|  |  |
| --- | --- |
| 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) { |  |
| **multi\_view\_mv\_pred\_flag**[ i ] | u(1) |
| **multi\_view\_residual\_pred\_flag**[ i ] | u(1) |
| } |  |
| if ( i%1) { |  |
| **enable\_dmm\_flag**[ i ] | u(1) |
| **~~use\_mvi\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| } |  |
| **}** |  |
| } |  |

G.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in subclause 8.5 apply with the following modification:

– ~~If~~ ~~use\_mvi\_flag is equal to 1 and ( PredMode is equal to MODE\_SKIP or PredMode is equal to MODE\_INTER and merge\_flag[ xP ][ yP ] is equal to 1 ) and merge\_idx[ xP ][ yP ] is equal to 0, the following ordered steps apply:~~

1. ~~The variable TextureModeDepth[ x0 >> Log2MinCbSize ][ y0 >> Log2MinCbSize ] is set equal to the current value of the variable cbDepth.~~
2. ~~The syntax elements split\_coding\_unit\_flag[  ][  ] as well as the motion vectors and reference indices are inherited from the co-located region of the corresponding texture picture.~~

– All invocations of the process specified in subclause 8.5.1 are replaced with invocations of the process specified in subclause .

G.8.5.2.1.1 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 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, and the prediction utilization flags predFlagL0 and predFlagL1 are derived as specified by the following ordered steps:

1. The derivation process for spatial merge candidates as specified in subclause 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 8.5.2.1.7 is invoked with luma location ( xP, yP ), the width and the height of the luma prediction block nPbW and nPbH, and refIdxLX as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector mvLXCol. The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.

availableFlagCol = availableFlagL0Col | | availableFlagL1Col (G‑73)   
predFlagLXCol = availableFlagLXCol (G‑74)

1. Depending on multi\_view\_mv\_pred\_flag, the following applies.

– If multi\_view\_mv\_pred\_flag is equal to 0, the variable availableFlagInterView is set equal to 0.

– Otherwise (multi\_view\_mv\_pred\_flag is equal to 1), the derivation process for the inter-view merge candidate as specified in subclause G.8.5.2.1.9 is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH, and the reference view identifier refViewIdx set equal to 0 as the inputs and the outputs are the flag availableFlagInterView, the prediction utilization flags predFlagL0InterView and predFlagL1InterView, the reference indices refIdxL0InterView and refIdxL1InterView, and the motion vectors mvL0InterView and mvL1InterView.

1. Depending on DepthFlag, the following applies.

– If DepthFlag is equal to 0, the variable availableFlagTexture is set equal to 0.

– Otherwise (DepthFlag is equal to 1), the derivation process for the texture merging candidate as specified in subclause 8.5.2.x.x is invoked with the luma location ( xP, yP ), the variables nPSW and nPSH as the inputs and the outputs are the flag availableFlagTexture, the prediction utilization flags predFlagL0Texture and predFlagL1Texture, the reference indices refIdxL0Texture and refIdxL1Texture, and the motion vectors mvL0Texture and mvL1Texture.

1. The merge candidate list, mergeCandList, is constructed as specified by the following ordered steps:.
   1. The variable numMergeCand is set equal to 0.
   2. When availableFlagInterView is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to InterView and the variable numMergeCand is increased by 1.
   3. When availableFlagTexture is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to Texture and the variable numMergeCand is increased by 1.
   4. When availableFlagA1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A1 and the variable numMergeCand is increased by 1.
   5. When availableFlagB1 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B1 and the variable numMergeCand is increased by 1.
   6. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0 and the variable numMergeCand is increased by 1.
   7. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0 and the variable numMergeCand is increased by 1.
   8. When availableFlagB2 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B2 and the variable numMergeCand is increased by 1.
   9. When availableFlagCol is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to Col and the variable numMergeCand is increased by 1.
2. The variable numOrigMergeCand is set equal to numMergeCand .
3. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause 8.5.2.1.3 is invoked with 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, 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.
4. 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.
5. The variable MergeIdx is derived as follows.

* ~~If use\_mvi\_flag is equal to 0,~~ MergeIdx is set equal to merge\_idx[ xP][ yP ].
* ~~Otherwise (use\_mvi\_flag is equal to 1), MergeIdx is set equal to merge\_idx[ xP][ yP ] - 1.~~

1. The following assignments are made with N being the candidate at position MergeIdx in the merging candidate list mergeCandList ( N = mergeCandList[ MergeIdx ] ) and X being replaced by 0 or 1:

mvLX[ 0 ] = mvLXN[ 0 ] (G‑75)

mvLX[ 1 ] = mvLXN[ 1 ] (G‑76)

refIdxLX = refIdxLXN (G‑77)

predFlagLX = predFlagLXN (G‑78)

1. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nPbW + nPbH ) is equal to 12, the following applies.

refIdxL1 = −1 (G‑79)

predFlagL1 = 0 (G‑80)

1. When the MergeIdx is not equal to 0, the following applies.

IvpMvFlagL0[ xP , yP ] = 0 (G‑81)

IvpMvFlagL1[ xP , yP ] = 0 (G‑82)

G.8.5.2.x.x Derivation process for the texture merging candidate

This process is not invoked when depthflag 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,

Outputs of this process are:

– a flag availableFlagTexture specifying whether the texture merging candidate is available,

– the prediction utilization flags predFlagL0Texture and predFlagL1Texture,

– the reference indices refIdxL0Texture and refIdxL1Texture (if availableFlagTexture is equal to 1),

– the motion vectors mvL0Texture and mvL1Texture (if availableFlagTexture is equal to 1).

The variable availableFlagTexture is set equal to 0. The variables predFlagL0Texture and predFlagL1Texture are set equal to 0. The variables refIdxL0Texture and refIdxL1Texture are set equal to −1. Both components of the motion vectors mvL0Texture and mvL1Texture are set equal to 0. The variables TextureMvFlagL0[ xP, yP ] and TextureMvFlagL1[ xP, yP ] are set equal to 0.

The texture luma location ( xRef, yRef ) is derived by

xRef = Clip3( 0, PicWidthInSamplesL – 1, xP + ( ( nPSW – 1 ) >> 1 ) ) (G‑)  
yRef = Clip3( 0, PicHeightInSamplesL – 1, yP + ( ( nPSH – 1 ) >> 1 ) ) (G‑)

Let refCU be the coding unit that covers the luma location ( xRef, yRef ) in the texture picture with ViewIdx equal to current ViewIdx.

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

1. Let refPredFlagLY, with Y being replaced by 0 and 1, be the variables predFlagLY for the prediction unit refPU. Let refRefIdxLY, with Y being replaced by 0 and 1, be the variables refIdxLY for the prediction unit refPU. Let refMvLY, with Y being replaced by 0 and 1, be the variables mvLY for the prediction unit refPU. Let refRefPicListLY, with Y being replaced by 0 and 1, be the reference picture list RefPicListLY for the prediction unit refPU in the texture picture with ViewIdx equal to current ViewIdx.
2. When refPredFlagL0 is equal to 1, the following apply,

mvL0Texture[ 0 ] = refMvL0[ 0 ] (G‑)  
mvL0Texture[ 1 ] = refMvL0[ 1 ] (G‑)  
refIdxL0 = refRefIdxL0 (G‑)  
predFlagL0Texture = 1 (G‑)

availableFlagTexture = 1 (G‑)

1. When the current slice is a B slices and refPredFlagL1 is equal to 1, the following apply,

mvL1Texture[ 0 ] = refMvL1[ 0 ] (G‑)  
mvL1Texture[ 1 ] = refMvL1[ 1 ] (G‑)  
refIdxL1 = refRefIdxL1 (G‑)  
predFlagL1Texture = 1 (G‑)

availableFlagTexture = 1 (G‑)