H.7.3.8.5 General Coding unit syntax

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

H.7.4.9.5 Coding unit semantics

**~~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.8.5.3.3.1 General

…

* + - * The variable resPredFlag is derived as specified in the following:
        1. resPredFlag = ~~( iv\_res\_pred\_weight\_idx != 0 ) &&   
            ( PicOrderCnt( RefPicListX[ refIdxLX ] ) != PicOrderCntVal )~~

( adaptive\_arp\_idx != 0 ) (‑)

…

H.8.5.3.3.7 Bilinear sample interpolation and residual prediction process

…

Output of this process are:

* the (nPbW)x(nPbH) array predSamplesLXL,
* the (nPbW / 2)x(nPbH / 2) arrays predSamplesLXCb and predSamplesLXCr.

If PicOrderCnt(RefPicListX [ refIdxLX ]) is equal to PicOrderCntVal, the following ordered steps apply:

* + The variable DV is set equal to mvLX.
  + The variable DVC is set equal to mvCLX.
  + The variable refViewIdxSel is set equal to the reference view order index of RefPicListX [ refIdxLX ].
  + The derivation process for motion vector candidates in residual prediction as specified in subclause  is invoked with the luma location ( xP, yP ), the variables nPbW and nPbH, the variable DV, the variable refViewIdxSel and the index of the motion vector AdaptiveArpIdx as the input; and the output are the motion vector candidate mvCand, the prediction list indication candidate Y and the reference index candidate refIdxCandLY
    - X is set equal to Y.
    - mvLX is set equal to mvCand.
    - refIdxLX is set equal to refIdxCandLY
    - The derivation process for chroma motion vectors in subclause 8.5.2.1.8 is invoked with mvLX and refIdxLX as inputs and the output being mvCLX.

Otherwise, the following ordered step apply:

* + 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 variable DV is set equal to dvCand.
    - The variable DVC is set equal to dvCand.
    - The variable refViewIdxSel is set equal to refViewIdxCand.

The motion vector mvLX is scaled as specified in the following:

…

The arrays currIvSamplesLXL, currIvSamplesLXCb, and currIvSamplesLXCr  are derived as specified in the following:

* Let currIvRefPic be the picture with PicOrderCnt( currIvRefPic ) equal to PicOrderCntVal and ViewIdx equal to ~~RefViewIdx[ xP ][ yP ]~~ refViewIdxSel.
* Let the reference picture sample arrays currIvRefPicLXL, currIvRefPicLXCb, and currIvRefPicLXCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause 8.7 for the previously-decoded picture currIvRefPic.
* The arrays currIvSamplesLXL, currIvSamplesLXCb, and currIvSamplesLXCr are derived by invoking the bilinear sample interpolation process specified in subclause with the luma locations ( xCb, yCb ) and ( xBl, yBl ), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX equal to ~~MvDisp[ xP ][ yP ]~~ DV and mvCLX equal to ~~MvDisp[ xP ][ yP~~] DVC, and the, and the reference arrays with currIvRefPicLXL, currIvRefPicLXCb and currIvRefPicLXCr as the inputs.

The arrays refIvSamplesLXL, refIvSamplesLXCb, and refIvSamplesLXCr are derived as specified in the following:

* Let refIvRefPic be the picture with PicOrderCnt( refIvRefPic ) equal to PicOrderCnt( RefPicListX[ 0 ] ) and ViewIdx equal to ~~RefViewIdx[ xP ][ yP ]~~ refViewIdxSel.
* Let the reference picture sample arrays refIvRefPicLXL, refIvRefPicLXCb, and refIvRefPicLXCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause 8.7 for the previously-decoded picture refIvRefPic.
* The arrays refIvSamplesLXL, refIvSamplesLXCb, and refIvSamplesLXCr are derived by invoking the bilinear sample interpolation process specified in subclause with the luma locations ( xCb, yCb ), ( xBl, yBl ), the luma prediction block width nPbW, the luma prediction block height nPbH,, the motion vector mvLX equal to ( mvLX + ~~MvDisp[ xP ][ yP ]~~ DV ) and the motion vector mvCLX equal to ( mvCLX + ~~MvDisp[ xP ][ yP ]~~ DVC ), and the reference arrays with refIvRefPicLXL, refIvRefPicLXCb and refIvRefPicLXCr as the inputs.

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

…

predSamplesLXL[ x ][ y ] = predSamplesLXL[ x ][ y ]+   
( ~~(~~  currIvSamplesLXL[ x ][ y ] − refIvSamplesLXL[ x ][ y ] ~~)>>  shiftVal~~) (‑214)

...

predSamplesLXCb[ x ][ y ] = predSamplesLXCb[ x ][ y ] +   
(~~(~~currIvSamplesLXCb[ x ][ y ] − refIvSamplesLXCb[ x ][ y ]~~)  >>  shiftVal~~) (‑215)

…

predSamplesLXCr[ x ][ y ] = predSamplesLXCr[ x ][ y ] +   
( ~~(~~currIvSamplesLXCr[ x ][ y ] − refIvSamplesLXCr[ x ][ y ] ~~)  >>  shiftVal~~) (‑216)

…

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.8.5.6** **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 view order index refViewIdx,
* 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 reference index candidate refIdxCandLY, with Y being 0 or 1.

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 reference index list refIdxList is empty.

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

The motion vector DVoffset[1] is set equal to ( nPbW \*2 + 4, nPbH \*2 + 4 ),

For iLocation = 0 and 1, the following applies

* + For X in the range of 0 to 1, inclusive, the following applies:
    - The derivation process for a temporal inter-view motion vector candidate as specified in subclause  is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, the prediction list indication X, the view order index refViewIdx and the disparity vector DV + DVoffset [ iLocation ] as the inputs and the outputs are the flag availableFlagLXIvMC, the motion vector mvLXIvMC and the reference index refIdxLXIvMC.
    - If availableFlagLXIvMC is equal to 1, nMvFound is less than nMvIdx+1 and there is no K satisfying 0<=K<nMvFound && mvList[K]==mvLXIvMC && PicOrderCnt ( RefPicListLXList[K][ refIdxList[K]]) ==PicOrderCnt (RefPicListLX[ refIdxLXIvMC ] ), then mvList[nMvFound] is set equal to mvLXIvMC, XList[nMvFound] is set equal to X, refIdxList[nMvFound] is set equal to refIdxLXIvMC, 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:
    - 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 mvFoundFlag is set equal to 1;
    - If mvFoundFlag is equal to 1, nMvFound is less than nMvIdx+1 and there is no K satisfying 0<=K<nMvFound&&mvList[K]==MvLXN[ xN ][ yN ]&&PicOrderCnt ( RefPicListLXList[K] [refIdxList[K]])==PicOrderCnt(RefPicListLX[RefIdxLX[ xN ][ yN ]]), then mvList[nMvFound] is set equal to MvLXN[ xN ][ yN ], XList[nMvFound] is set equal to X, refIdxList[nMvFound] is set equal to RefIdxLX [ xN ][ yN ], 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 = 0 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 there is no K satisfying 0<=K< nMvFound && mvList[K] == mvL0Col && PicOrderCnt( RefPicListLXList[K][ refIdxList[K]]) == PicOrderCnt( RefPicListL0[0] ), then mvList[nMvFound] is set equal to mvL0Col, XList[nMvFound] is set equal to 0, refIdxList[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, refIdxList[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 ], Y is set equal to XList[nMvIdx], and refIdxCandLY is set equal to refIdxList[nMvIdx].

**H.8.5.7** **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.

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

The motion vector DVoffset[1] is set equal to ( nPbW \*2 + 4, nPbH \*2 + 4 )

For isAlignedFlag = 1 and 0, the following applies

* + For iPic from 0 to NumDdvCandPics − 1, inclusive, the following ordered steps apply
    - The motion vector mvScaled is derived as follows,
      * colPocDiff = DiffPicOrderCnt( currPic, refPicListX[ refIdxCandLX] );
      * currPocDiff = DiffPicOrderCnt( currPic, DdvCandPicsList[iPic] );
      * td = Clip3( −128, 127, colPocDiff );
      * tb = Clip3( −128, 127, currPocDiff );
      * tx = ( 16384 + ( Abs( td )  >>  1 ) ) / td;
      * distScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 )  >>  6 );
      * mvScaled = Clip3( −32768, 32767, Sign( distScaleFactor \* MValgined[isAlignedFlag]) \*   
        ( ( Abs( distScaleFactor \* MValgined[isAlignedFlag]) + 127 )  >>  8 ) ).
    - For iLocation = 0 and 1, the following applies:
      * 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+ DVoffset[iLocation], 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, nDvFound is less than nDvIdx+1 and there is no K satisfying 0<=K< nDvFound && dvList[K] == dvTmp && refViewIdxList[K] == refViewIdxTmp, 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:
    - The flag dvFoundFlag is set equal to 0.
    - 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, then dvFoundFlag is set equal to 1;
    - If dvFoundFlag is equal to 1, nDvFound is less than nDvIdx+1 and there is no K satisfying 0<=K< nDvFound && dvList[K] == MvLXN[ xN ][ yN ] && refViewIdxList[K] == ViewIdx( RefPicListLX [RefIdxLX[ xN ][ yN ]] ), 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.

* + - If availableN is equal to 1, nDvFound is less than nDvIdx+1 and there is no K satisfying 0<=K< nDvFound && dvList[K]==MvDisp[ xN ][ yN ]  && refViewIdxList[K] ==refViewIdxN, then dvList[nDvFound] is set equal to MvDisp[ xN ][ yN ], refViewIdxList[nDvFound] is set equal to refViewIdxN, and nDvFound++.

If depth\_refinement\_flag[ nuh\_layer\_id ] is equal to 1, nDvFound is less than nDvIdx+1 and there is no K satisfying 0<=K<nDvFound && dvList[K]==MvRefinedDisp[xCb][yCb] && refViewIdxList[K] == 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].

Table ‑12 – 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 ‑17 – Values of variable initValue for ~~iv\_res\_pred\_weight\_idx ctxIdx~~adaptive\_arp\_idx

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | **~~iv\_res\_pred\_weight\_idx~~** **adaptive\_arp\_idx** | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **~~5~~** | **~~6~~** | **~~7~~** |
| **initValue** | 154 | 154 | 154 | 154 | 154 | ~~154~~ | ~~154~~ | ~~154~~ |

Table ‑28 – 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 ‑32 – 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 ) |