H.7.3.8.5

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { | Descriptor |
| … |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| if ( iv\_res\_pred\_flag[ nuh\_layer\_id ] && RpRefPicAvailFlag ) |  |
| **~~iv\_res\_pred\_weight\_idx~~** | ~~ae(v)~~ |
| **adaptive\_arp\_idx** | ae(v) |
| … |  |
| } |  |

H.7.4.9.5

**~~iv\_res\_pred\_weight\_idx~~** ~~specifies the index of the weighting factor used for residual prediction. iv\_res\_pred\_weight\_idx equal to 0 specifies that residual prediction is not used for the current coding unit. iv\_res\_pred\_weight\_idx not equal to 0 specifies that residual prediction is used for the current coding unit. When not present, the value of iv\_res\_pred\_weight\_idx is inferred to be equal to 0.~~

~~When DefaultDispFlag[ x0 ][ y0 ] is equal to 1, iv\_res\_pred\_weight\_idx shall be equal to 0.~~

**adaptive\_arp\_idx** specifies the index of the disparity vector or motion vector used for residual prediction.adaptive\_arp\_idx equal to 0 specifies that residual prediction is not used for the current coding unit. adaptive\_arp\_idx not equal to 0 specifies that residual prediction is used for the current coding unit. When not present, the value of adaptive\_arp\_idx is inferred to be equal to 0. The variable AdaptiveArpIdx is set equal to adaptive\_arp\_idx – 1.

**H. 8.5.3.2.1**

**…**

6. The merging candidate list, extMergeCandList, is constructed as follows:

…

if( availableFlagVSP && !ic\_flag && ~~iv\_res\_pred\_weight\_idx~~ adaptive\_arp\_idx = = 0 )

extMergeCandList[ i++ ] = VSP

…

14. The variable vspModeFlag is derived as specified in the following:

vspModeFlag = mergeCandIsVspFlag && !ic\_flag && ( ~~iv\_res\_pred\_weight\_idx~~ adaptive\_arp\_idx = = 0 )

H.8.5.3.3.1

…

The variable resPredFlag is derived as specified in the following:

resPredFlag = ( adaptive\_arp\_idx != 0 ) ~~( iv\_res\_pred\_weight\_idx != 0 ) &&   
 ( PicOrderCnt( RefPicListX[ refIdxLX ] ) != PicOrderCntVal ) &&~~   
 RpRefPicAvailFlagLX &&  
 RefRpRefAvailFlag[ RefViewIdx[ xP ][ yP ] ]

…

H.8.5.3.3.7 Bilinear sample interpolation and residual prediction process

…

Depending on ivRefFlag and RpRefIdxLX, the following applies:

If ivRefFlag is equal to 0 and RpRefIdxLX is not equal to −1, the variable availFlag is set equal to 1, the variable refIdxLX is set equal to RpRefIdxLX, The derivation process for disparity vector candidates in residual prediction as specified in subclause  is invoked with the luma location ( xP, yP ), the variables nPbW and nPbH, the motion vector mvLX, the reference index refIdxLX and the index of the disparity vector AdaptiveArpIdx as the input; and the output are the disparity vector candidate dvCand and the reference view order index candidate refViewIdxCand. The residual prediction motion vector scaling process as specified in subclause is invoked with the prediction list utilization variable equal to X, the motion vector mvLX, and the RefPicListX[ refIdxLX ] and as inputs and modified mvLX as output.

Otherwise, when ivRefFlag is equal to 1, the following applies:

The derivation process for motion vector candidates in residual prediction as specified in subclause 7.6 is invoked with the luma location ( xP, yP ), the variables nPbW and nPbH, the variable mvLX, the variable RefPicListX [ refIdxLX ] and the index of the motion vector AdaptiveArpIdx as the input; and the output are the motion vector candidate mvCand, and the prediction list utilization variable Y.

…

Depending on ivRefFlag, the variables rpPic, rpRefPic, mvRp and curRefIdx are derived as specified in the following:

* + If ivRefFlag is equal to 0, the following applies:
    - Let rpPic be the picture with PicOrderCnt( rpPic ) equal to PicOrderCntVal and ViewIdx equal to refViewIdxCand.
    - Let rpRefPic be the picture with PicOrderCnt( rpRefPic ) equal to RefPicListX[ RpRefIdxLX ] ) and ViewIdx equal to refViewIdxCand,
    - The variable mvRp is set equal to dvCand.
    - The variable curRefIdx is set equal to RpRefIdxLX.
  + Otherwise (ivRefFlag is equal to 1), the following applies:
    - Let rpPic be the picture RefPicListY[ RpRefIdxLY ].
    - Let rpRefPic be the picture with PicOrderCnt( rpRefPic ) equal to PicOrderCnt( rpPic ) and ViewIdx equal to viewindex( RefPicListX [ refIdxLX ]).
    - The variable mvRp is set equal to mvCand.
    - The variable currRefIdx is set equal to RpRefIdxLY.

~~The variable shiftVal is set equal to ( iv\_res\_pred\_weight\_idx − 1 ).~~

...

* 1. predSamplesLXL[ x ][ y ] = predSamplesLXL[ x ][ y ] +   
     ( ( rpSamplesLXL[ x ][ y ] − rpRefSamplesLXL[ x ][ y ] )  ~~>>  shiftVal~~ )
  2. …
  3. predSamplesLXCb[ x ][ y ] = predSamplesLXCb[ x ][ y ] +   
     ( ( rpSamplesLXCb[ x ][ y ] − rpRefSamplesLXCb[ x ][ y ] )  ~~>>  shiftVal~~ )
  4. …
  5. predSamplesLXCr[ x ][ y ] = predSamplesLXCr[ x ][ y ] +   
     ( ( rpSamplesLXCr[ x ][ y ] − rpRefamplesLXCr[ x ][ y ] )  ~~>>  shiftVal~~ )
  6. …

H.8.5.5 Derivation process for disparity vectors

…

For i from 0 to NumDdvCandPics − 1, inclusive, the following applies when availableDV is equal to 0:

* The derivation process for a disparity vector from temporal neighbour block as specified in subclause H.8.5.5.1 is invoked with the luma location ( xCb, yCb ), the variable i and the variable nCbS as inputs, and the outputs are the flag availableDV, the disparity vector mvDisp and the reference view order index refViewIdx.

…

H.8.5.5.1 Derivation process for a disparity vector from temporal neighbour blocks

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 i specifying the collocated picture,
* a variable nCbS specifying the size of the current luma coding block.

…

The flag availableFlag is set equal to 0, and mvDisp is set equal to ( 0, 0 ).

~~For i from 0 to NumDdvCandPics − 1, inclusive, the following ordered steps apply and the whole decoding process of this sub-clause terminates once availableFlag is set to 1.~~

…

**H.8.5.3.3.7.5** **Derivation process for** **disparity vector candidates in residual prediction**

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a motion vector MV,
* a reference index candidate refIdxCandLX, with X being 0 or 1.
* a variable nDvIdx specifying the disparity vector candidate index.

Outputs of this process are

* the disparity vector candidate dvCand,
* the reference view order index candidate refViewIdxCand.

The variable currPic specifies the current picture.

The variable nDvFound is set equal to 0.

The motion vector candidate list dvList is empty.

The view order index list refViewIdxList is empty.

The motion vector MValgined[0] is set equal to ( 0,0 ).

The motion vector MValgined[1] is set equal to MV.

For isAlignedFlag = 1 and 0, the following applies

* + For iPic from 0 to NumDdvCandPics − 1, inclusive, the following ordered steps apply when nDvFound is less than nDvIdx+1

The derivation process for a motion vector scaling as specified in subclause H.8.5.8.1 is invoked with MValgined[isAlignedFlag], DdvCandPicsList[iPic], currPic and refPicListX [ refIdxCandLX ] as inputs, and the outputs is mvScaled.

The derivation process for a disparity vector from temporal neighbour block as specified in subclause H.8.5.5.1 is invoked with the luma location ( xCb, yCb )+ mvScaled, the variable iPic and the variable nCbS as inputs, and the outputs are the flag availableDVTmp, the disparity vector dvTmp and the reference view order index refViewIdxTmp.

If availableDVTmp is equal to 1, and (nDvFound == 0 || dvList[0] != dvTmp || refViewIdxList[0] != refViewIdxTmp) && dvTmp[0] != 0, then dvList [nDvFound] is set equal to dvTmp, refViewIdxList [nDvFound] is set equal to refViewIdxTmp, and nDvFound++.

For each N being A1, B1 and ( xN, yN ) being ( xCb − 1,  yCb + nCbS − 1 ), ( xCb + nCbS − 1,  yCb − 1 ), respectively, the following ordered steps apply.

* + The derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to the ( xCb, yCb ) and the luma location ( xN, yN ) as the input and the output assigned to availableN.
  + When availableN is equal to 1 and PredMode[ xN ][ yN ] is equal to MODE\_INTRA, availableN is set equal to 0.

For each N being A1, B1 and ( xN, yN ) being ( xCb − 1,  yCb + nCbS − 1 ), ( xCb + nCbS − 1,  yCb − 1 ), respectively, the following ordered steps apply.

* + For each X from 0 to 1, the following applies when nDvFound is less than nDvIdx+1:
    - If availableN is equal to 1, RefIdxLX[ xN ][ yN ] is greater than or equal to 0, PredFlagLX[ xN ][ yN ] is equal to 1, and RefPicListX[ RefIdxLX[ xN ][ yN ] ] is an inter-view reference picture of the current picture, the following applies:
      * If (nDvFound == 0 || dvList[0] != dvTmp || refViewIdxList[0] != refViewIdxTmp) && dvTmp != 0, then dvList[nDvFound] is set equal to MvLXN[ xN ][ yN ], refViewIdxList[nDvFound] is set equal to ViewIdx (RefPicListLX [RefIdxLX [ xN ][ yN ]] ), and nDvFound++.

For each N being A1, B1 and ( xN, yN ) being ( xCb − 1,  yCb + nCbS − 1 ), ( xCb + nCbS − 1,  yCb − 1 ), respectively, the following ordered steps apply when nDvFound is less than nDvIdx+1 and availableN is equal to 1:

* + - If depth\_refinement\_flag[ nuh\_layer\_id ]is equal to 1, dvTmp is set equal to MvRefinedDisp[ xN ][ yN ]; Otherwise, dvTmp is set equal to MvDisp[ xN ][ yN ].
    - If (nDvFound == 0 || dvList[0] != dvTmp || refViewIdxList[0] != refViewIdxTmp) && dvTmp != 0, then dvList[nDvFound] is set equal to dvTmp, refViewIdxList[nDvFound] is set equal to RefViewIdx[ xN ][ yN ], and nDvFound++.

If depth\_refinement\_flag[ nuh\_layer\_id ] is equal to 1, nDvFound is less than nDvIdx+1 and (nDvFound == 0 || dvList[0] != MvRefinedDisp[xCb][yCb] || refViewIdxList[0] != RefViewIdx[xCb][yCb]), then dvList[nDvFound] is set equal to MvRefinedDisp[xCb][yCb], refViewIdxList [nDvFound] is set equal to RefViewIdx[xCb][yCb], and nDvFound ++.

If nDvFound is equal to 0, then dvList[nDvFound] is set equal to (0,0), refViewIdxList [nDvFound] is set equal to 0, and nDvFound++.

If nDvFound is less than nMvIdx+1, then dvList[nDvFound] is set equal to dvList[0] + (4,0), refViewIdxList [nDvFound] is set equal to refViewIdxList[0] and nDvFound++.

If nDvFound is less than nMvIdx+1, then dvList[nDvFound] is set equal to dvList[0] + (-4,0), refViewIdxList [nDvFound] is set equal to refViewIdxList[0] and nDvFound++.

dvCand is set equal to dvList[nDvIdx] and refViewIdxCand is set equal to refViewIdxList [nDvIdx].

**H.8.5.3.3.7.6** **Derivation process for motion vector candidates in residual prediction**

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a disparity vector DV,
* a picture RefPic,
* a variable nMvIdx specifying the disparity vector candidate index.

Outputs of this process are

* the motion vector candidate mvCand,
* the prediction list indication candidate Y,

The view order index refViewIdx is set equal to viewindex(RefPic).

The variable currPic specifies the current picture.

The variable nMvFound is set equal to 0.

The motion vector candidate list mvList is empty.

The reference list indication list XList is empty.

The derivation process for a motion vector from a reference block for residual prediction as specified in subclause is invoked with ( xP, yP ), nPbW and nPbH, RefPic, and mvLX, as inputs, and availFlag, motion vector mvT and prediction list utilization variable Z as outputs. When availFlag is equal to 1, mvList[nMvFound] is set equal to mvT, XList[nMvFound] is set equal to Z and nMvFound++.

For each N being A1, B1 and ( xN, yN ) being ( xCb − 1,  yCb + nCbS − 1 ), ( xCb + nCbS − 1,  yCb − 1 ), respectively, the following ordered steps apply.

* + The derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to the ( xCb, yCb ) and the luma location ( xN, yN ) as the input and the output assigned to availableN.
  + When availableN is equal to 1 and PredMode[ xN ][ yN ] is equal to MODE\_INTRA, availableN is set equal to 0.
  + For each X from 0 to 1, the following applies when nMvFound is less than nMvIdx+1:
    - The flag mvFoundFlag is set equal to 0.
    - When availableN is equal to 1, RefIdxLX[ xN ][ yN ] is greater than or equal to 0, PredFlagLX[ xN ][ yN ] is equal to 1, and RefPicListX[ RefIdxLX[ xN ][ yN ] ] is a temporal reference picture of the current picture, then the following applies,
      * The derivation process for a motion vector scaling as specified in subclause H.8.5.3.3.7.3 is invoked with X, MvLXN[ xN ][ yN ] and refPicListX[RefIdxLX[ xN ][ yN ]] as inputs, and the outputs is mvScaled.
      * If ( nMvFound == 0 || mvList[0] != mvScaled ||  XList[0] != X ), then mvList[nMvFound] is set equal to mvScaled, XList[nMvFound] is set equal to X and nMvFound++.

The derivation process for temporal luma motion vector prediction in subclause is invoked with the luma location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the variable refIdxL0Col = RpRefIdxL0 as inputs, and the output being the availability flag availableFlagL0Col and the temporal motion vector mvL0Col.

If availableFlagL0Col is equal to 1, nMvFound is less than nMvIdx+1 and ( nMvFound == 0 || mvList[0] != mvScaled ||  XList[0] != X ), then mvList[nMvFound] is set equal to mvL0Col, XList[nMvFound] is set equal to 0, and nMvFound++.

If nMvFound is equal to 0, then mvList[nMvFound] is set equal to (0,0), XList[nMvFound] is set equal to 0, and nMvFound++.

If nMvFound is less than nMvIdx+1, then mvList[nMvFound] is set equal to mvList[0] + (4,0), XList[nMvFound] is set equal to XList[0], refIdxList[nMvFound] is set equal to refIdxList[0], and nMvFound++.

If nMvFound is less than nMvIdx+1, then mvList[nMvFound] is set equal to mvList[0] + (-4,0), XList[nMvFound] is set equal to XList[0], refIdxList[nMvFound] is set equal to refIdxList[0], and nMvFound++.

mvCand is set equal to mvList[nMvIdx ] and Y is set equal to XList[nMvIdx].

Table -10 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Syntax element** | **ctxIdxTable** | **initType** | | |
| **0** | **1** | **2** |
| coding\_unit( ) depth\_mode\_parameters( ) | … | … | … | … | … |
| ~~iv\_res\_pred\_weight\_idx~~  adaptive\_arp\_idx |  |  | 0..~~3~~4 | ~~4..7~~  0…4 |
| … | … | … | … | … |

Table ‑13 – Values of initValue for ~~iv\_res\_pred\_weight\_idx~~ adaptive\_arp\_idx ctxIdx

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of iv\_res\_pred\_weight\_idx** | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **~~5~~** | **~~6~~** | **~~7~~** |
| **initValue** | ~~162~~147 | 153 | 154 | ~~162~~154 | ~~162~~154 | ~~153~~ | ~~154~~ | ~~162~~ |

Table ‑20–Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

| **Syntax structure** | **Syntax element** | **Binarization** | |
| --- | --- | --- | --- |
| **Process** | **Input parameters** |
| coding\_unit( ) | ~~iv\_res\_pred\_weight\_idx~~  adaptive\_arp\_idx | TR | cMax = ~~2~~ 3, cRiceParam = 0 |

Table ‑23 – Specification of ctxIdxInc using left and above syntax elements

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **condL** | **condA** | **ctxIdxInc** |
| ~~iv\_res\_pred\_weight\_idx~~  adaptive\_arp\_idx | ~~iv\_res\_pred\_weight\_idx~~~~[ xL ][ yL ]~~  adaptive\_arp\_idx[ xL ][ yL ] | ~~i~~~~v\_res\_pred\_weight\_idx[ xA ][ yA ]~~  adaptive\_arp\_idx [ xA ][ yA ] | ( condL && availableL ) + ( condA && availableA ) |