3( 0, PicHeightInSamplesL – 1, yP + y + (mvDisp[ 1 ] >> 2 ) ) (G‑)
  2. The sample predSamplesL[ x, y ] is modified by
     + 1. deltaL = ( w0 \* refResSamplesL[ xR0, y ] + w1 \* refResSamplesL[ xR1, y ] + 4 ) >> 3 (G‑)  
          predSamplesL[ x, y ] = predSamplesL[ x, y ] + ( deltaL >> log2resPredDenom ) (G‑)

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‑)  
          xR1 = Clip3( 0, PicWidthInSamplesL / 2 – 1, xP / 2 + x + (mvDisp[0] >> 3 ) + 1 ) (G‑)  
          w0 = 8 – mvDisp[0] + ( (mvDisp[0] >> 3 ) << 3 ) (G‑)  
          w1 = mvDisp[0] − ( (mvDisp[0] >> 3 ) << 3 ) (G‑)
  2. The sample predSamplesCb[ x, y ] is modified by
     + 1. deltaCb = ( w0 \* refResSamplesCb[ xR0, y ] + w1 \* refResSamplesCb[ xR1, y ] + 8 ) >> 4 (G‑)  
          predSamplesCb[ x, y ] = predSamplesCb[ x, y ] + ( deltaCb  >> log2resPredDenom ) (G‑)
  3. The sample predSamplesCr[ x, y ] is modified by
     + 1. deltaCr = ( w0 \* refResSamplesCr[ xR0, y ] + w1 \* refResSamplesCr[ xR1, y ] + 8 ) >> 4 (G‑)  
          predSamplesCr[ x, y ] = predSamplesCr[ x, y ] + ( deltaCr  >> log2resPredDenom ) (G‑)

}