J.7.3.3.4 Slice header in 3D-AVC extension syntax

|  |  |  |
| --- | --- | --- |
| if ( nal\_unit\_type  = =  21  && ( slice\_type != I && slice\_type != SI )) { |  |  |
| if( DepthFlag ) |  |  |
| **depth\_weighted\_pred\_flag** | 2 | u(1) |
| if( ! DepthFlag && avc\_3d\_extension\_flag ) { |  |  |
| **dmvp\_flag** | 2 | u(1) |
| if(!DepthFlag && dmvp\_flag && (NumDepthViews  = =  0  | | ( ViewCompOrder(0,view\_idx ) < ViewCompOrder( 1, view\_idx )) { |  |  |
| **global\_depth** | 2 | ue(v) |
| **temporal\_mv\_flag** | 2 | u(1) |
| } |  |  |
| if( seq\_view\_synthesis\_flag ) |  |  |
| **slice\_vsp\_flag** | 2 | u(1) |
| } |  |  |
| if ( 3dv\_acquisition\_idc != 1 &&   ( depth\_weighted\_pred\_flag | | dmvp\_flag ) ) |  |  |
| **dps\_id** | 2 | ue(v) |
| } |  |  |

J.7.4.3.4 Slice header in 3D-AVC semantics

J‑ – Respective syntax elements for pre\_slice\_header\_src, pre\_ref\_lists\_src, pre\_pred\_weight\_table\_src and pre\_dec\_ref\_pic\_marking\_src

|  |  |
| --- | --- |
| **Prediction indication syntax element** | **Respective syntax elements** |
| pre\_slice\_header\_src | colour\_plane\_id, frame\_num, field\_pic\_flag, bottom\_field\_flag, idr\_pic\_id, pic\_order\_cnt\_lsb, delta\_pic\_order\_cnt\_bottom, delta\_pic\_order\_cnt[ 0 ], delta\_pic\_order\_cnt[ 1 ], redundant\_pic\_cnt, direct\_spatial\_mv\_pred\_flag, cabac\_init\_idc, sp\_for\_switch\_flag, slice\_qs\_delta, disable\_deblocking\_filter\_idc, slice\_alpha\_c0\_offset\_div2, slice\_beta\_offset\_div2, slice\_group\_change\_cycle, depth\_weighted\_pred\_flag, dmvp\_flag, global\_depth, temporal\_mv\_flag, slice\_vsp\_flag, dps\_id |
| pre\_ref\_lists\_src | num\_ref\_idx\_active\_override\_flag, num\_ref\_idx\_l0\_active\_minus1, num\_ref\_idx\_l1\_active\_minus1 and reference picture list modification syntax table |
| pre\_pred\_weight\_table\_src | pred\_weight\_table( ) syntax structure |
| pre\_dec\_ref\_pic\_marking\_src | dec\_ref\_pic\_marking( ) syntax structure |

**dmvp\_flag** is used in the decoding process for inter prediction, inter-view prediction, view synthesis prediction and adaptive luminance compensation as specified in subclause . When not present, termporal\_mv\_flag is inferred to be equal to 0.

**global\_depth** specifies a global depth value to be used for the motion vector prediction in the texture view components.

**temporal\_mv\_flag** equal to 0 specifies that temporal motion vectors in previously coded frames are not used for the motion vector prediction in the texture view components. termporal\_mv\_flag equal to 1 specifies that temporal motion vectors in previously coded frames are used for the motion vector prediction in the texture view components. When not present, termporal\_mv\_flag is inferred to be equal to 0.

J.8.2 3D-AVC inter prediction, inter-view prediction, view synthesis prediction and adaptive luminance compensation

This process is invoked when decoding P and B macroblock types and when nal\_unit\_type is equal to 21.

Outputs of this process are Inter prediction samples for the current macroblock that are a 16x16 array predL of luma samples and when ChromaArrayType is not equal to 0 two (MbWidthC)x(MbHeightC) arrays predCb and predCr of chroma samples, one for each of the chroma components Cb and Cr.

When DepthFlag is equal to 0, dmvp\_flag is equal to 1, and the depth view components are available, the variables DepthRefPicList0, DepthRefPicList1 for B slices, and DepthCurrPic are specified as follows. The variable DepthRefPicList0 is specified to consist of the depth view components of the view component pairs for which the texture view components are in RefPicList0 in the order that RefPicList0[ i ] and DepthRefPicList0[ i ] form a view component pair for any value of i = 0.. num\_ref\_idx\_l0\_active\_minus1. The variable DepthRefPicList1 is specified for B slices to consist of the depth view components of the view component pairs for which the texture view components are in RefPicList1 in the order that RefPicList1[ i ] and DepthRefPicList1[ i ] form a view component pair for any value of i = 0.. num\_ref\_idx\_l1\_active\_minus1. The variable DepthCurrPic is specified to be the decoded sample array of the depth view component of the view component pair for which the texture view component is the current texture view component.

The partitioning of a macroblock is specified by mb\_type. Each macroblock partition is referred to by mbPartIdx. When the macroblock partitioning consists of partitions that are equal to sub-macroblocks, each sub-macroblock can be further partitioned into sub-macroblock partitions as specified by sub\_mb\_type[ mbPartIdx ]. Each sub-macroblock partition is referred to by subMbPartIdx. When the macroblock partitioning does not consist of sub-macroblocks, subMbPartIdx is set equal to 0.

The following steps are specified for each macroblock partition or for each sub-macroblock partition.

The functions MbPartWidth( ), MbPartHeight( ), SubMbPartWidth( ), and SubMbPartHeight( ) describing the width and height of macroblock partitions and sub-macroblock partitions are specified in Tables , , , and .

Diparity vecor defaultDV is set to as follows:

log2Div = BitDepthY + 6  
srcIndex = ViewIdTo3DVAcquisitionParamIndex( srcViewId )  
refIndex = ViewIdTo3DVAcquisitionParamIndex( refViewId )  
dispVal = ( NdrInverse[ gloval\_depth ] \* DisparityScale[ dps\_id ][ srcIndex ][ refIndex ] +   
 ( DisparityOffset[ dps\_id ][ srcIndex ][ refIndex ] << BitDepthY ) +   
 ( 1 << ( log2Div – 1 ) ) ) >> log2Div When nal\_unit\_type is equal to 21, DepthFlag is equal to 0, TextureFirstFlag is equal to 1, InterViewRefAvailable is equal to 1 and dmvp\_flag is equal to 1, DvMBX is set equal to defaultDV when CurrMbAddr is equal to first\_mb\_in\_slice, and subclause is invoked.

The range of the macroblock partition index mbPartIdx is derived as follows:

– If mb\_type is equal to B\_Skip or B\_Direct\_16x16 or if both mb\_type is equal to P\_Skip and MbVSSkipFlag is equal to 1, mbPartIdx proceeds over values 0..3.

– Otherwise (mb\_type is not equal to B\_Skip or B\_Direct\_16x16), mbPartIdx proceeds over values 0..NumMbPart( mb\_type ) − 1.

For each value of mbPartIdx, the variables partWidth and partHeight for each macroblock partition or sub-macroblock partition in the macroblock are derived as follows:

– If mb\_type is not equal to P\_8x8, P\_8x8ref0, B\_Skip, B\_Direct\_16x16, or B\_8x8, subMbPartIdx is set equal to 0 and the following applies:

– If mb\_type is equal to P\_Skip and MbVSSkipFlag is equal to 1, partWidth and partHeight are derived as:

partWidth = 8 (J-6)

partHeight = 8 (J-7)

– Otherwise, partWidth and partHeight are derived as:

partWidth = MbPartWidth( mb\_type ) (J-8)

partHeight = MbPartHeight( mb\_type ) (J-9)

– Otherwise, if mb\_type is equal to P\_8x8 or P\_8x8ref0, or mb\_type is equal to B\_8x8 and sub\_mb\_type[ mbPartIdx ] is not equal to B\_Direct\_8x8, subMbPartIdx proceeds over values 0..NumSubMbPart( sub\_mb\_type[ mbPartIdx ] ) − 1, and partWidth and partHeight are derived as:

partWidth = SubMbPartWidth( sub\_mb\_type[ mbPartIdx ] ) (J-10)

partHeight = SubMbPartHeight( sub\_mb\_type[ mbPartIdx ] ). (J-11)

– Otherwise (mb\_type is equal to B\_Skip or B\_Direct\_16x16, or mb\_type is equal to B\_8x8 and sub\_mb\_type[ mbPartIdx ] is equal to B\_Direct\_8x8), subMbPartIdx proceeds over values 0..3, and partWidth and partHeight are derived as:

partWidth = 4 (J-12)

partHeight = 4 (J-13)

When ChromaArrayType is not equal to 0, the variables partWidthC and partHeightC are derived as:

partWidthC = partWidth / SubWidthC (J-14)  
partHeightC = partHeight / SubHeightC (J-15)

Let the variable MvCnt be initially set equal to 0 before any invocation of clause , or for the macroblock.

The Inter prediction process for a macroblock partition mbPartIdx and a sub-macroblock partition subMbPartIdx consists of the following ordered steps:

1. The following applies:

– If nal\_unit\_type is equal to 21 and DepthFlag is equal to 0, the following applies:

– If mb\_alc\_skip\_flag is equal to 1 or mb\_alc\_flag is equal to 1, subclause is invoked.

– Otherwise, if dmvp\_flag is equal to 1, subclause is invoked.

– Otherwise, subclause 8.4.1 is invoked.

– Otherwise, the derivation process for motion vector components and reference indices as specified in clause  is invoked.

Inputs to the processes in subclauses , and are:

– a macroblock partition mbPartIdx,

– a sub-macroblock partition subMbPartIdx.

Outputs of the processes in subclauses , and are:

– luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1

– reference indices refIdxL0 and refIdxL1

– prediction list utilization flags predFlagL0 and predFlagL1

– the sub-macroblock partition motion vector count subMvCnt.

1. The variable MvCnt is incremented by subMvCnt.
2. When (weighted\_pred\_flag is equal to 1 and (slice\_type % 5) is equal to 0 or 3) or (weighted\_bipred\_idc is greater than 0 and (slice\_type % 5) is equal to 1), the following applies:

– If mb\_alc\_skip\_flag is equal to 1 or mb\_alc\_flag is equal to 1, subclause is invoked.

– Otherwise, the derivation process for prediction weights as specified in clause  is invoked.

Inputs to these processes in subclauses 8.4.3 and are:

– reference indices refIdxL0 and refIdxL1

– prediction list utilization flags predFlagL0 and predFlagL1

Outputs of these processes in subclauses 8.4.3 and are variables for weighted prediction logWDC, w0C, w1C, o0C, o1C with C being replaced by L and, when ChromaArrayType is not equal to 0, Cb and Cr.

1. When (nal\_unit\_type is equal to 21 and depth\_weighted\_pred\_flag is equal to 1 and (slice\_type % 5) is equal to 0 or 3) or (depth\_weighted\_bipred\_flag is equal to 1 and (slice\_type % 5) is equal to 1), the derivation process for prediction weights in depth-range-based weighted prediction in subclause is invoked.
2. The decoding process for Inter prediction samples as specified in clause  is invoked.

Inputs to this process are:

– a macroblock partition mbPartIdx,

– a sub-macroblock partition subMbPartIdx,

– variables specifying partition width and height for luma and chroma (if available), partWidth, partHeight, partWidthC (if available), and partHeightC (if available),

– luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– reference indices refIdxL0 and refIdxL1,

– prediction list utilization flags predFlagL0 and predFlagL1,

– variables for weighted prediction logWDC, w0C, w1C, o0C, o1C with C being replaced by L and, when ChromaArrayType is not equal to 0, Cb and Cr.

Outputs of this process are inter prediction samples (pred); which are a (partWidth)x(partHeight) array predPartL of prediction luma samples and when ChromaArrayType is not equal to 0 two (partWidthC)x(partHeightC) arrays predPartCr, and predPartCb of prediction chroma samples, one for each of the chroma components Cb and Cr.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made:

MvL0[ mbPartIdx ][ subMbPartIdx ] = mvL0 (J-16)

MvL1[ mbPartIdx ][ subMbPartIdx ] = mvL1 (J-17)

RefIdxL0[ mbPartIdx ] = refIdxL0 (J-18)

RefIdxL1[ mbPartIdx ] = refIdxL1 (J-19)

PredFlagL0[ mbPartIdx ] = predFlagL0 (J-20)

PredFlagL1[ mbPartIdx ] = predFlagL1 (J-21)

The location of the upper-left sample of the macroblock partition relative to the upper-left sample of the macroblock is derived by invoking the inverse macroblock partition scanning process as described in clause  with mbPartIdx as the input and ( xP, yP ) as the output.

The location of the upper-left sample of the sub-macroblock partition relative to the upper-left sample of the macroblock partition is derived by invoking the inverse sub-macroblock partition scanning process as described in clause  with subMbPartIdx as the input and ( xS, yS ) as the output.

The macroblock prediction is formed by placing the macroblock or sub-macroblock partition prediction samples in their correct relative positions in the macroblock, as follows.

The variable predL[ xP + xS + x, yP + yS + y ] with x = 0..partWidth − 1, y = 0..partHeight − 1 is derived by:

predL[ xP + xS + x, yP + yS + y ] = predPartL[ x, y ] (J-22)

When ChromaArrayType is not equal to 0, the variable predC with x = 0..partWidthC − 1, y = 0..partHeightC − 1, and C in predC and predPartC being replaced by Cb or Cr is derived by:

predC[ xP / SubWidthC + xS / SubWidthC + x, yP / SubHeightC + yS / SubHeightC + y ] = predPartC[ x, y ]  
 (J-23)

J.8.2.1 Derivation process for motion vector components and reference indices

Inputs to this process are:

– a macroblock partition mbPartIdx,

– a sub-macroblock partition subMbPartIdx.

Outputs of this process are:

– luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– reference indices refIdxL0 and refIdxL1,

– prediction list utilization flags predFlagL0 and predFlagL1,

– a motion vector count variable subMvCnt.

For the derivation of the variables mvL0 and mvL1 as well as refIdxL0 and refIdxL1, the following applies:

– If mb\_type is equal to P\_Skip, the following applies:

– If MbVSSkipFlag is equal to 0, the following applies:

– If nal\_unit\_type is equal to 21 and DepthFlag is equal to 0 and dmvp\_flag is equal to 1, the depth-based derivation process for luma motion vectors for skipped macroblock in P and SP slices in subclause is invoked with the output being the luma motion vectors mvL0 and refernece indices refIdxL0, and predFlagL0 is set equal to 1.

– Otherwise (nal\_unit\_type is not equal to 21 or DepthFlag is equal to 1 or dmvp\_flag is equal to 0), the derivation process for luma motion vectors for skipped macroblock in P and SP slices in subclause 8.4.1.1 is invoked with the output being the luma motion vectors mvL0 and reference indices refIdxL0, and predFlagL0 is set equal to 1.

– Otherwise (MbVSSkipFlag is equal to 1), the derivation process for luma motion vectors for VSP skipped macroblock in P and SP slices in subclause is invoked with mbPartIdx as input and with the output being the luma motion vectors mvL0 and reference indices refIdxL0, and predFlagL0 is set equal to 1.

– mvL1 and refIdxL1 are marked as not available and predFlagL1 is set equal to 0. The motion vector count variable subMvCnt is set equal to 1.

– Otherwise, if mb\_type is equal to B\_Skip or B\_Direct\_16x16 or sub\_mb\_type[ mbPartIdx ] is equal to B\_Direct\_8x8, the following applies.

– The variable vspFlag is specified as follows:

vspFlag = !( ( mb\_type = = B\_Skip && MbVSSkipFlag = = 0 ) | |   
 ( ( mb\_type = = B\_Direct\_16x16 | | sub\_type[ mbPartIdx ] = = B\_Direct\_8x8 ) &&   
 !mb\_direct\_type\_flag ) ) (J-24)

– If vspFlag is equal to 0 and nal\_unit\_type is equal to 21 and DepthFlag is equal to 0 and dmvp\_flag is equal to 1, the depth-based derivation process for luma motion vectors for B\_Skip, B\_Direct\_16x16, and B\_Direct\_8x8 in B slices in subclause is invoked with mbPartIdx and subMbPartIdx as the input and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, the motion vector count variable subMvCnt, and the prediction utilization flags predFlagL0 and predFlagL1.

– Otherwise, if both of the following are true:

– vspFlag is equal to 0, and

– nal\_unit\_type is not equal to 21 or DepthFlag is equal to 1 or dmvp\_flag is equal to 0,

the derivation process for luma motion vectors for B\_Skip, B\_Direct\_16x16, and B\_Direct\_8x8 in B slices in subclause 8.4.1.2 is invoked with mbPartIdx and subMbPartIdx as the input and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, the motion vector count variable subMvCnt, and the prediction utilization flags predFlagL0 and predFlagL1.

– Otherwise (vspFlag is equal to 1), the derivation process in subclause is invoked with with mbPartIdx as input and with the output being the luma motion vectors mvL0 and mvL1 and reference indices refIdxL0 and refIdxL1.

– Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, refIdxLX, and in Pred\_LX and in the syntax elements ref\_idx\_lX and mvd\_lX, the following applies:

1. The variables refIdxLX and predFlagLX are derived as follows:

– If MbPartPredMode( mb\_type, mbPartIdx ) or SubMbPredMode( sub\_mb\_type[ mbPartIdx ] ) is equal to Pred\_LX or to BiPred,

refIdxLX = ref\_idx\_lX[ mbPartIdx ] (J-25)

predFlagLX = 1 (J-26)

– Otherwise, the variables refIdxLX and predFlagLX are specified by

refIdxLX = −1 (J-27)

predFlagLX = 0 (J-28)

1. The motion vector count variable subMvCnt is set equal to predFlagL0 + predFlagL1.
2. The variable currSubMbType is derived as follows:

– If the macroblock type is equal to B\_8x8, currSubMbType is set equal to sub\_mb\_type[ mbPartIdx ].

– Otherwise (the macroblock type is not equal to B\_8x8), currSubMbType is set equal to "na".

1. The following applies:

– If VspRefLXFlag[ mbPartIdx ] is equal to 0 or both VspRefLXFlag[ mbPartIdx ] is equal to 1 and bvsp\_flag\_lX[ mbPartIdx ] is equal to 0, the following applies:

– When predFlagLX is equal to 1 and DepthFlag is equal to 0 and dmvp\_flag is equal to 1, the derivation process for luma motion vector prediction in subclause  is invoked with mbPartIdx subMbPartIdx, refIdxLX, and currSubMbType as the inputs and the output being mvpLX.

– When predFlagLX is equal to 1 and either DepthFlag is equal to 1 or dmvp\_flag is equal to 0, the derivation process for luma motion vector prediction in subclause 8.4.1.3 is invoked with mbPartIdx subMbPartIdx, refIdxLX, and currSubMbType as the inputs and the output being mvpLX.

– The luma motion vectors are derived by

mvLX[ 0 ] = mvpLX[ 0 ] + mvd\_lX[ mbPartIdx ][ subMbPartIdx ][ 0 ] (J-29)

mvLX[ 1 ] = mvpLX[ 1 ] + mvd\_lX[ mbPartIdx ][ subMbPartIdx ][ 1 ] (J-30)

– Otherwise (VspRefLXFlag[ mbPartIdx ] is equal to 1 and bvsp\_flag\_lX[ mbPartIdx ] is equal to 1), the following apply:

the depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, ( textureX, textureY ) equal to the location of the top-left sample of macroblock partition mbPartIdx, tBlWidth equal to the width the macroblock partition mbPartIdx, tBlHeight equal to the height the macroblock partition mbPartIdx, srcViewId equal to view\_id and refViewId equal to the view\_id of refIdxLX as inputs and the output assigned to mvLX[ 0 ] and mvLX[ 1 ] is set equal to 0.

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

J.8.2.1.1 Depth-based disparity value derivation process

Inputs to this process are

– a decoded depth view component depthPic,

– the location (textureX, textureY) of the block in a texture view component for which the disparity value is derived,

– the width tBlWidth and the height tBlHeight of the block in a texture view component for which the disparity value is derived,

– the view\_id value srcViewId of the texture view component for which the disparity value is derived, and

– the view\_id value refViewId of the reference view for the disparity value.

Output of this process is a disparity value dispVal.

The derivation of the disparity value dispVal is specified with the following ordered steps.

1. The variables depthX, depthY, blWidth and blHeight are specified as follows:

dHM = depth\_hor\_mult\_minus1 + 1  
dVM = depth\_ver\_mult\_minus1 + 1  
depthX = Clip3( DepthCropLeftCoord, DepthCropRightCoord,   
 ( ( textureX + grid\_pos\_x[ srcViewId ] )\* dHM ) >> depth\_hor\_rsh )  
depthY = Clip3( DepthCropTopCoord, DepthCropBottomCoord,   
 ( ( textureY + grid\_pos\_y[ srcViewId ] ) \* dVM ) >> depth\_ver\_rsh )  
depthXN = Clip3( DepthCropLeftCoord, DepthCropRightCoord,   
 ( ( textureX + grid\_pos\_x[ srcViewId ] + tBlWidth – 1 ) \* dHM ) >> depth\_hor\_rsh ) (J-31)  
depthYN = Clip3( DepthCropTopCoord, DepthCropBottomCoord,   
 ( ( textureY + grid\_pos\_y[ srcViewId ] + tBlHeight – 1 ) \* dVM ) >> depth\_ver\_rsh )  
blWidth = depthXN – depthX + 1  
blHeight = depthYN – depthY + 1

1. The variable maxDepth is specified as follows:

maxDepth = INT\_MIN  
for( j = 0; j < blHeight; j += ( blHeight – 1 ) )  
 for( i = 0; i < blWidth; i += ( blWidth – 1 ) ) (J-32)  
 if( depthPic[ depthX + i, depthY + j ] > maxDepth )   
 maxDepth = depthPic[ depthX + i, depthY + j ]

1. The variable dispVal is specified as follows:

log2Div = BitDepthY + 6  
srcIndex = ViewIdTo3DVAcquisitionParamIndex( srcViewId )  
refIndex = ViewIdTo3DVAcquisitionParamIndex( refViewId ) (J-33)  
dispVal = ( NdrInverse[ maxDepth ] \* DisparityScale[ dps\_id ][ srcIndex ][ refIndex ] +   
 ( DisparityOffset[ dps\_id ][ srcIndex ][ refIndex ] << BitDepthY ) +   
 ( 1 << ( log2Div – 1 ) ) ) >> log2Div

J.8.2.1.2 Depth-based derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb\_type is equal to P\_Skip, nal\_unit\_type is equal to 21, DepthFlag is equal to 0, dmvp\_flag is equal to 1 and MbVSSkipFlag is equal to 0.

Outputs of this process are:

– the motion vector mvL0,

– the reference index refIdxL0.

For the derivation of the motion vector mvL0 and refIdxL0 of a P\_Skip macroblock type, the following ordered steps are specified:

1. The process specified in subclause is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag equal to 0 as input and the output is assigned to the motion vector mvL0 and the reference index refIdxL0.
2. When refIdxL0 is equal to -1, the following applies:

– The reference index refIdxL0 is set to 0.

– The derivation process for luma motion vector prediction in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, refIdxL0, and currSubMbType = “na” as the inputs and the output being mvL0.

J.8.2.1.3 Derivation process for luma motion vectors for VSP skipped macroblocks in P and SP slices

This process is invoked when mb\_type is equal to P\_Skip, nal\_unit\_type is equal to 21, DepthFlag is equal to 0, and MbVSSkipFlag is equal to 1.

Inputs to this process are current macroblock partition index mbPartIdx.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The inverse macroblock scanning process as specified in subclause 6.4.1 is invoked with CurrMbAddr as the input and the output is assigned to ( x1, y1 ).

The inverse macroblock partition scanning process specified in subclause 6.4.2.1 is invoked with mbPartIdx as the input and the output assigned to ( dx1, dy1 ).

The reference index refIdxL0 for a VSP skipped macroblock is derived as the inter-view picture that appears first in RefPicList0.

The variable refViewId is set equal to the view\_id of the inter-view picture refIdxL0.

The depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, textureX equal to x1 + dx1, textureY equal to y1 + dy1, tBlWidth equal to 16, tBlHeight equal to 16, srcViewId equal to view\_id and refViewId equal to refViewId as inputs and the output assigned to mvL0[ 0 ].

mvL0[ 1 ] is set equal to 0.

J.8.2.1.4 Derivation process for luma motion vectors for B\_Skip, B\_Direct\_16x16, and B\_Direct\_8x8

Inputs to this process are current macroblock partition index mbPartIdx and subMbPartIdx.

Outputs of this process are the reference indices refIdxL0, refIdxL1, the motion vectors mvL0 and mvL1, the motion vector count variable subMvCnt, and the prediction list utilization flags, predFlagL0 and predFlagL1.

For the derivation of output, the following ordered steps are specified:

1. Let the variable currSubMbType be set equal to sub\_mb\_type[ mbPartIdx ].
2. The process specified in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType and listSuffixFlag set equal to 0 as input and the output is assigned to the motion vector mvL0 and the reference index refIdxL0.
3. The process specified in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType and listSuffixFlag set equal to 1 as input and the output is assigned to the motion vector mvL1 and the reference index refIdxL1.
4. When both reference indices refIdxL0 and refIdxL1 are equal to -1, the following applies:

– The reference index refIdxL0 is set equal to 0.

– The derivation process for luma motion vector prediction in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, refIdxLX (with X being 0 or 1), and currSubMbType as the inputs and the output being mvLX.

J.8.2.1.5 Derivation process for the motion vector in inter-view reference

Inputs to this process are mbPartIdx, subMbPartIdx, and listSuffixFlag.

Outputs of this process are the motion vector mvCorrespond and the reference index refIdxCorrespond.

Inter-view reference picture InterViewPic and an offset vector dV are derived as follows:

– The following applies to derive an inter-view reference picture or inter-view only reference picture InterViewPic and to set the variable interViewAvailable:

interViewAvailable = 0  
for( cIdx = 0; cIdx <= num\_ref\_idx\_l0\_active\_minus1 && !interViewAvailable; cIdx++ )  
 if ( view order index of RefPicList0[ cIdx ] is equal to 0 ) {  
 InterViewPic = RefPicList0[ cIdx ] (J-34)  
 interViewAvailable = 1  
 }

– If interViewAvailable is equal to 1 and TextureFirstFlag is equal to 0, the following ordered steps apply:

– The inverse macroblock scanning process as specified in subclause 6.4.1is invoked with CurrMbAddr as the input and the output is assigned to ( x1, y1 ).

– The inverse macroblock partition scanning process specified in subclause 6.4.2.1 is invoked with mbPartIdx as the input and the output assigned to ( dx1, dy1 ).

– The inverse sub-macroblock partition scanning process specified in subclause 6.4.2.2 is invoked with mbPartIdx and subMbPartIdx as the input and the output assigned to ( dx2, dy2 ).

– the depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, textureX equal to x1 + dx1 + dx2, textureY equal to y1 + dy1 + dy2, tBlWidth equal to the width the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx, tBlHeight equal to the height the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx, srcViewId equal to view\_id and refViewId equal to the view\_id of InterViewPic as inputs and the output assigned to dV[ 0 ] and dV[ 1 ] is set to 0.

– Otherwise (interViewAvailable is equal to 1 and TextureFirstFlag is equal to 1), the following ordered steps apply:

– The process specified in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and with reference indices refIdxCandL0[ i ] and the motion vectors mvCandL0[ i ] as outputs with i equal to 0, 1, and 2 corresponding to neighbouring partition A, B, and C, respectively.

– The process specified in subclause  is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 1 as input and with reference indices refIdxCandL1[ i ] and the motion vectors mvCandL1[ i ] as outputs with i equal to 0, 1, and 2 corresponding to neighbouring partition A, B, and C, respectively.

- dV is derived as specified by the following ordered steps:

– Set DvAvailable[ i ] and mvCand[ i ] with i equal to 0, 1, and 2 corresponding to neighbouring partitions A, B, and C, respectively, as follows

for( i = 0; i < 3; i++ )  
 if( view order index of RefPicList0[ refIdxCandL0[ i ] ] is equal to 0 ) {  
 DvAvailable[ i ] = 1  
 mvCand[ i ] = mvCandL0[ i ]  
 } else if( view order index of RefPicList1[ refIdxCandL1[ i ] ] is equal to 0 ) {   
 DvAvailable[ i ] = 1  
 mvCand[ i ] = mvCandL1[ i ]  
 } else  
 DvAvailable[ i ] = 0

– If DvAvailable[ 0 ]+DvAvailable[ 1 ]+DvAvailable[ 2 ] is equal to 1, the following applies:

dV = mvCand[ i ][ 0 ]

-- Otherwise, the following steps apply in order:

For each value of i equal to 0, 1, and 2, when DvAvailable[ i ] is equal to 0, mvCand[ i ] is set to DvMVX.

– Each component of the variable dV is derived as follows:

dV = Median( mvCand[ 0 ][ 0 ], mvCand[ 1 ][ 0 ], mvCand[ 2 ][ 0 ] )

The refIdxCorrespond and mvCorrespond are set as follows.

– If interViewAvailable is equal to 0, refIdxCorrespond is set to -1, and mvCorrespond[ 0 ] and mvCorrespond[ 1 ] are both set to 0.

– Otherwise, the following step applies in order.

– The variable luma4x4BlkIdx is derived as (4 \* mbPartIdx + subMbPartIdx).

– The inverse 4x4 luma block scanning process as specified in subclause is invoked with luma4x4BlkIdx as the input and ( x, y ) as the output. In addition, ( xCorrespond, yCorrespond ) is set equal to ( x + ( dV[ 0 ] >> 4 ), y + ( dV[ 1 ] >> 4 ) ) and mbAddrCorrespond is set equal to ( ( CurrMbAddr / PicWidthInMbs ) + ( dV[ 1 ] >> 6 ) ) \* PicWidthInMbs + ( CurrMbAddr % PicWidthInMbs ) + ( dV[ 0 ] >> 6 ).

– Set mbTypeCorrespond to the syntax element mb\_type of the macroblock with address mbAddrCorrespond inside the picture InterViewPic. When mbTypeCorrespond is equal to P\_8x8, P\_8x8ref0, or B\_8x8, subMbTypeCorrespond is set to be the syntax element sub\_mb\_type of the macroblock with address mbAddrCorrespond inside the picture InterViewPic.

– Set mbPartIdxCorrespond to the macroblock partition index of the corresponding partition and subMbPartIdxCorrespond to the sub-macroblock partition index of the corresponding sub-macroblock partition. The derivation process for macroblock and sub-macroblock partition indices as specified in subclause is invoked with the luma location equal to ( xCorrespond, yCorrespond ), the macroblock type equal to mbTypeCorrespond, and when mbTypeCorrespond is equal to P\_8x8, P\_8x8ref0, or B\_8x8, the list of sub-macroblock types subMbTypeCorrespond as the inputs and the outputs are the macroblock partition index mbPartIdxCorrespond and the sub-macroblock partition index subMbPartIdxCorrespond.

– The motion vector mvCorrespond and the reference index refIdxCorrespond are derived as follows.

– If the macroblock mbAddrCorrespond is coded as Intra prediction mode, both components of mvCorrespond are set equal to 0 and refIdxCorrespond is set equal to –1.

– Otherwise (the macroblock mbAddrCorrespond is not coded as Intra prediction mode), the prediction utilization flags predFlagLXCorrespond is set equal to PredFlagLX[ mbPartIdxCorrespond ], the prediction utilization flag of the macroblock partition mbAddrCorrespond\mbPartIdxCorrespond of the picture InterViewPic. In addition, the following applies.

– When predFlagLXCorrespond is equal to 1, the mvCorrespond and the reference index refIdxCorrespond are set equal to MvLX[ mbPartIdxCorrespond ][ subMbPartIdxCorrespond ] and RefIdxLX[ mbPartIdxCorrespond ], respectively, which are the motion vector mvLX and the reference index refIdxLX that have been assigned to the (sub-)macroblock partition mbAddrCorrespond\mbPartIdxCorrespond\subMbPartIdxCorrespond inside the picture InterViewPic.

J.8.2.1.6 Derivation process for luma motion vectors for VSP skipped/direct macroblocks in B slices

Inputs to this process are current macroblock partition index mbPartIdx.

Outputs of this process are the motion vector mvL0, mvL1 and the reference index refIdxL0, refIdxL1.

The inverse macroblock scanning process as specified in subclause 6.4.1is invoked with CurrMbAddr as the input and the output is assigned to ( x1, y1 ).

The inverse macroblock partition scanning process specified in subclause 6.4.2.1 is invoked with mbPartIdx as the input and the output assigned to ( dx1, dy1 ).

The reference index refIdxLX for a VSP skipped/direct macroblock is derived as the inter-view reference component that appears first in the reference picture list X, with X being replaced by 0 or 1. When there is no inter-view picture in the reference picture list X, refIdxLX is set equal to 0.

The variable refViewIdX is set equal to the view\_id of the inter-view reference component refIdxLX

The motion vector mvLX, with X being replaced by 0 or 1, is derived as follows.

– The depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, textureX equal to x1 + dx1, textureY equal to y1 + dy1, tBlWidth equal to the width the macroblock partition mbPartIdx, tBlHeight equal to the height the macroblock partition mbPartIdx, srcViewId equal to view\_id and refViewId equal to the refViewIdX as inputs and the output assigned to mvLX[ 0 ].

– mvLX[ 1 ] is set equal to 0.

J.8.2.1.7 Derivation process for luma motion vector prediction

Inputs to this process are:

– the macroblock partition index mbPartIdx,

– the sub-macroblock partition index subMbPartIdx,

– the reference index of the current partition refIdxLX (with X being 0 or 1),

– the variable currSubMbType.

Output of this process is the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The specifications of subclause 8.4.1.3 apply with the following changes.

– The following additional sentence is applied

If refIdxLX is not equal to refIdxLXN for any N = A, B, or C and X equal to 0 or 1, the following applies:

mbAddrN\mbPartIdxN\subMbPartIdxN is marked as not available  
refIdxLXN = -1  
mvLXN 0 ] = 0 (J-35)  
mvLXN[ 1 ] = 0

after the following paragraph in subclause 8.4.1.3:

The derivation process for the neighbouring blocks for motion data in subclause  is invoked with mbPartIdx, subMbPartIdx, currSubMbType, and listSuffixFlag = X (with X being 0 or 1 for refIdxLX being refIdxL0 or refIdxL1, respectively) as the input and with mbAddrN\mbPartIdxN\subMbPartIdxN, reference indices refIdxLXN and the motion vectors mvLXN with N being replaced by A, B, or C as the output.

– The following additional sentence is applied

– Otherwise, if refIdxLX is a reference index to an inter-view reference component or an inter-view only reference component, the depth-based derivation process for median luma motion vector prediction in subclause is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C, and refIdxLX as the inputs and the output is assigned to the motion vector predictor mvpLX.

– Otherwise, if refIdxLX is a reference index to a reference picture which is not an inter-view reference component or an inter-view only reference component, the depth-based derivation process for median luma temporal motion vector prediction in subclause is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C, and refIdxLX as the inputs and the output is assigned to the motion vector predictor mvpLX.

after the following paragraph in subclause 8.4.1.3

– Otherwise, if MbPartWidth( mb\_type ) is equal to 8, MbPartHeight( mb\_type ) is equal to 16, mbPartIdx is equal mvpLX = mvLXCto 1, and refIdxLXC is equal to refIdxLX, the motion vector predictor mvpLX is derived by:

mvpLX = mvLXC

J.8.2.1.7.1 Depth-based derivation process for median luma motion vector prediction

Inputs to this process are:

– the neighbouring partitions mbAddrN\mbPartIdxN\subMbPartIdxN (with N being replaced by A, B, or C),

– the motion vectors mvLXN (with N being replaced by A, B, or C) of the neighbouring partitions,

– the reference indices refIdxLXN (with N being replaced by A, B, or C) of the neighbouring partitions,,

– the reference index refIdxLX of the current partition.

Output of this process is the motion vector prediction mvpLX.

When either partition mbAddrN\mbPartIdxN\subMbPartIdxN is not available or refIdxLXN is not equal to refIdxLX, mvLXN is derived as specified by the following:

– If TextureFirstFlag is equal to 0, the following steps apply in order:

1. The inverse macroblock scanning process as specified in subclause 6.4.1is invoked with CurrMbAddr as the input and the output is assigned to ( x1, y1 ).

2. The inverse macroblock partition scanning process specified in subclause 6.4.2.1 is invoked with mbPartIdx as the input and the output assigned to ( dx1, dy1 ).

3. The inverse sub-macroblock partition scanning process specified in subclause 6.4.2.2 is invoked with mbPartIdx and subMbPartIdx as the input and the output assigned to ( dx2, dy2 ).

4. The modification process of inter-view motion vector in median luma motion vector prediction as specified in subclause is invoked with depthPic being equal to DepthRefPicList0[ refIdxL0 ], mbx1 being equal to x1 and mby1 being equal to y1 as inputs and the output is assigned to the motion vector mvLXN.

– Otherwise (TextureFirstFlag is equal to 1), mvLXN is set equal to ( DvMBX , 0 ).

Each component of the motion vector prediction mvpLX is given by the median of the corresponding vector components of the motion vector mvLXA, mvLXB, and mvLXC:

mvpLX[ 0 ] = Median( mvLXA[ 0 ], mvLXB[ 0 ], mvLXC[ 0 ] ) (J-36)  
mvpLX[ 1 ] = Median( mvLXA[ 1 ], mvLXB[ 1 ], mvLXC[ 1 ] ) (J-37)

If TextureFirstFlag is equal to 1, DvMBX is set to as follows:

DvMBX = mvpLX[0]+mvd\_lX[mbPartIdx][subMbPartIdx][0]J.8.2.1.7.1.1 Modification process for inter view motion vector in median luma motion vector prediction

Inputs to this process are

– depth reference view component depthPic,

– the location of a top-left sample ( mbx1, mby1 ) of the current macroblock.

Outputs of this process are:

– the motion vector mv.

Let refViewId be the view\_id value of depthPic.

The variable mv is derived as follows:

– The depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, textureX equal to mbx1, textureY equal to mby1, tBlWidth equal to 16, tBlHeight equal to 16, srcViewId equal to view\_id and refViewId equal to the refViewId as inputs and the output assigned to mv[ 0 ].

– mv[ 1 ] is set equal to 0.

J.8.2.1.7.2 Depth-based derivation process for median luma temporal motion vector prediction

Inputs to this process are:

– the neighbouring partitions mbAddrN\mbPartIdxN\subMbPartIdxN (with N being replaced by A, B, or C),

– the motion vectors mvLXN (with N being replaced by A, B, or C) of the neighbouring partitions,

– the reference indices refIdxLXN (with N being replaced by A, B, or C) of the neighbouring partitions,

– the reference index refIdxLX of the current partition.

Output of this process is the motion vector prediction mvpLX.

When either partition mbAddrN\mbPartIdxN\subMbPartIdxN is not available or refIdxLXN is not equal to refIdxLX, mvLXN is derived as specified by the following ordered steps:

1. When TextureFirstFlag is equal to 0, the inverse macroblock scanning process as specified in subclause 6.4.1is invoked with CurrMbAddr as the input and the output is assigned to ( x1, y1 ).

2. When TextureFirstFlag is equal to 0, the inverse macroblock partition scanning process specified in subclause 6.4.2.1 is invoked with mbPartIdx as the input and the output assigned to ( dx1, dy1 ).

3. When TextureFirstFlag is equal to 0, the inverse sub-macroblock partition scanning process specified in subclause 6.4.2.2 is invoked with mbPartIdx and subMbPartIdx as the input and the output assigned to ( dx2, dy2 ).

4. When TextureFirstFlag is equal to 0, the process specified in subclause  is invoked with depthPic set to DepthCurrPic, mbx1 set to x1, mby1 set to y1 and listSuffixFlag as input and InterViewPic, an offset vector dV and a variable interViewAvailable as outputs.

5. When TextureFirstFlag is equal to 1, dV is set equal to ( DvMBX , 0 ) and a variable interViewAvailable is set equal to InterViewRefAvailable.

6. The refIdxCorrespond and mvCorrespond are set as follows.

– If interViewAvailable is equal to 0, refIdxCorrespond is set to -1, and mvCorrespond[ 0 ] and mvCorrespond[ 1 ] are both set to 0.

– Otherwise, the following steps apply in order.

– The variable luma4x4BlkIdx is set equal to ( 4 \* mbPartIdx + subMbPartIdx ).

– The inverse 4x4 luma block scanning process as specified in subclause is invoked with luma4x4BlkIdx as the input and ( x, y ) as the output. In addition, ( xCorrespond, yCorrespond ) is set equal to ( x + ( dV[ 0 ] >> 4 ), y + ( dV[ 1 ] >> 4 ) ) and mbAddrCorrespond is set equal to ( ( CurrMbAddr / PicWidthInMbs ) + ( dV[ 1 ] >> 6 ) ) \* PicWidthInMbs + ( CurrMbAddr % PicWidthInMbs ) + ( dV[ 0 ] >> 6 ).

– Set mbTypeCorrespond to the syntax element mb\_type of the macroblock with address mbAddrCorrespond inside the picture InterViewPic. When mbTypeCorrespond is equal to P\_8x8, P\_8x8ref0, or B\_8x8, subMbTypeCorrespond is set to be the syntax element sub\_mb\_type of the macroblock with address mbAddrCorrespond inside the picture InterViewPic.

– Set mbPartIdxCorrespond to the macroblock partition index of the corresponding partition and subMbPartIdxCorrespond to the sub-macroblock partition index of the corresponding sub-macroblock partition. The derivation process for macroblock and sub-macroblock partition indices as specified in subclause is invoked with the luma location equal to ( xCorrespond, yCorrespond ), the macroblock type equal to mbTypeCorrespond, and when mbTypeCorrespond is equal to P\_8x8, P\_8x8ref0, or B\_8x8, the list of sub-macroblock types subMbTypeCorrespond as the inputs and the outputs are the macroblock partition index mbPartIdxCorrespond and the sub-macroblock partition index subMbPartIdxCorrespond.

– The motion vector mvCorrespond and the reference index refIdxCorrespond are derived as follows.

– If the macroblock mbAddrCorrespond is coded as Intra prediction mode, both components of mvCorrespond are set equal to 0 and refIdxCorrespond is set equal to –1.

– Otherwise (the macroblock mbAddrCorrespond is not coded as Intra prediction mode), the prediction utilization flags predFlagLXCorrespond is set equal to PredFlagLX[ mbPartIdxCorrespond ], the prediction utilization flag of the macroblock partition mbAddrCorrespond\mbPartIdxCorrespond of the picture InterViewPic. In addition, the following applies.

– When predFlagLXCorrespond is equal to 1, the mvCorrespond and the reference index refIdxCorrespond are set equal to MvLX[ mbPartIdxCorrespond ][ subMbPartIdxCorrespond ] and RefIdxLX[ mbPartIdxCorrespond ], respectively, which are the motion vector mvLX and the reference index refIdxLX that have been assigned to the (sub-)macroblock partition mbAddrCorrespond\mbPartIdxCorrespond\subMbPartIdxCorrespond inside the picture InterViewPic.

7. The motion vectors mvLXN is derived as follows.

– If refIdxCorrespond is equal to refIdxLX, the following applies:

mvLXN[ 0 ] = mvCorrespond[ 0 ]  
mvLXN[ 1 ] = mvCorrespond[ 1 ] (J-38)

– Otherwise, the following applies:

mvLXN[ 0 ] = 0  
mvLXN[ 1 ] = 0

8. The following applies for the derivation of mvpLX[ 0 ] and mvpLX[ 1 ]:

mvpLX[ 0 ] = Median( mvLXA[ 0 ], mvLXB[ 0 ], mvLXC[ 0 ] ) (J-39)  
mvpLX[ 1 ] = Median( mvLXA[ 1 ], mvLXB[ 1 ], mvLXC[ 1 ] ) (J-40)

J.8.2.1.7.2.1 Derivation process for the disparity vector and the inter-view reference

Inputs to this process are depth reference view component depthPic, the location of a top-left sample ( mbx1, mby1 ) of the current macroblock and the listSuffixFlag.

Outputs of this process are a picture InterViewPic, an offset vector dV and a variable interViewAvailable

The variable interViewAvailable is set equal to 0.

The following applies to derive an inter-view reference picture or inter-view only reference picture, InterViewPic, with X set to 1 when listSuffixFlag is 1 or 0 otherwise:

for( cIdx = 0;cIdx<num\_ref\_idx\_l0\_active\_minus1 + 1 && !interViewAvailable; cIdx ++)  
 if ( view order index of RefPicList0[ cIdx ] is equal to 0) {  
 InterViewPic = RefPicList0[ cIdx ] (J-41)  
 interViewAvailable = 1  
 }

When interViewAvailable is equal to 1, the depth-based disparity value derivation process in subclause is invoked with depthPic equal to DepthCurrPic, textureX equal to mbx1, textureY equal to mby1, tBlWidth equal to 16, tBlHeight equal to 16, srcViewId equal to view\_id and refViewId equal to view\_id of InterViewPic as inputs and the output assigned to dV.

J.8.2.1.8 Neighbouring block based disparity vector derivation process

Input to this process is a macroblock currMB.

Let the variable availableDvFlag equal to 0, ( xP, yP ) be equal to the output of the subclause 6.4.2.1 (the location of upper-left luma sample for currMB partition 0).

The variables dvMBCur and DvMBX are derived as specified by the following ordered steps:

1. For each X from 0 to 1, the following steps apply in order.

– When availableDvFlag is equal to 0 and RefPicListY[ 0 ] is available (with Y equal to 1-X), the following applies:

– Set refPicListCol0 to the reference picture list 0 of RefPicListY[ 0 ].

– mvColL0 and refIdxColL0 are set to the motion vector mvL0 and reference index refIdxL0 that have been assigned to the block covering ( xP + 16, yP + 16 ) in picture RefPicListY[ 0 ], respectively.

– When refPicListCol0[ refIdxColL0 ] is available, the view order index of refPicListCol0[ refIdxColL0 ] is unequal to the view\_idx, and mvColL0[ 0 ] is unequal to 0, dvMBCur is set to mvColL0[ 0 ] and availableDvFlag is set to 1.

2. When availableDvFlag is equal to 0, dvMBCur is set to DvMBX.

3. DvMBX is set to dvMBCur.