H.7.3.9.1 General Coding unit syntax

|  |  |
| --- | --- |
| if( PartMode = = PART\_2Nx2N ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| if ( iv\_res\_pred\_flag[ nuh\_layer\_id ] && TempRefPicInListsFlag ) |  |
| **~~iv\_res\_pred\_weight\_idx~~** | ~~ae(v)~~ |
| **adaptive\_dv\_idx** | ae(v) |
| } else if( PartMode = = PART\_2NxN ) { |  |
| … |  |

H.7.4.9.1 General 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.~~

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

H.8.5.2.2 Decoding process for inter prediction samples

…

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

**…**

H.8.5.2.2.6 Bilinear sample interpolation and residual prediction process

…

The variable shiftVal is set equal to ~~( iv\_res\_pred\_weight\_idx − 1 )~~ 0.

…

H.8.5.4 Derivation process for disparity vectors

…

The flag availableDV is set equal to 0, and both components of the disparity vector mvDisp are set equal to 0.

The variable curDVNum is set equal to 0, refViewIdxList[i] is set equal to -1 for i=0,1,2, and both components of the disparity vector mvDispList[i] for i=0,1,2, are set equal to 0.

…

The derivation process for a disparity vector from temporal neighbour block as specified in is invoked with the luma location ( xC, yC ), and the variable nCS as inputs, and the outputs are the flag availableDV, the disparity vector mvDisp and the reference view order index refViewIdx.

If availableDV is equal to 1, the following applies in order,

mvDispList[curDVNum]=mvDisp,

refViewIdxList[curDVNum]=refViewIdx

curDVNum++.

When ~~availableDV is equal to 0,~~ curDVNum is lower than 3, for each N …

…

8. Set availableDV equal to 0. For each X from 0 to 1, the following applies:

When availableDV is equal to 0, availableN is equal to 1, RefIdxLX[ xN ][ yN ] is greater than or equal to 0, and PredFlagLX[ xN ][ yN ] is equal to 1, the following applies:

If RefPicListX[ RefIdxLX[ xN ][ yN ] ] is an inter-view reference picture, the following applies:

refViewIdx = ViewIdx( RefPicListX[ RefIdxLX[ xN ][ yN ] ] )

mvDisp = MvLXN[ xN ][ yN ]

~~availableDV = 1~~

If there is one index p which is equal to or greater than 0 and lower than curDVNum satisfying mvDisp is equal to mvDispList[p] and refViewIdx is equal to refViewIdxList[p], availableDV is set equal to 0. Else, the following applies:

mvDispList[curDVNum]=mvDisp,

refViewIdxList[curDVNum]=refViewIdx

curDVNum++

availableDV = 1

…

~~When availableDV is equal to 0,~~ for each N being A1, B1, B0, A0, and B2, the following applies.

When ~~availableDV is equal to 0~~ curDVNum is lower than 3, and availableFlagIvpMvN is equal to 1, the following applies:

mvDisp = ivpMvDispN

refViewIdx = refViewIdxN

~~availableDV = 1~~

If there is one index p which is equal to or greater than 0 and lower than curDVNum satisfying mvDisp is equal to mvDispList[p] and refViewIdx is equal to refViewIdxList[p], nothing is done. Else, the following applies:

mvDispList[curDVNum]=mvDisp,

refViewIdxList[curDVNum]=refViewIdx

curDVNum++

~~When availableDV is equal to 0,~~ ~~refViewIdx is set equal to 0, and mvDisp is set equal to ( 0, 0 ).The variable mvRefinedDisp is set equal to mv~~~~Disp.~~

When curDVNum is lower than 3 and depth\_refinement\_flag[ nuh\_layer\_id ]is equal to 1, the following ordered steps apply for i=0 to curDVNum-1:

* 1. The derivation process for disparity sample array as specified in subclause is invoked with the luma locations xC, yC, the disparity vector mvDispList[i], the view identifier refViewIdxList[i], the variable nPSW equal to nCS, the variable nPSH equal to nCS, the variable nSubBlkW equal to nCS, and the variable nSubBlkH equal to nCS, as the inputs, and the output is the array disparitySamples of size (nCS)x(nCS).
  2. The horizontal component of the disparity vector mvRefinedDispList[i][ 0 ] is set equal to disparitySamples[ 0 ][ 0 ].

When curDVNum is greater than 0 and lower than 3, the following applies.

If mvRefinedDispList[0] is not equal to mvDispList[0]

If curDVNum is equal to 1 or mvRefinedDispList[0] is not equal to mvDispList[1], the following applies:

mvDispList[curDVNum]= mvRefinedDispList[0]

mvRefinedDispList[curDVNum]= mvRefinedDispList[0]

refViewIdxList[curDVNum]= refViewIdxList[0]

curDVNum++

If curDVNum is equal to 2, mvRefinedDispList[1] is not equal to mvDispList[0] and mvRefinedDispList[1] is not equal to mvDispList[1], the following applies:

mvDispList[curDVNum]= mvRefinedDispList[1]

mvRefinedDispList[curDVNum]= mvRefinedDispList[1]

refViewIdxList[curDVNum]= refViewIdxList[1]

curDVNum++

When curDVNum is lower than 3, the following ordered steps apply:

* + 1. refViewIdxList[curDVNum] is set equal to 0, and mvDispList[curDVNum] is set equal to ( 0, 0 ).
    2. When depth\_refinement\_flag[ nuh\_layer\_id ] is equal to 1, the following ordered steps apply:
       1. The derivation process for disparity sample array as specified in subclause is invoked with the luma locations xC, yC, the disparity vector mvDispList[curDVNum], the view identifier refViewIdxList[curDVNum], the variable nPSW equal to nCS, the variable nPSH equal to nCS, the variable nSubBlkW equal to nCS, and the variable nSubBlkH equal to nCS, as the inputs, and the output is the array disparitySamples of size (nCS)x(nCS).
       2. The horizontal component of the disparity vector mvRefinedDispList [curDVNum][ 0 ] is set equal to disparitySamples[ 0 ][ 0 ]. The horizontal component of the disparity vector mvDispList[curDVNum][0] is set equal to disparitySamples[ 0 ][ 0 ].
       3. If there is one index p which is equal to or greater than 0 and lower than curDVNum satisfying mvDispList[curDVNum] is equal to mvDispList[p] and refViewIdxList[curDVNum]is equal to refViewIdxList[p], nothing is done. Else, curDVNum++.

When curDVNum is lower than 3, the following applies.

If mvRefinedDispList[0] is not equal to mvDispList[0]

If curDVNum is equal to 1 or mvRefinedDispList[0] is not equal to mvDispList[1], the following applies:

mvDispList[curDVNum]= mvRefinedDispList[0]

mvRefinedDispList[curDVNum]= mvRefinedDispList[0]

refViewIdxList[curDVNum]= refViewIdxList[0]

curDVNum++

If curDVNum is equal to 2, mvRefinedDispList[1] is not equal to mvDispList[0] and mvRefinedDispList[1] is not equal to mvDispList[1], the following applies:

mvDispList[curDVNum]= mvRefinedDispList[1]

mvRefinedDispList[curDVNum]= mvRefinedDispList[1]

refViewIdxList[curDVNum]= refViewIdxList[1]

curDVNum++

When curDVNum is lower than 3, the following applies:

K is set equal to 0. OffsetList[0] is set equal to -2, OffsetList[0] is set equal to 2.

The following applies until curDVNum is equal to 3

mvRefinedDispList [curDVNum] = mvDispList[curDVNum]= mvDispList[0]

refViewIdxList[curDVNum]= refViewIdxList[0]

mvDispList [curDVNum][ 0 ] = mvDispList [curDVNum][ 0 ] + OffsetList[K].

curDVNum++

K++

mvDisp is set equal to mvDispList[max(AdaptiveDvIdx,0) ].

refViewIdx is set equal to refViewIdxList[0].

mvRefinedDisp is set equal to mvRefinedDispList[1].

availableDV is set equal to 1.

…

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\_dv\_idx |  |  | 0..~~3~~4 | ~~4..7~~  0…4 |
| … | … | … | … | … |

Table ‑18 – Values of variable initValue for ~~iv\_res\_pred\_weight\_idx ctxIdx~~adaptive\_dv\_idx

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

Table ‑26 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ~~iv\_res\_pred\_weight\_idx~~  adaptive\_dv\_idx | 1 | TU, cMax = ~~2~~3 | 0 |  | 0 |
| 2 | 0 |  | 1 |

Table H‑31 – Specification of ctxIdxInc using left and above syntax elements

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